CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Computer Science

Bachelor's Thesis

Automatic analysis of worker bee behavior in the vicinity of the honeybee queen

Dominik Dvořáček
Open informatics

May 2022
Supervisor: doc. Ing. Tomáš Krajník, Ph.D.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Dvořáček Dominik**	Personal ID number:	**492169**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Computer Science**		
Study program:	**Open Informatics**		
Specialisation:	**Software**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Automatic analysis of worker bee behavior in the vicinity of the honeybee queen

Bachelor’s thesis title in Czech:

Automatická analýza chování včelích dělnic v blízkosti královny

Guidelines:

Description:
The aim of the project is to implement a system for automatic analysis of the surroundings of the honeybee queen. The system should be capable of detecting and localizing the honeybee queen and the court bees in her vicinity.
Requirements:
1) Learn (from the literature) about the basics of the honeybee behaviors and the interactions within the beehive colony.
2) Learn about the systems capable of automated detection, tracking and behavior analysis of the honeybees within the beehive.
3) Establish a set of key performance indicators (KPI) that characterize the performance of the aforementioned systems in the context of the project aim.
4) Select the most relevant method(s) and extend them so their performance in queen and court detection can be assessed.
5) Assess the method(s) performance using the selected KPIs and datasets consisting of videos captured in observation hives. Discuss the results.
6) Based on the results, integrate a pipeline for honeybee queen and court detection.

Bibliography / sources:

[1] Boenisch, Franziska, Benjamin Rosemann, Benjamin Wild, David Dormagen, Fernando Wario, and Tim Landgraf. 2018. “Tracking All Members of a Honey Bee Colony Over Their Lifetime Using Learned Models of Correspondence”. Frontiers in Robotics and AI 5 (April). https://doi.org/10.3389/frobt.2018.00035.
[2] Bozek, Katarzyna, Laetitia Hebert, Yoann Portugal, Alexander S. Mikheyev, and Greg J. Stephens. 2021. “Markerless tracking of an entire honey bee colony”. Nature Communications 12 (1). https://doi.org/10.1038/s41467-021-21769-1.
[3] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).
[4] Schmickl et al. The Queen and her Robotic Court: A Minimally-Invasive Form of Ecosystem Hacking: In International Science Fiction Prototyping Conference (SciFi-It' 2021).

Name and workplace of bachelor’s thesis supervisor:

doc. Ing. Tomáš Krajník, Ph.D. Artificial Intelligence Center FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **02.02.2022** Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

doc. Ing. Tomáš Krajník, Ph.D. Supervisor’s signature	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgement / Declaration

Many thanks to doc. Ing. Tomáš Krajník, Ph.D. for valuable advice during the elaboration of this thesis. Thanks also to Arash Sadeghi Amjadi, Ing. Tomáš Rouček and Bc. Jiří Ulrich for their help. I would also like to express my gratitude to my family and girlfriend for their support during the writing of this thesis.

The access to the computational infrastructure of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics" is also gratefully acknowledged.

I declare that I have written the submitted thesis independently and that I have listed all the information sources used in accordance with the Methodological Guideline on the Observance of Ethical Principles in the Preparation of University Theses.

Prague, 20 May 2022

....

Abstrakt / Abstract

Práce se zaobírá výběrem, implementací a hodnocením systémů schopných provádět automatickou detekci, sledování a klasifikaci včel v okolí královny. Systémy a metody řešení jsou popsané jak v teoretické rovině, tak z technického pohledu implementace a aplikace těchto metod na obrazové datasety získané z experimentální aparatury. Vybraný systém je rozšířen o schopnost detekce včel blízkých královně tzv. dvůr. Na základě výsledků experimentů je implementován výpočetní uzel, který rozšiřuje sestavenou pipeline založenou na systému ROS a určenou pro automatizovaný záznam a analýzu obrazových dat z pozorovacího úlu. Práce slouží jako příspěvek do mezinárodního výzkumného projektu RoboRoyale.

Klíčová slova: včela medonosná, Apis mellifera, počítačové vidění, detekce, sledování, strojové učení, robotika, ROS

Překlad titulu: Automatická analýza chování včelích dělnic v blízkosti královny

The thesis deals with selecting, implementing, and evaluating systems capable of performing automatic detection, monitoring, and classification of worker bees in the vicinity of the honeybee queen. The systems and methods of addressing the problem are described theoretically and from the technical point of view of the implementation and application of these methods to image datasets obtained from an experimental setup. The selected system is extended with the capability to identify bees in the queen's vicinity, called the court. Based on the experimental results, a computational node is implemented to extend the constructed ROS-based pipeline for automated recording and analysis of image data from the observation hive. The work contributes to the international research project RoboRoyale.

Keywords: honey bee, Apis mellifera, computer vision, detection, tracking, machine learning, robotics, ROS

Contents /

1 Introduction 1

2 RoboRoyale project 2

3 Honey bee biology 4
3.1 Behaviour 4
3.2 Behaviour in the vicinity of the queen 6
3.3 Bee biology in robotics 6

4 Image data extraction systems 8
4.1 Machine learning 8
4.2 Detection 9
4.2.1 Performance metrics 10
4.2.2 R-CNN 11
4.2.3 SSD 11
4.2.4 YOLO 12
4.2.5 YOLOR 13
4.3 Tracking 13
4.3.1 Performance metrics 14
4.3.2 SORT 15
4.3.3 DeepSORT 15
4.3.4 ByteTrack 15
4.4 Bee-specific methods 15
4.4.1 Marker-based methods . . . 16
4.4.2 Marker-less methods 17

5 Materials and Methods 19
5.1 Key performance indicators . . 19
5.1.1 Detection 19
5.1.2 Tracking 20
5.1.3 Court detection 20
5.1.4 Other system properties . . 20
5.2 Experimental setup 21
5.3 Robotic Operating System . . . 22
5.4 ROS pipeline 24
5.5 Datasets 24
5.6 Libraries used 27
5.7 Other software 27

6 Results 28
6.1 YOLO detector 28
6.2 Markerless bee detector 30
6.3 ByteTrack tracker 31
6.4 Markerless bee tracker 31
6.5 Court bee detection 31
6.6 ROS pipeline 33
6.6.1 Court bee detection node . 33
6.6.2 YOLO detection node . . . 34
6.6.3 Twitter node 34

7 Discussion 35
7.1 Selection of the methods 35
7.2 Comparison of systems 36
7.2.1 Detection 36
7.3 Tracking 36
7.4 Court bee detection 37
7.5 Possible future improvements . 37

8 Conclusion 39

References 40

A Attached files 47
A.1 Data 47
A.2 Code Base 47

Tables / Figures

6.1 YOLO model performance 29
6.2 Markerless detection model performance 31
6.3 ByteTrack tacking system performance 31
6.4 Markerless Tracker system performance 31
6.5 Court detection method performances 33
7.1 Detection assessment results 36
7.2 Tracking assessment results 37

2.1 Robot structure 2
2.2 Ecosystem hacking principle 3
3.1 Honeybee queen 4
3.2 Court event 6
4.1 R-CNN principle 11
4.2 YOLO detection principle 13
4.3 Demonstration of the markers with correction code 16
4.4 Demonstration of the markers without correction code 17
4.5 Detection pipeline of markerless tracking project 18
5.1 Setup photo 21
5.2 Setup diagram 22
5.3 ROS pipeline diagram 24
5.4 Dataset honeycomb image 25
5.5 Dataset crops (with and without annotation) 26
6.1 YOLO mAP scores 28
6.2 YOLO loss functions 29
6.3 YOLO detection visualization . 29
6.4 Training process of the Markerless detection 30
6.5 Court detection dataset 32
7.1 Court detection decision boundary 37

Chapter 1 Introduction

The honey bee (*Apis mellifera*) has been the focus of human attention since ancient times. Apart from the purely pragmatic interest in bee products, especially honey, it is also for ecological reasons. The honey bee is an irreplaceable pollinator in nature [1]. Another aspect of bee life that has fascinated biologists worldwide is the ability of bee colonies to function as a single unit. The modes of communication between individuals, the hierarchical structure of the colony, the changes in roles depending on the age of the individual, and the ability of the colony to survive for long years despite the short life span of individual members, are just a short list of characteristics that are a constant subject of research.

However, the species is in danger. Numerous studies indicate that the number of bees on earth is steadily declining, and this process will affect our lives as well. This thesis contributes to the research efforts of experts from the international RoboRoyale project to reverse this negative trend and help the bee population thrive again. It seeks to achieve this by helping at the level of individual colonies, which involves designing a bio-hybrid system using robotics and computer science so that the behaviour of the bee society can be influenced positively.

The goal is to create robotic agents operating in the vicinity of the honeybee queen. To be able to understand the behaviour of these animals better, create sophisticated, continuously improving models of their activity or navigate robotic agents inside the beehive, appropriate positional data of individuals needs to be automatically extracted from the behaviour observed. This work is focused on the research and experimental testing of methods that address the described issue or are potentially applicable to tackle the problem. The selected methods are extended with a system for detecting bees from the vicinity of the queen (court bees) and integrated into a pipeline for the automatic collection of image data from the observation hive.

The thesis content is structured as follows. For the introduction, objectives and methods of the RoboRoyale project are presented and elaborated. A brief introduction to the biological fundamentals of honey bee behaviour, including the findings on which the project methodology is based, follows. The overlap of this biological discipline with robotics is also described. The next chapter presents the theoretical foundations of computer vision and machine learning that are used in the scope of the thesis. Also, in this chapter, the reader can find a description of the principle of each method considered for a potential application. In the Materials and Methods section, the selected key performance indicators are described, based on which the performance of the tested methods was assessed. This chapter also includes descriptions of the various approaches, systems, and datasets used or developed to meet the objective of the thesis. Subsequently, the results of the experiments are documented, explained, and put into context in the discussion. The systems are evaluated and compared.

Chapter 2
RoboRoyale project

RoboRoyale [2] is an interdisciplinary EU-funded research project focused on the development of a bio-hybrid system. The project aims to influence the behaviour of the biological superorganism in the form of the honey bee colony in a positive and informed way. Following the long-term decline in the number of pollinators in the ecosystem [3–4], especially bees, this project seeks to develop ways to help the colonies of this essential species. The core idea is to introduce micro-robots that should be able to operate autonomously in the hive and blend in with the colony. The project thus combines the latest findings in robotics, computer science, biology and biocompatibility research.

The main focus of the research is the honey bee queen. This individual is responsible for the reproductive performance of the entire colony and, therefore, its long-term prosperity. "*We plan to strengthen her egg laying activity, in order to create a non-fragmented, strong and thus also energetically efficient broodnest at the core of the colony. This will boost the colony's population growth, thus also its foraging activity and its survival [5].*" By directly influencing the queen, it should therefore be possible to affect the rest of the colony in a relatively non-invasive yet effective way.

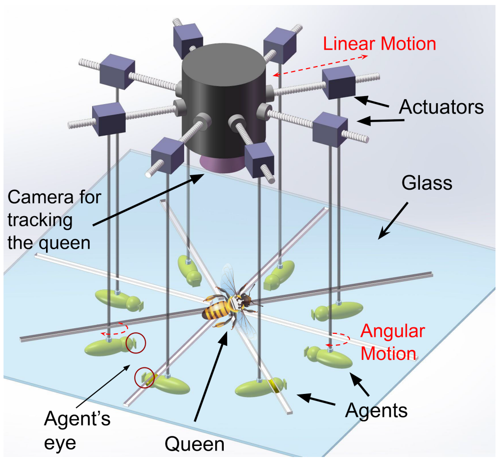

Figure 2.1. Internal structure of the proposed RoboRoyale system including eight worker bee agents [5].

Therefore, the project aims to introduce a multi-robot system operating in the vicinity of the queen. These robots should substitute some of the bees that operate near the queen. These individuals (so-called court bees) are responsible for feeding, grooming and cleaning the queen, as well as pheromone transmission and communication. Taking over some of these roles by a robotic system may allow the queen and, ultimately, the colony to change their behaviour. Proving this hypothesis would bring new insights to research on combined bio-robotic systems.

An essential capability of robots will be the transfer and provision of protein food to the queen. Conventional methods do not allow the possibility of controlling the amount of food. However, its impact on the colony can be substantial in terms of reproductive strength. Increased brood quantity should translate into a greater need for food and, therefore, stimulation of food foraging [2].

The behaviour of robotic agents plays a crucial role in their acceptance. In order to achieve this behaviour, a model needs to be developed to predict the future behaviour of court bees. A simplified model has already been presented by [6]. It predicts the movement of both the honeybee queen and her court bees. The model is based on two interconnected finite automata controlling the movement of the queen and the workers, respectively. Transitions between states are influenced by the state of the environment, in particular temperature differences in the hive and queen pheromone dispersal. The model depends on many parameters estimated from own empirical observations or earlier published findings. There was no significant difference from the observation hive data obtained by manual annotation in the statistical validation of the model simulation of the number of court bees. Also, a noticeable similarity between the simulation and the experimental recording can be found by empirical observation of the accumulated image. This model can serve as the basis for a more sophisticated machine learning-based model that will continuously improve even after the deployment of the robotic system, reflecting the various reactions of individuals to the behaviour of artificial agents [6].

The project aims to explore the so-called *Ecosystem Hacking* concept. The idea of stabilizing an ecosystem using active technological agents. There are several strategies to this problem, but the one implemented in this project is a minimally invasive approach using *Organismic Augmentation*, which involves artificially augmenting a selected organism with new capabilities. In the case of bees, the whole colony is considered as an organism [6].

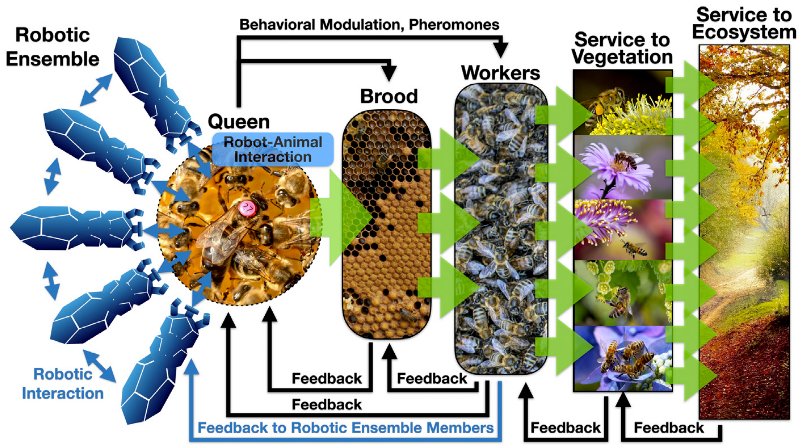

Figure 2.2. A demonstration of the principle of a minimally-invasive form of ecosystem hacking in the RoboRoyale project. The figure shows how robotic agents could indirectly affect the entire ecosystem [6].

The system that the project aims to develop would have many practical applications. Foremost it is the actual colony stabilization and continuous monitoring of the bees' health and condition. There is a possibility of improvement in the efficiency of the colony based on information obtained from long-term weather forecasts. In the event of a forecast of unfavourable weather, brood production could be reduced, lowering the energy investment in offspring with little chance of reaching maturity. Conversely, assuming the weather suitable for foraging, it would be possible to encourage brood production and thus prepare the colony for a period of optimal maximum activity [6].

Chapter 3
Honey bee biology

The honey bee (*Apis mellifera*) is a eusocial insect. Eusociality is the highest level of organization of sociality. It is characterized by cooperative brood care or division of labour between individual members. Therefore groups of specialized non-reproductive workers are formed, called *'castes'* [7]. Honey bee colonies can be described as a so-called superorganism. It can survive in nature for many years with constant population renewal.

The body of the bee is divided into three parts. On the *head* the sensory organs such as the compound eye and antennae are located. The mouth is equipped with mandibles. The *thorax* is the middle part of the body to which the wings and legs are attached. It is the locomotor centre of the bee body. The rear and longest part is called the *abdomen* [1].

3.1 Behaviour

The essential individual in the colony is the queen. This bee differs from the other castes and can be recognised by the elongated shape of its abdomen. The queen is the only bee in the hive with a fully developed reproductive system so that she can produce eggs with full genetic makeup - female eggs. Therefore, she is responsible for the colony's growth and expansion. A queen hatches from the same type of fertilised egg as a worker. However, she is fed royal jelly throughout the larval stage, unlike the worker larvae, which are fed a less nutritious diet from day three onwards. Approximately five days after the emergence, the queen leaves the hive for the so-called nuptial flight to attract male bees (*drones*) for mating. She usually mates with multiple drones to fill her *spermatheca*, from which she later fertilises the laid eggs. During peak activity, the queen can lay up to 2,500 eggs per day [1]. The bee queen can live up to 8 years [8].

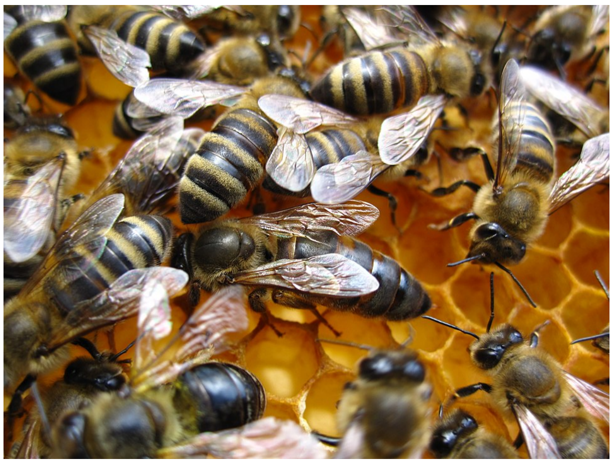

Figure 3.1. Honeybee queen (in the center) has longer abdomen, than worker bees *(Tashkoskim. Courtesy of wikimedia.org).*

Like the queen, worker bees grow from fertilised diploid eggs (which have a complete gene complement). There are four stages in the development of the bee: egg, larva, pupa and adult. The larvae hatch from the egg after about three days. For approximately five days, the developing larvae are fed by nurse bees, which take care of the brood. After that, the larvae cells are sealed with wax caps, and the pupas are formed. After about 13 more days, the developed individual chews through the wax cap.

The role (temporal caste) changes with the age of the worker bee. They usually work as nurse bees for the first ten days of their lives - cleaning the hive and caring for the brood. For approximately five days, their job is to build new combs. The worker bees serve as depositors of collected nectar by the twentieth day of life. Eventually, the bee becomes a forager. Usually, this role remains for the bee for the rest of her life. Working bees mostly perish before reaching senescence. According to research on bee survivorship [9], foraging bees face a constant probability of death per hour of foraging activity. *"However, if prevented from shifting to foraging activity they are recorded as having a maximum lifespan of 75–135 days. Over winter, worker bees can also develop into a stress-resistant form (called the "diunitus" stage) and their maximum adult lifespan is recorded as 140–320 days over winter [10]."*

It has been shown that bees have developed certain types of communication [11]. In the case of communication between foragers, bees communicate the location of the food they have found by means of so-called dances. The issue of bee communication has been studied in detail by Nobel Prize-winning physiologist Karl von Frisch, who divided the communication dances describing the food location into two types, the round dance and the waggle dance [12]. If the food source is close, the bee performs a round dance that does not contain information about the direction in which the source is located. Suppose the source is more than 100 meters away. In that case, the foraging bee performs a waggle dance, a figure-of-eight movement, which conveys information about both the direction and situation and the distance of the food source. However, recent studies show that bees have only one dance, which looks different depending on the distance of the source [11].

Pheromones play a crucial role in how colony members interact with each other. *"Pheromones are chemical substances secreted by an animal's exocrine glands that elicit a behavioral or physiological response by another animal of the same species [13]."* Bees have a highly complex system of pheromone communication. Every individual, regardless of caste, secretes a specific pheromone. The other bees receive the pheromones mainly through their antennae. There are many types of bee pheromones, and they have many functions, whether they are caste-specific pheromones or pheromones secreted only under certain circumstances (e.g. danger). An essential group of pheromones are the compounds secreted by the queen. These substances can have both short- and long-term effects on the entire population [14].

Bees can regulate temperature inside the hive. In the case of high temperatures, the bees perform ventilation by shivering their wings. In the winter months, when it is necessary to keep the internal temperature above a certain threshold, the bees form winter clusters where the temperature is kept around 20°C. The queen stops laying eggs and returns to laying again shortly after the winter solstice, depending on the weather conditions and geographical location [1].

3.2 Behaviour in the vicinity of the queen

An important aspect of worker bee behaviour for this thesis is the behaviour in the vicinity of the queen. According to previously published observations, specific patterns in this behaviour can be observed [15]. Insights suggest that the queen is not in direct contact with all colony members but only with a narrow subset of workers called court bees.

The bees residing in the vicinity of the queen frequently lick her body, taking up the specific pheromones she secretes. This scent is then passed on to other workers through food exchange. The sign of the queen's presence and the condition is thus spread throughout the hive. These bees are constantly cleaning the queen. It is in the colony's best interest for the queen to thrive, and it is the frequent cleaning that reduces the possibility of her becoming ill or infested with parasites [16].

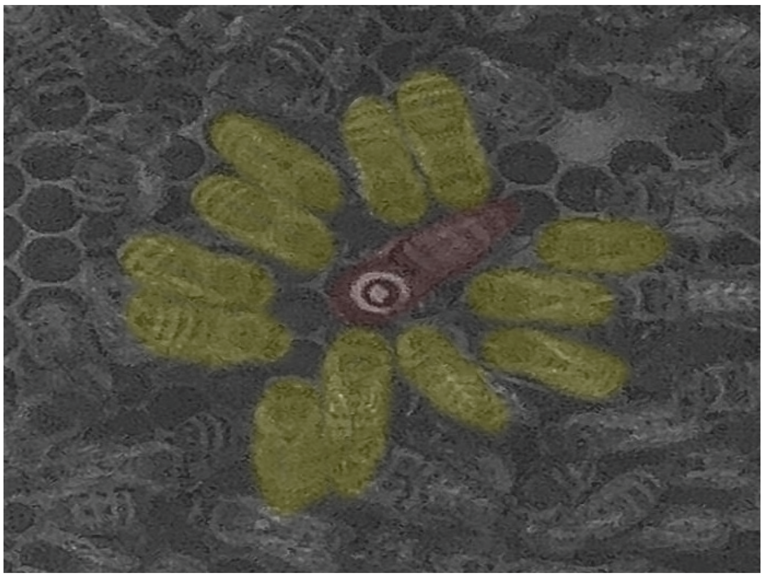

Figure 3.2. Court event recorded by the experimental setup used in the thesis to collect datasets. For illustration, the queen is highlighted in red and the court bees in yellow.

According to [15], the behaviour of the queen and the associated behaviour of her court can be divided into several states: idle, patrolling, receiving food and laying eggs. When the queen appears to be in an idle state, her movement is limited. Minimal movement is often associated with the aforementioned cleaning and licking of the queen. On the other hand, the patrolling state is associated with vigorous locomotion in which the queen transfers between brood areas. During feeding, the queen is provided with nutrition from the mouth of one of the court bees. In the case of egg-laying, the queen slides her abdomen into an empty cell and places an egg inside. New individuals hatch afterwards from these eggs.

3.3 Bee biology in robotics

Bee behaviour is the inspiration for many systems that address a wide range of problems. Many algorithms implementing different aspects of bee behaviour are part of the field of swarm intelligence. It refers to the activity of many independent agents whose sensing is limited to their surroundings. However, based on self-defined responses to various stimuli, the agents as a group can perform a specific task [17].

An area which is an extension and application of swarm intelligence is swarm robotics. It brings together multiple disciplines, engineering, computer science and biological research. The effort in this field is to bring the knowledge of swarm intelligence to the physical world through groups of robots. This approach has many obstacles but also many advantages. Robots operating in swarms are usually not equipped with powerful computing units, so the complexity of the algorithms needs to be minimised. At the same time, many problems are related to robotics in general, such as noise in communication or other limitations in sensory data. On the other hand, swarms of robots can be very resilient to the failures of individuals; they can react flexibly to environmental changes or cover larger areas, for example, in space exploration problems [18].

Several solutions implement elements of bee behaviour in swarm robotics as well. An example is the project of a swarm of foraging robots [19], which can find application in space exploration. Other examples can be systems like CosΦ [20] or Phormica [21], which seek to apply pheromone communication between agents using light-based non-chemical methods.

Chapter 4
Image data extraction systems

In this chapter, the principles of state-of-the-art methods that are used for image data acquisition are presented. In the context of this thesis, these are methods capable of distinguishing individual objects in the form of bees and determining their location - detection. Determining the position of particular objects over time is then dealt with by tracking methods. The systems presented here fall under the domain of computer vision. *"A computer vision system processes images acquired from an electronic camera, which is like the human visual system where the brain processes images derived from the eyes [22]."* Most of the methods described here are based on the machine learning approach, which will be explained in the following section.

4.1 Machine learning

Machine learning (ML) is an approach to programming in which a function (called a model) is derived from data provided and can be applied to data previously unseen. The data provided is called training data [23]. The principles on which ML is based have been well known for a relatively long time. Still, it was foremost with the development of GPUs and the associated massive parallelisation capability that this approach became mainstream.

Machine learning can be divided into three main paradigms: supervised learning, unsupervised learning, and reinforcement learning [23]. Which category a method falls into depends on the information that is attached to the training data. In the case of supervised learning, each sample is associated with a value from the model output space (called a label). In the case of unsupervised learning, no additional information is provided, and the algorithm is tasked with, for example, finding specific patterns or clusters in the provided data. Reinforcement learning deals with decision making when the associated value is delayed. Thus, the decision depends on the agent's reward after a series of steps [23].

An important concept of supervised learning, also for this paper, is classification. Classification is a problem where the model function is of the form $f\colon \mathcal{X} \to \mathcal{K}$, where $\mathcal{K}$ is a discrete set of classes $\mathcal{K} = \{0, \ldots, k\}$, so that the empirical error on the training data is minimised. $\mathcal{X}$ represents a set of observable features in the form of single values, vectors, images (matrices, tensors), or some other data format [23]. The resulting model is called a classifier.

In the following, selected supervised learning methods used in this work will be introduced. *K-nearest neighbours (KNN)* [24] is a method based on data classification based on the distance of each sample. A sample falls into the most abundantly represented class in the set of its k nearest neighbours. All dimensions must have a given metric in which distances can be calculated. The neighbour search is optimised using K-D trees [24]. The *Support Vector Machine (SVM)* method [25] is based on linear data separation in such a way that the gap width between samples from both classes is maximised. The method is robust to outliers, and despite being linear, it can be applied to linearly nonseparable data after applying the so-called kernel trick for dimensionality lifting [25].

Neural networks (NN) is a traditional machine learning concept that resembles the biological structure of the brain. It consists of multiple building blocks (neurons) that have the following structure:

$$y = g(\sum_{j=1}^{d} w_j x_j + b),$$

where x_j is the j-th component of the input vector $\mathbf{x}$, d is the number of dimensions of the input vector, w_j is the j-th component of the weight vector, and b is the so-called bias. The weights and biases are determined based on the input data in the learning process. The function g is a nonlinear activation function. Collections of neurons are often connected in multiple layers with different numbers of these nodes. The connection works on a one to each basis; therefore, such layers are called fully connected [23].

Convolutional neural networks (CNN) are widely used, especially for image processing [23]. Compared to fully connected NNs, they reduce the required number of weights by sharing them among neurons. Convolutional layers consist of so-called filters, which are these shared weights. Each neuron in a convolutional layer receives input only from a subset of the inputs from the previous layer (receptive field). Similar architectures using deep neural networks are often referred to as deep learning [23].

4.2 Detection

Detection is a branch of computer vision that focuses on image classification [26]. The system determines whether a given image contains a particular object or not. Depending on the specific method, the system then provides additional information such as the number of objects, their position in the image and other data (e.g. orientation in the context of bee recognition). An object, in the sense of image detection, is a class of objects, persons or animals that possesses common features such as colour or shape. [26].

Examples of traditional detection methods include the *Scale-Invariant Feature Transform (SIFT) [27]* or the *Histogram of Oriented Gradients (HOG) [28].* The SIFT method is able to extract points from an image that are invariant to rotation, displacement and resizing. Based on the extracted points, a classifier, here nearest-neighbour, decides the resulting class. The HOG divides the image into a square grid, and for each segment, a histogram of the intensity gradients is computed. Combining all histograms produces a feature vector describing the image. Detection then works based on classification using a machine learning method. Here the SVM method is used.

A noteworthy milestone in the field of image detection was the work of Paul Viola and Michael Jones [29], who applied the machine learning technique AdaBoost for fast face recognition in images. The methods used today; however, are commonly based on deep learning (see 4.2). We can divide them into two groups, namely two-stage and one-stage.

In the first stage of the inference process, two-stage algorithms use a neural network to create regions. Generated regions are forwarded for further processing. In the next stage, the classification of the found regions is performed, i.e., assigning classes and creating positional information (e.g. bounding boxes). Typical adopters of this class of systems are the R-CNN family of algorithms. On the other hand, one-stage detectors estimate positional information directly from the input image. They can generally be considered faster and are thus mainly used where fast real-time processing is required.

The YOLO family of algorithms and the SSD series of detectors can be mentioned as representatives [30].

4.2.1 Performance metrics

Two basic metrics are essential for measuring detection quality - precision and recall. These metrics are only related to binary decision making; thus, they are calculated for each class separately. The metrics are defined as follows:

$$Precision = \frac{TP}{TP + FP}, \qquad Recall = \frac{TP}{TP + FN}.$$

TP is the number of correctly detected objects, FP is the number of detected objects that belong to another class or are not objects, and FN is the number of undetected objects. Precision thus represents the quality of positive detections. The recall is the ratio of objects correctly detected by the classifier.

A commonly employed metric is the F1 score. The F1 is the harmonic mean of both precision and recall. Therefore, it takes into account both metrics. It is defined as ($\times$ refers to multiplication):

$$F_1 = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Different precision-recall pairs will be obtained by filtering the detection results according to their probability (as determined by the classifier). These pairs can be plotted in a graph for different probability thresholds. The resulting graph is called a precision-recall curve (here PR). The area under this graph is called the average precision (AP). Since the observations are discrete, the area is replaced by a weighted sum in practice. Also, the curve derived from the empirical data is typically not smooth. Thus, the evaluation of AP would be highly imprecise and more data sensitive. Therefore, interpolation methods are used to calculate AP. A widely used method is an 11-point interpolation. The resulting AP is then calculated as follows:

$$AP_{11} = \frac{1}{11} \sum_{R \in \{0, 0.1, \ldots, 0.9, 1\}} PR_{interp}(R),$$

$where$

$$PR_{interp}(R) = \max_{\tilde{R}: \tilde{R} \geq R} PR(\tilde{R}).$$

Thus, the evaluation of the curve is only performed at eleven evenly spaced points, with the appropriate value being sought from the threshold to the right. An alternative is an all-point interpolation, which performs the same interpolation process but for all recall values, regardless of their distribution [31].

The most commonly used metric for assessing the quality of a detection model is mean average precision (mAP). The mAP is determined simply as the mean of all average precisions over the individual classes part of the detection:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i.$$

In order to decide which detection category the detection belongs to (TP, TN, FP, FN), the intersection over union (IOU) method is used in bounding box detection systems. As the name implies, the calculation for a single detection is the area of intersection of the detection and ground-truth box to the area of their union:

$$IOU = \frac{Detection \cap GroundTruth}{Detection \cup GroundTruth}.$$

Thus, we obtain a measure of how well the detection fits the annotated data. A threshold is set to decide whether such a detection matches the truth. If the IOU of the detection is less than this threshold, it is taken as false. The results of all the fore-mentioned methods depend on the choice of this threshold [31].

4.2.2 R-CNN

The original R-CNN [32] system uses a selective search method to create so-called region proposals. These regions represent parts of the image where a classifiable object could be located. Each of the regions (about 2000) is cut and transformed to a specified size. The resulting images are forwarded to the convolutional neural network. The output is a 4096-dimensional vector characterising the object. A multi-class SVM [25] method is then used in the feature vector space for final classification. *"R-CNN made a breakthrough in object detection and improved detection accuracy on the VOC 2012 dataset by more than 30% [33]."*

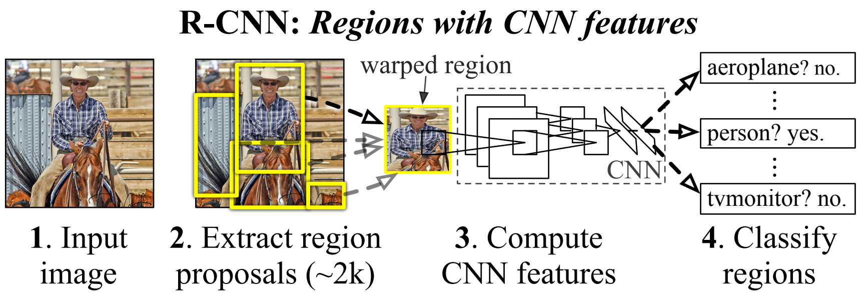

Figure 4.1. Overview of the R-CNN detection system [32].

In the following years, many improvements were developed that contributed to better accuracy but predominantly higher speed, which was a significant bottleneck for R-CNN. In 2015, the Fast R-CNN system [34] was introduced. The whole image is first fed to a convolutional neural network. The output is a map of convolutional features, from which regions are extracted using the Region of Interest pooling method. This improvement caused a drastic increase in inference speed without sacrificing the quality of the results.

In the same year, the Faster R-CNN system [35] was introduced to optimise the inference process further. Region proposals are no longer extracted using a pooling method but another neural network, which shares some layers with the feature extraction network. Further improvements to the R-CNN method were made in 2017 by the Mask R-CNN method, which is based on R-CNN [33].

4.2.3 SSD

The Single Shot MultiBox Detector [36] (SSD) no longer uses a phase where candidate regions are proposed, thus speeding up the inference and learning process considerably. In the first step, feature maps are extracted from the image. In the case of the method proposed, it is a VGG16 convolutional neural network. The input image is divided into a network of regular squares (locations). For a set of predefined bounding boxes and each location, the probability that a particular box contains a specific class is then derived for each class, including one reserved for *no class*. In addition to the individual confidence values for each class, the location and size of the detected bounding box relative to the predefined one is also output. The predefined final set of bounding boxes is generated in the learning phase based on K-means clustering. Thus, these boxes correspond to the usual sizes of all classes contained in the ground truth dataset. The grid of location squares is multiple with different sizes so that the system is able to detect objects with different dimensions [36]. Despite that, SSD does not achieve good results in detecting small objects [33].

The DSSD [37] enhancement introduced in 2017 uses the ResNet-101 architecture for feature map extraction while introducing further improvements to the detection step that lead to better performance, especially on small objects [33].

4.2.4 YOLO

In contrast to two-stage methods, YOLO [38] (You Only Look Once) views the detection problem as a regression problem in the space of bounding boxes and class confidence. It uses a single pass through a neural network for detection, which analyses the input image directly. Thus, YOLO exhibits a significant jump in speed compared to previous methods [38].

The image is first resized to a 448×448 px, then fed into the convolutional neural network. The network's outputs are individual bounding boxes, including the probabilities of each class. Eventually, those bounding boxes whose confidence exceeds a specified threshold are selected. The neural network function works based on partitioning the image into a 7×7 region grid. Bounding boxes of objects whose centre belongs to a particular segment are detected separately. The neural network consists of convolutional layers whose task is to extract features from each segment. This is followed by several fully connected layers, which are responsible for predicting the boxes and class probabilities [38].

The loss function of the YOLO algorithm is composed of three functions that take into account multiple optimised aspects. Bounding box regression loss represents the variation in box positions and sizes, object loss reflects the certainty with which the predictions are made, and the classification loss considers the error in the class assignment for multi-class detectors [39].

The first version of the YOLO algorithm had many drawbacks in the form of inaccuracy and struggle with objects at segment boundaries. Therefore, an improvement was introduced a year later with the YOLOv2 [40] algorithm. For example, in this version, the fully connected layers responsible for box prediction were replaced by a system using the initial box sizes from the training data, similar to the SSD method. Furthermore, class detection was tied to individual bounding boxes. These and other improvements dramatically increased the detection accuracy [33].

The next generation YOLOv3 [41] implements features such as binary cross-entropy loss function, multi-scale framework and feature-pyramid for greater adaptability to different sized objects. Furthermore, Darknet-53 architecture is used as the backbone of feature extraction, based on Darknet-19 and ResNet architectures. These improvements have led to further improvements in accuracy, especially on small objects [41].

Two additional versions of the YOLO algorithm were released in succession. Version 4 introduces many innovations, which include mosaic data enhancement, self-adversarial

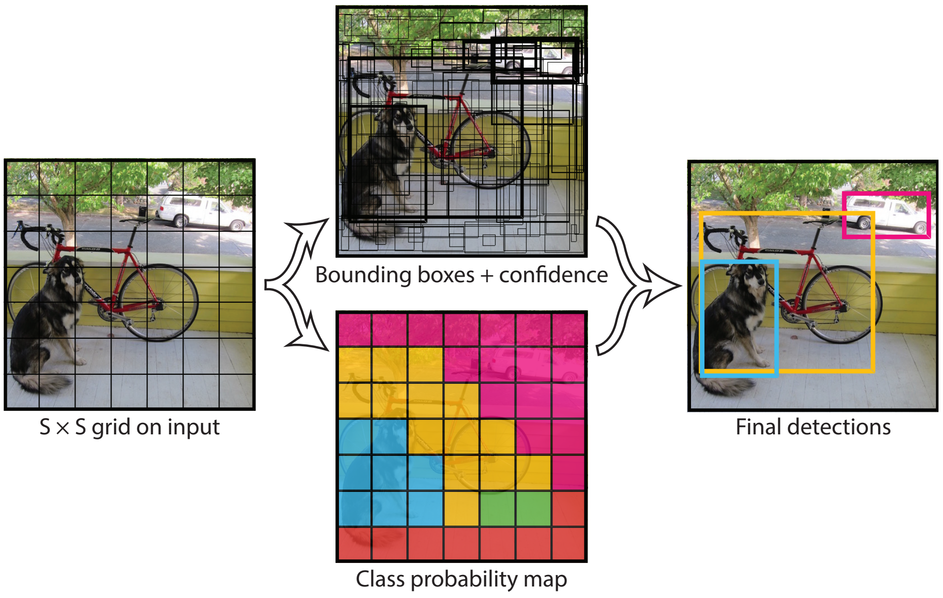

Figure 4.2. Simplified demonstration of the bounding box detection process using the YOLO method [38].

training and cross mini-batch normalisation [42]. YOLOv5 [43] does not introduce much novelty; however, it implements the system in Python using the PyTorch library, making integration easier for many authors.

4.2.5 YOLOR

You Only Learn One Representation [44] (YOLOR) is currently considered state-of-the-art in the field of real-time image detection. The authors of this method aim to create a multipurpose neural network that uses not only the knowledge learned from the training data designed for a given task (explicit knowledge) but also the subconscious knowledge (implicit knowledge) that the system acquires while learning different tasks.

While multi-task learning is still in a hypothesis state, the quality of the model for object detection has been demonstrated. When tested on the MS COCO benchmark dataset, YOLOR achieved comparable mean accuracy (mAP) to Scaled-YOLOv4 (an enhanced version of the v4 classifier) at twice the inference rate [44].

4.3 Tracking

Object tracking is a field of computer vision that deals with determining the location, path, and characteristics of an object of interest. *"The typical objectives of object tracking are the determination of the number of objects, their identities and their states, such as positions, velocities and in some cases their features [45]."*

A major challenge for multi-object detection systems is the various sources of uncertainty. Objects can move in unpredictable ways, their numbers changing over time. Camera images, or sensory measurements, are generally affected by noise, making the task more difficult. Especially for multi-object detection and dense object crowding detection, they face different types of occlusions that make subsequent identification difficult.

Object tracking methods can be classified according to several criteria. First of all, single-object, which, as the name suggests, deals with detecting only one object instance in a series of consecutive images, and muti-object detection, which extends the detection capabilities to multiple objects independent of each other [30]. In this thesis, we will mainly deal with multi-object detection systems.

The division between online and offline methods divides the systems according to sequence completeness requirements. Offline methods perform analysis on the entire video sequence. They can thus use additional information based on this fact. Offline methods are then often transformed into a graph search problem, where the vertices are single detections, and the edges are distances and time deviations between individual detections. On the other hand, online methods can be applied to real-time object tracking without knowledge of future situation evolution [46].

Another possible categorisation of methods is generative and discriminative. Traditional methods fall into the class of generative models. These methods include the *mean-shift* tracking method, *particle filter* and *Kalman filter [30]*. This chapter will discuss selected discriminative methods based on deep learning, which are currently considered state-of-the-art [46].

4.3.1 Performance metrics

Widely used metrics for evaluating the quality of a tracking system and for comparing individual methods are Multiple Object Tracking Accuracy (MOTA) and Identification F1-score (IDF1) [47].

The MOTA metric is defined as follows:

$$\text{MOTA} = 1 - \frac{\sum_t DetFN_t + DetFP_t + IDSw_t}{\sum_t N_t},$$

where $DetFN_t$ is the number of all miss-detected objects, $DetFP_t$ is the number of false positives for all classes, $IDSw_t$ is the number of identifier swaps, and N_t is the number of ground truth annotations; all at time t. The formula shows that detection errors easily overwhelm the errors caused by imperfections in the tracking algorithm, causing identifier swaps and dropouts. Despite that, the MOTA metric is the most widely adopted [47].

IDF1, in contrast to MOTA, gives more weight to correct tracklet joining over accurate detection. The calculation of this metric uses a bijective representation between ground-truth trajectories and the trajectories predicted by the method under test. This matching is optimal in the sense that it maximises the number of frames in which the matched trajectories overlap. This value is called IDTP (Identification True Positives). An overlap is counted if the IOU of the detection and prediction is greater than or equal to zero in a particular frame. The Hungarian algorithm [48] is used to optimise the matching [49]. Then the identification precision and recall can be defined:

$$\text{ID-Recall} = \frac{IDTP}{N}, \qquad \text{ID-Precision} = \frac{IDTP}{\hat{N}},$$

where N is the total number of ground truth boxes and $\hat{N}$ is the number of boxes in the predicted tracks. IDF1 itself is then defined as follows [47]:

$$\text{IDF1} = \frac{IDTP}{\frac{1}{2}(N + \hat{N})}.$$

The HOTA (Higher Order Tracking Accuracy) metric is also frequently used. Unlike the two methods presented here, it combines both detection and tracking quality fairly. At the same time, it can be decomposed into multiple metrics that focus on different aspects of the quality of the method output [49].

4.3.2 SORT

Simple Online and Realtime Tracking is a tracking method based on convolutional neural networks. FrRCNN (a variant of R-CNN) was employed as the detection backbone. The estimation model is based on the Kalman filter, and the identifier assignment is based on the IOU metric and optimised using the Hungarian algorithm [50]. SORT achieved 74.6 MOTA (76.9 IDF1) on the MOT17 dataset. Although it achieved results comparable to state-of-the-art methods at the time, it faced several problems (e.g., poor performance in an overlap setting) [51].

4.3.3 DeepSORT

DeepSORT is an improved version of the original SORT method. The authors replaced the association metric used with a metric that combines motion and appearance [52]. The authors were thus able to increase the amount of time an object could be located behind another object, greatly reducing the number of identity swaps. DeepSORT achieved 75.4 MOTA (77.2 IDF1) on the MOT17 dataset [51].

4.3.4 ByteTrack

The main idea behind ByteTrack is to use bounding boxes from the full range of confidence values as extracted from the detection method. Thus, it contrasts with previously developed methods that only analysed detection boxes with high object probability. Objects with lower confidence can often indicate the presence of an occluded object [53]. Byte track obtained 80.3 MOTA (77.3 IDF1) on the MOT17 dataset. ByteTrack represents the current state-of-the-art method for multi-object detection [51].

4.4 Bee-specific methods

Data collection from observation hives was a purely manual exercise in the past. *"Over the last decades, various aspects of the social interactions in honey bee colonies have been investigated with remarkable efforts in data collection [54]."* However, with relatively recent breakthroughs in computer vision and machine learning, these technologies are increasingly being used for research purposes in the animal kingdom as well.

An essential step in data extraction is the detection of individuals. It is necessary to determine the location of individual bees in a video recording of honeycomb activity. It is also desirable to determine the orientation of the colony members found, as this information is further used in the search for movement trajectories. Optional information extracted from the video footage can also be the size of the bee in the form of an oriented or non-oriented square bounding of the found object. The size information can be further used, for example, to distinguish between different types of bees, such as queens or drones.

Multiple object tracking is a challenging topic in computer vision. It has to deal with difficulties such as changing appearance, irregular motion, dynamic illumination, overlapping objects and also object substitution [55]. In the case of generating trails of individuals, it is necessary to link the separate occurrences marked and identified in the recognition phase in each image. Due to the problems mentioned above, detections of a given individual may be missing in some images or image sequences. Short sections of traces called tracklets are created. The task of the algorithm is to correctly combine the resulting segments so that the movement in time of the missing detections corresponds to the most probable state.

4.4.1 Marker-based methods

A frequently used method for analysing bee behaviour is the marking of individuals using distinctive markers. An example could be the experiment of Thomas D. Seeley [1], who used multicoloured numbered tags to experimentally study phenomena such as nectar collection and water collection. Manual tagging of bees is a tedious activity but shows many undeniable advantages. Bees are easily identifiable and safely recognised even after leaving and re-entering the hive. Furthermore, bees can be easily identified outside the observation hive, which paves the way for various field observations and monitoring of foraging. Apart from the requirement for manual tagging, a significant disadvantage of the described method is the necessity to tag new individuals constantly. If the method relies on identifiers with a finite number of unique values, this can hinder long-term observation. Another problem may be that extraneous markings interfere with the bees' natural colouring and may expose them to predators in the outdoor environment. At the same time, frequent entry into the hive to add markers for new individuals disrupts the colony and potentially affects the collected data. Despite that, marker methods have been used in a number of automated systems.

Several systems use manual marking of individual bees for identification using small fiducial markers containing coded images (similar to QR codes). Solutions such as [56] use a 26-bit correction code to minimise recognition errors. The fiducial marker system used in this solution is bCode. This allows the true bee identifier to be computed in the case of partial label occlusion. However, this approach requires the usage of an expensive industrial camera with very high resolution. The camera lens also plays an essential role, as high sharpness while maintaining the highest possible lens speed is required. Another disadvantage is that the dependence on correct tag detection is high, and images in which the bee is not detected are not considered. Thus, many tracklets (segments of the true trajectories) are created.

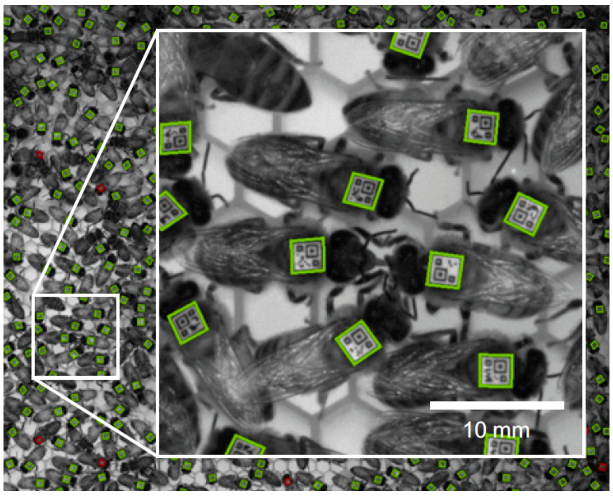

Figure 4.3. Demonstration of the application of bCode markers to detect and track bees to analyse social networks among colony members [56].

The system BeesBook presented in [54] dispenses with the need to add a correction code and allows more space for the identifier itself, thus reducing the demands on the camera resolution. It computes the probability of the identifier in the case of insufficient visibility from the spatio-temporal data of the last occurrence of the candidate label.

The tracking task in BeesBook consists of two phases. First, tracklets of consecutive detections are created. For each frame, the selection of the subsequent detection to ex-

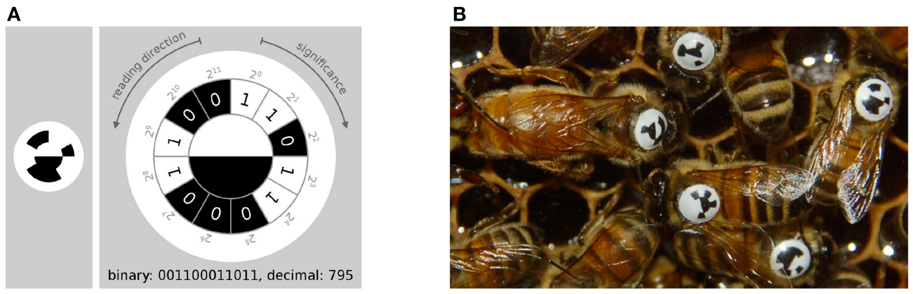

Figure 4.4. **A** Example of a tag used in BeesBook. The centre of the tag consists of two contrasting semicircles that define the bee's orientation, with the white semicircle pointing towards the head of the bee and the black one towards the abdomen. Around the rim is a series of twelve white and black segments that decode the bee's identification number (up to 4,096 possible identifiers). **B** Example of tag placement on the thorax of a bee [54].

tend an existing tracklet is made from the available detections up to a certain distance threshold. Three different features are then used for comparison, based on the positions of the detections being compared and the Manhattan distance probabilities of the bit identifiers. SVM [25] machine learning technique is then employed to obtain the probability that a pair of detections belongs to the same bee. The Hungarian algorithm [48] is then used to find the optimal extension of the tracklets. A tracklet is terminated if the optimal extension assignment has a probability of less than one half.

The second phase of tracking is merging the created tracklets. For this, the authors experimentally derive a set of six features that define the relation of the end of one tracklet to the start of a candidate tracklet. For example, this set contains the Euclidean distance between the end of one tracklet and the beginning of another, the forward and backward extrapolation distances error between the ends of candidate tracklets, or the difference in the angles of rotation of the bee at the end and the beginning of candidate tracklet fragments. The set also contains metrics that take into account the difference between the detected identifiers. Deciding whether one tracklet is a continuation of another is done by a machine learning classifier, this time a random forest classifier [57]. With the presented two-step method, the authors have shown a significant improvement in the quality and accuracy of the generated trajectories compared to a method that depends only on identifier recognition. The proportion of completed trajectories without errors was improved by 67%.

4.4.2 Marker-less methods

Advances in research and development of convolutional neural network architectures allow this technology to be used for both multi-object detection and pose detection and feature extraction for object identification [58]. Methods such as [59] have demonstrated that it is possible to use statistical machine learning methods to analyse bee behaviour without the use of markers, albeit on a smaller scale. Methods like [60] in turn show how convolutional neural networks can be used to identify objects, namely different species of small animals (ants and fish).

In the system presented in [58], the authors completely do away with the need for physically tagging bees, using a feature called 'pixel personality' to help with object identification. *"Importantly, previous work has shown that seemingly identical organisms do carry distinct visual features, also termed 'pixel personality', which can be quantified and leveraged for markerless tracking [58]."*

Detection works in two phases. It is based on a modified U-Net architecture [61] that was originally developed for biomedical image data segmentation. The model extracts regions of pixels that form a single individual. For each pixel, the output is the decided class membership and also the likely orientation angle of the bee. In the second stage of detection, the output values for each pixel are used to form clusters with statistically similar values. The orientation is then extracted using the PCA method as the first principal component of pixels belonging to one cluster. The body orientation is then decided by the mean value of the extracted angles belonging to a particular cluster.

For tracking, identification is again utilised together with the spatio-temporal orientation of the bee. At the start of the algorithm execution, all detections are treated as trajectories of length 1. The trajectories are then incrementally extended, considering only detections within a certain, dynamically determined distance from the end of the tracklet. The algorithm terminates if there are no more free detections not involved in any trajectory.

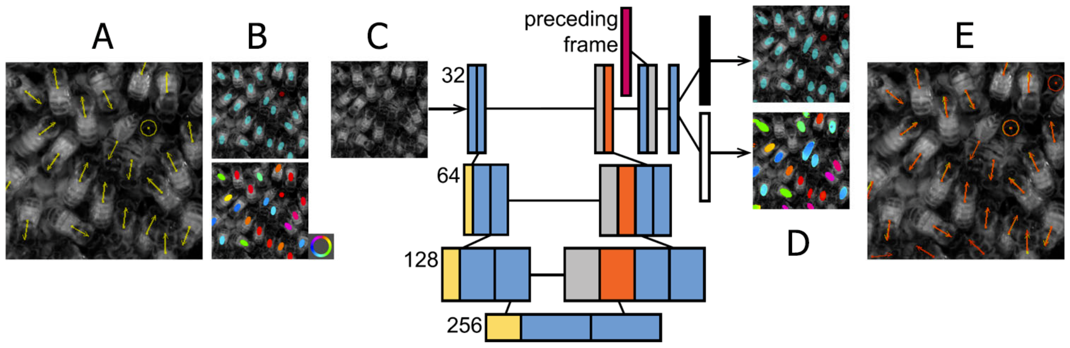

Figure 4.5. Example of the detection pipeline used by the neural network in the project for markerless bee tracking [58]. **A** Input annotated frame. **B** The Input annotations are used to create pixel regions, top: class definition, bottom: orientation definition (the colour assigned to the angle can be deducted from the colour circle). **C** Schematic of the U-Net architecture used, including an illustration of the enhancement of using the inference output from the previous frame as a prior. **D** Output evaluation of individual pixels. **E** Resulting detection produced by clustering the output regions.

Identification is based on feature extraction. The Inception V3 architecture [62] was chosen for this task. A 64-dimensional feature vector representing the particular bee is extracted from the cuts of all candidate detections. The Euclidean distance between the last detection vector in the extended trajectory and the candidate detection is taken into account in the spatio-temporal detection fusion algorithm.

According to the authors, the system's performance can be considered relatively high. 77% of the bees were correctly detected on a dataset upon which the system was not trained. On the training dataset, 70-86% of the bees were recorded as assigned to the correct trajectory using cross-validation.

Chapter 5
Materials and Methods

5.1 Key performance indicators

Key performance indicator (KPI) is a performance metric used mainly in business and economics. Established indicators assess the level of success of an organisation or project. *"KPIs act as a set of measures focusing on those sides of organisational performance that are critical for the success of the organization [63]."*

In general, KPIs can be thought of as metrics that characterise the performance of a system in crucial areas of the addressed problem. In the case of the systems compared in this thesis, the key performance metrics can be divided into three areas in which the systems are compared separately - detection, tracking and court bees detection.

5.1.1 Detection

The detection metrics chosen to assess the performance of the systems are either generic or proprietary. Because the outputs of the methods under review take various forms, it was necessary to establish a set of common metrics that fairly represent performance on the test dataset.

Precision/Recall and F1-score (see 4.2.1) on the set of detected bounding boxes were selected as suitable common metrics. Methods such as YOLO create rectangles that encode the area filled by the detected object with a certain probability. Markerless tracking [58], on the other hand, detects the centre of the bee and the orientation of its body. It also does not support computing the detection's confidence. The output detections from the markerless system can be converted into a set of bounding boxes based on the size of the bee and the elliptical shape of its body. In this way, both methods can be unified to the same output format, except for the detection probabilities. The aforementioned metrics do not depend on these probabilities and yet are richly indicative of the quality of bee detection. A detection is considered successful if its highest IOU value against all ground-truth boxes exceeds a specified threshold. A threshold of 0.5 was chosen for the evaluation. The code from the Treesfive/calculate-iou [64] project was used to calculate the intersection over union value between two bounding boxes.

The individual methods also offer additional metrics. However, these can be used to determine the quality of only a single method or compare the quality of different models of a single method. In the case of the YOLO detectors, this is mainly the mAP metric (see 4.2.1). This is based on the existence of probabilities of individual detections and thus gives a better indication of the quality of the detection independent of the chosen threshold for the probabilities. The Merkerless method offers several metrics used to calculate loss functions employed to optimise segmentation.

- Foreground overlap expresses the number of pixels detected by the segmentation model as part of a bee to the total number of pixels containing bees from the ground truth dataset.

- Background overlap defines the ratio of the number of segmented pixels not covered by bees to the number of background pixels from the ground truth dataset.
- Class error defines the number of misclassified pixels to the number of pixels in the image. Classification is performed between the visible bee, bee inside cell and background classes.
- Angle error, the average deviation of the detected angles from the ground truth angles.

  However, these metrics are only related to the segmentation extraction phase. Thus, they do not reflect the quality of the resulting detections derived from the segmentation.

5.1.2 Tracking

The traditional MOTA and IDF1 metrics (see 4.3.1) were chosen to evaluate the tracking quality. These methods are again based on bounding box evaluation. A threshold of 0.5 IOU is set to match the predicted bounding box and the ground truth box. As with the detection metrics, the output detections of the markerless method are converted to bounding boxes based on knowledge of bee size and elliptical body shape.

The markerless method offers as an additional metric the overlap between ground truth trajectories and the tracks detected [58].

5.1.3 Court detection

A machine learning approach was chosen for the classification of court bees. Thus, the basic KPIs chosen are again precision/recall and f1-score for each class, as well as the accuracy metric. Accuracy is defined as [65]:

$$\text{Accuracy}(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} 1(\hat{y}_i = y_i),$$

where y is the vector of ground truth classes of individual samples and $\hat{y}$ is the vector of their predictions. $n_{samples}$ is the total number of samples. $1()$ is a function that returns one if the argument is true and zero otherwise.

In the context of machine learning, we can also define the metric as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

Thus, the ratio of all correct predictions to all observations.

5.1.4 Other system properties

The presented systems also have other characteristics that have to be taken into account when comparing their performance.

- Inference speed - How long does it take to process an hourly record. Does the system support parallelisation? Is the system capable of real-time processing and online processing?
- Computational power requirements - How demanding the system is in terms of computational power and memory.
- Detection capability in case of bee overlap
- Adaptability to various conditions - for example, light conditions
- Difficulty of experiment preparation - How difficult is the data preparation/data collection? Is any special preparation required prior to observation (e.g. having to mark individual bees manually)?

5.2 Experimental setup

For the purpose of collecting experimental data, a laboratory setup was created. This apparatus was built in collaboration with the Artificial Life Lab, Institute of Biology, University of Graz. A recently renovated double room in the building of the University of Graz, with one observation hive, was allocated for the installation. The setup consists of a separate double-sided observation hive, which contains two observation combs on top of each other. The observation hive consists of a wooden frame and two transparent Plexiglas walls. An opening is located at the bottom of the wooden frame, which the bees can use to move freely between the hive and the outside environment. At a distance of 60 cm from the hive face, there is a wooden frame. Two lighting panels are fixed on the sides of the frame, each with six Synergy 21 LED Retrofit 4x1W IR lamps with a wavelength of 850 nm. Two Arducam B0274 camera modules with an IMX477 1/2.3 12.3MP (4056×3040) sensor and a removable IR cut filter are attached to each of the two frames. The lens has a fixed focal length of 6mm with speed F1.2 and variable aperture. The cameras are directly connected to Nvidia Jetson Nano or Nvidia Jetson Xavier microcomputers that perform the data pre-processing. The entire installation, including the hive, is covered with black non-woven fabric to minimise light fluctuations and eliminate distracting elements that could affect bee behaviour and data quality. The collected data is then sent via a shared router to the master computer (CPU: AMD Ryzen 9 5900X 12-Core, GPU: Nvidia GeForce GTX 980, RAM: 2 $\times$ Corsair Vengeance LPX 32GB DDR4) for subsequent analysis and saving to a remote NAS server.

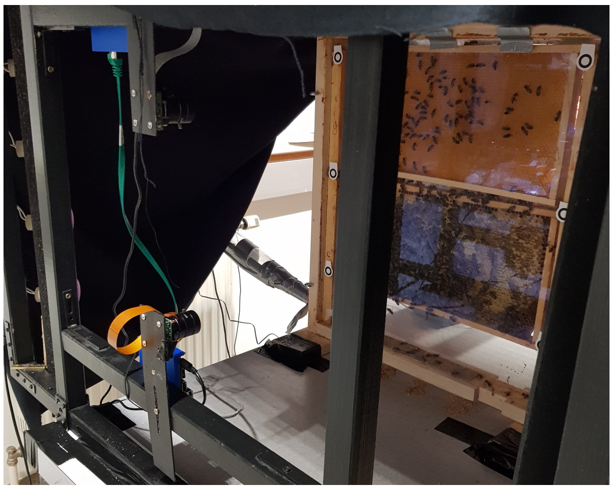

Figure 5.1. Demonstration of mounting the cameras on a wooden frame. The cameras are directly connected to the microcomputers.

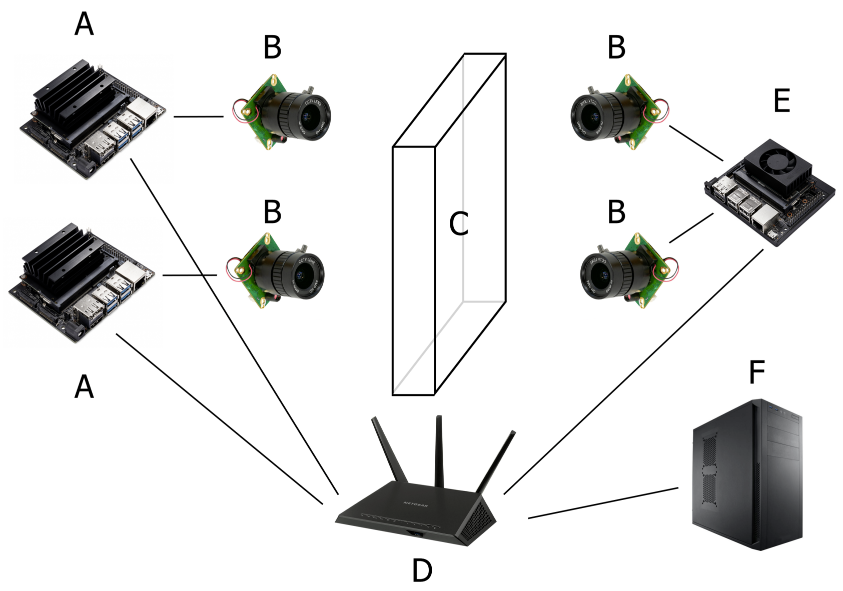

Figure 5.2. Connection diagram of individual components of the experimental setup. **A** Nvidia Jetson Nano microcomputer *(courtesy of nvidia.com)*, **B** Camera Arducam B0274 *(courtesy of arducam.com)*, **C** Observation hive, **D** Central router, **E** Nvidia Jetson Xavier microcomputer *(courtesy of nvidia.com)*, **F** Master computer.

5.3 Robotic Operating System

The Robotic Operating System (ROS) is a software layer on top of the host operating system that provides synchronous and asynchronous communication between the computational nodes of the robot control and monitoring system. ROS allows abstraction over the hardware structure of the robotic device and the interaction of its often distributed components [66].

ROS Noetic was used to implement the software layer of the experimental setup. The system is tasked with providing communication between the various components of the observation apparatus, which are responsible for recording image data from the observation hive and real-time analysis. ROS allows for creating and managing a scalable system, permitting easy diagnosis and visualisation of the resulting data at various levels.

ROS allows full use of the host operating system. In our case, this is Ubuntu Linux 20.04. Therefore, it is possible to use standard operating program tools in combination with a wide range of user packages provided within the ROS distribution or by other developers as part of the ROS ecosystem. Many tools and libraries that are often used in robotics are already included in the original installation of the system.

The philosophical goals of ROS, according to the system authors, are as follows [67]:

- Peer-to-peer
- Tools-based
- Multilingual
- Thin
- Free and Open-Source

Peer-to-peer refers to communication scheduling between compute nodes. In contrast to alternative client-server based systems, ROS can operate in heterogeneous topology networks of distributed systems. Thus, it can optimise traffic on individual physical links.

The individual components of ROS implement the loose coupling principle. They try to distribute the system's functionality as much as possible to increase stability in case of failure of certain components. The tool-based approach also emphasises many separate tools that can be used for other tasks such as diagnostics and visualisation that are not directly related to vital processes.

ROS supports many programming languages natively. This does not limit the user in their choice. It allows for a diverse composition of projects, allowing developers to choose a language according to a specific task or their preferences.

The system supports the independence of individual components from the base system so as to maximise reusability. The compute nodes themselves should then be 'thin', only combining functionality from different modules and libraries and integrating them into the system communication.

ROS is developed under a BSD license, which allows various applications, including commercial usage. This opens up the possibility of community development, which presents many potential improvements and regular updates and a higher probability of discovering various bugs, etc.

The basic building blocks of the system are nodes. These are the processes in which the logic itself is executed. Nodes communicate with each other by sending messages that have a clearly defined structure. ROS offers many types of predefined message types for standard communication purposes; however, it also offers the possibility to define custom composite message types. Messages are sent to a topic, a name (a form of address) where messages can be sent and from where they can be received. Such publishing is done in the form of broadcast (potentially multiple recipients). In the case of one-to-one interaction, it is possible to use a service which implements a request-response communication. Nodes can be further structured into packages and meta-packages, which helps maintain a clear structure of the project [67].

The fundamental component of the system is the master. The master name server provides the interconnection of the nodes. It is launched with the `roscore` command. The catkin tool is used to build packages. CMake language is employed to define the dependencies that catkin uses during the build. The individual dependencies are then listed in the file `CMakeList.txt`. The `rosrun` tool is used to run a single node. If a package is composed of multiple nodes, it usually makes sense to build a graph from these nodes, which will later be launched as a unit. The definition of such a network can be specified in an XML file with the *.launch* extension, along with shared parameters and other attributes. The `roslaunch` tool is used to launch a network of nodes defined in this manner [66].

The `rosbag` tool can be used to record messages received on a selected topic. The recordings are stored in files with the *.bag* extension. Replaying the bag file simulates the system's state at the time of recording. This allows to test system components or visualise the robot in the simulator.

Applications such as `RViz` or `rqt_image_view` can serve as data visualization tools. The `RViz` application acts as a subscriber node that listens on the specified topic. A wide variety of data can be visualised. Apart from 2D image data, 3D data such as point clouds generated by lidars or stereo camera systems (Intel RealSense, Kinect etc.) can be displayed.

5.4 ROS pipeline

Based on the ROS system, an automated pipeline was created. The task is to collect and archive data and analyse it online or share interesting statistics via social media.

For each of the four monitored combs, a separate pre-processing of the data is performed. The nodes `rr_camera`, `rr_whycon`, and `rr_cropper` are active on the individual microcomputers. The camera node serves as a driver for the hardware recording device. The WhyCode node implements a circular ID-less fiducial marker introduced first in [68], extended by an encoding scheme in [69], and improved with a full 6-DOF estimation in [70]. Detailed use of the marker in RoboRoyale is described in [71]. The tracked marker is placed on the queen's thorax. Whycon also handles the detection of tags at the corners of the combs, based on which positions a homography matrix is computed. The cropper node provides image cropping in the vicinity of the queen. Thus, it simulates a robotic camera manipulator moving simultaneously above the queen.

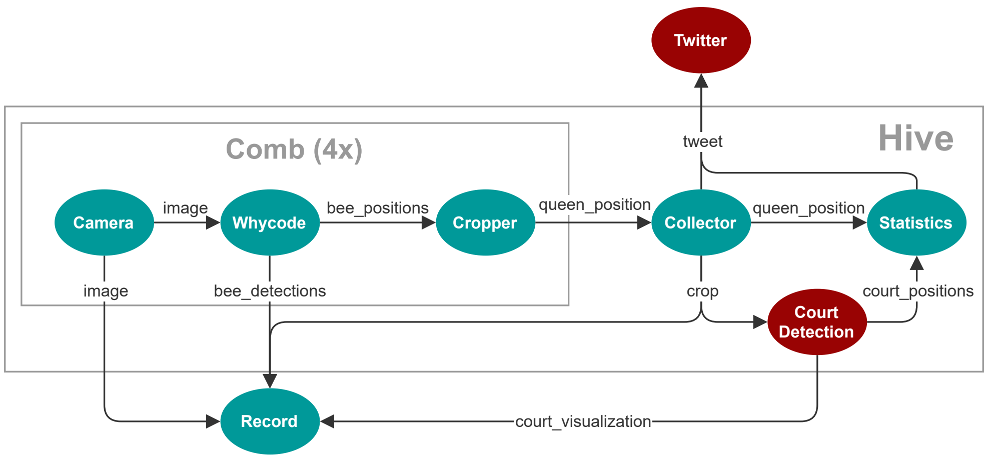

Figure 5.3. Simplified communication diagram between computational nodes of the experimental setup system. Ovals represent nodes, and arrows represent communication channels. The nodes created by the author are highlighted in burgundy.

The nodes in the comb group are also part of the hive group. The collector node handles the collection of data from individual devices. It also decides which of the received detections and crops of its surroundings indeed represent the queen and will be passed on to the court node (see 6.6.1). Data about the queen's movement and her court bees is received by the statistics node, which takes care of its archiving and aggregation. Interesting statistical data and logging information about the state of the system can be periodically sent to the Twitter account via the twitter node (see 6.6.3). Image and other data is stored on remote storage via the record node.

The system is implemented in such a way that it is scalable. Thus, there can be more hive groups, which allows parallel monitoring of more colonies at a time and therefore, more objective research conclusions can be drawn.

5.5 Datasets

Experiments were performed on data obtained from the experimental setup. A dataset consisting of images taken on March 17 and 18, 2022 was used to establish the YOLO detection model. The dataset consists of ten randomly selected images from combs

0 and 3. The cameras used for the two combs have different settings, and therefore the images do not have identical parameters and differ in the amount of noise and brightness. In an attempt to generalise the resulting model, five images from each of the aforementioned honeycombs are represented in this dataset. The images and their annotations are cut into 60 squares of size 1024×1024. The posted dataset is split into training and validation data at a ratio of 9:1. The training dataset contains a total of 3258 individual annotations, and the validation set 323 annotations.

The annotated detections are located in the text files. Each frame has a corresponding detection file. One line of the file contains the description of one detection bounding box in YOLO format:

```
<left> <top> <width> <height>   ,
```

where `left` is the number of pixels between the left edge of the box and the left edge of the image. Same for the `top` value, respectively. The `width` and `height` define the dimensions of the bounding box in pixels. All values are normalised by the width and height of the entire image [43].

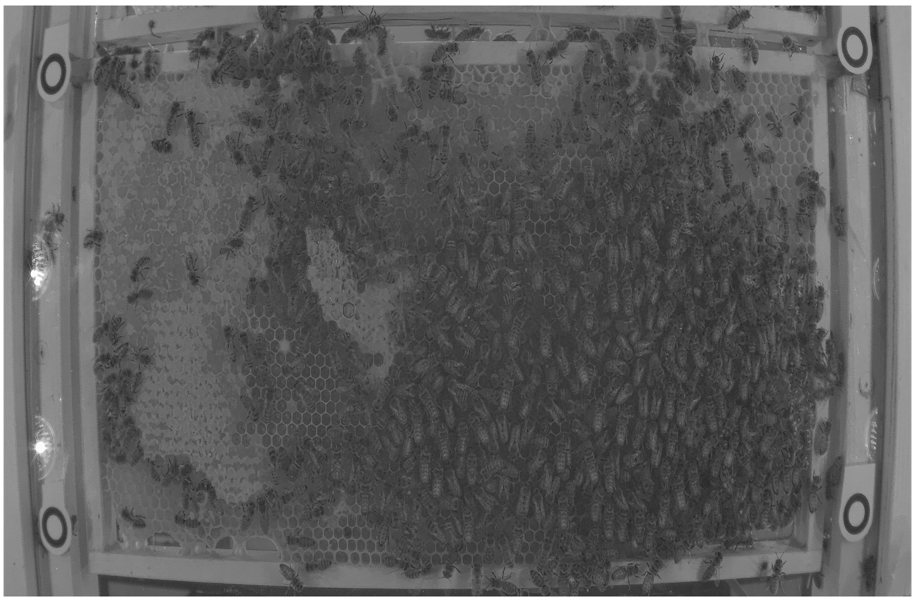

Figure 5.4. A sample of one frame from a dataset containing images of entire honeycombs. In this case, it is comb 0 from May 17, 2022.

The markerless bee detection model requires images in temporal sequence. Thus, it was impossible to use the annotations from the first dataset, created upon randomly selected images. A new dataset was constructed, consisting of forty 1024×1024 images that are aligned such that the queen is located in the centre of the crop. The images are from the 14th hour of recording on March 18, 2022. The images and annotations are reduced to half the size to match the pre-trained model of the U-Net segmentation neural network used. The detections, in this case, are in a proprietary format [58]. Again, each image file has an adequate text file associated with it. Each line contains one detection, representing one bee in the image. The detection line is in the format:

```
<x>,<y>,<class>,<orientation>   ,
```

where `x` and `y` represent the coordinates of the centre of the bee, calculated from the left and top edges of the image, respectively, `class` represents one of the supported classes

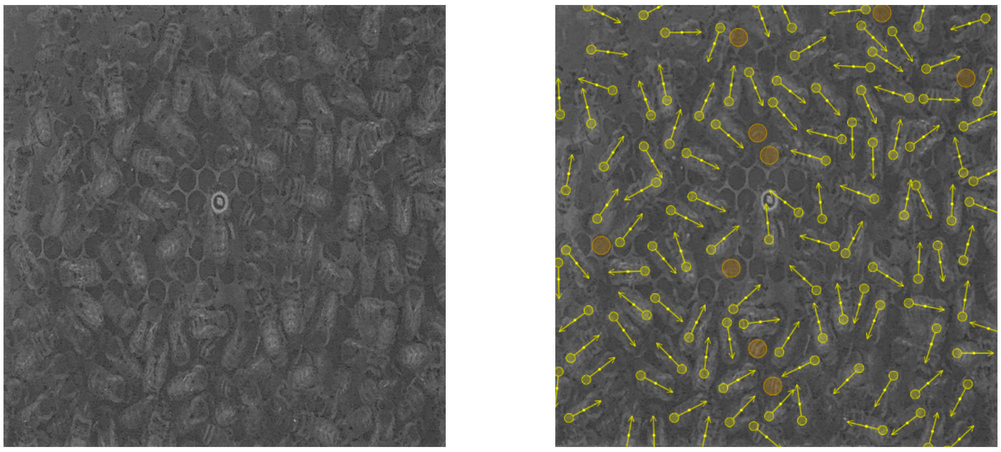

Figure 5.5. Example of cropped data centred on a honeybee queen. **Left**: sample of the frame itself. **Right**: sample frame with ground truth annotations.

(visible bee = 0, in-cell bee = 1), `orientation` is the direction of rotation of the bee's body. A bee with orientation 0 faces upwards, and its orientation value increases clockwise. Orientations are in degrees from the interval $[0; 360)$.

A test dataset was created to test and compare the performance of each detection and tracking method. It consists of nineteen consecutive images aligned to the honeybee queen's body. The images were taken on March 18, 2022, during the 17th hour of recording. Annotated detections in markerless system format also accompany the images. The dataset contains a total of 2320 annotation labels. In addition, the dataset contains a folder of ground truth verified trajectories to evaluate the tracking performance. Each unique identifier assigned to individual bees has a corresponding text file. One line of the file represents one detection belonging to a particular identifier. The lines of the file are in the format:

```
<frame-id>,<x>,<y>,<class>,<orientation>   ,
```

where `frame-id` represents the identification number of the frame in which the detection was made, the other parameters then match the format for the detections.

A separate dataset was created to train and validate a classifier, distinguishing bees within and outside the court. It consists of more or less randomly selected images from March 2022. The images used are again queen-centred and contain court events. Ground truth annotations have been kindly provided by experts from the University of Graz. The annotations apply the same format as the markless system but add an additional court bee class (= 2) to represent bees interacting directly with the queen.

Ground truth data annotations were created using a modified version of the Dense Object Annotation application [72]. The program has been extended with the ability to change the class of the placed annotations, zoom the image detail view, and add a new annotation class for bees that are part of the court.

A Python script was created to convert the annotations between the Markerless tracking and YOLO formats. The dimensions of the detection bounding box are derived as follows:

$$\text{BoxWidth} = 2\sqrt{a^2\cos^2(\alpha) + b^2\sin^2(\alpha)},$$

$$\text{BoxHeight} = 2\sqrt{a^2\sin^2(\alpha) + b^2\cos^2(\alpha)},$$

where a is half the length of the bee (semi-major axis of the ellipse), b is half the width of the bee (semi-minor axis of the ellipse), and α is the angle defined in the markerless annotation format description $+\frac{\pi}{2}$.

All created datasets are enclosed (see Appendix A).

5.6 Libraries used

In the following section, the programming libraries used in this thesis will be described. The OpenCV [73] library provides many tools for the manipulation of images and video. The algorithms provided facilitate the work, especially in computer vision research.

Scikit-learn [74] is a python library that provides efficient implementations of many machine learning algorithms. Individual methods offer a standard API, so the library allows easy experimentation with a wide range of methods.

Imbalanced-learn [75] is an open-source library that serves as an extension to Scikit-learn. The implemented algorithms deal with problems of imbalanced datasets. In the case of this work, under-sampling methods were used.

The MLxtend [76] library was used to render decision boundaries for each classifier. It primarily serves as an extension for Scikit-learn while implementing many additional algorithms suitable for building and evaluating machine learning models.

A number of visualisations were created during the experiments using the Matplotlib [77] library. The tool implements a straightforward API for creating many standard figures and graphs.

The py-motmetrics [78] library implements an API for assessing the quality of multi-object tracking algorithms using a set of defined metrics. The user defines an instance of the `MOTAccumulator` class, on which the user in each frame records the ground truth identifiers, the output identifiers of the tracking method and the relationship between them. These relationships are defined by a distance matrix, which, in the case of MOT, corresponds to the IOU values between every two boxes. This matrix can be calculated using the `distances.iou_matrix` method, also provided by the library.

5.7 Other software

The experiments were implemented in the Python programming language [79]. Python provides a friendly environment for creating machine learning models (see 5.6). At the same time, all assessed methods were also implemented in this language. Finally, the ROS system offers the integration of this language as one of the natively supported languages, for which it also provides solid documentation.

Some experiments were performed in the Google Colab [80] environment, which offers free cloud computing resources for research and development.

Computationally intensive experiments were performed on the high performance computing cluster RCI [81] at the Czech Technical University. The supercomputer offers 61 nodes with specific purposes (CPU, multi CPU and multi GPU installations), designed for computationally intensive research projects.

Weights & Biases [82] is a platform that offers the possibility of online monitoring of machine learning experiments. It allows to visualise in real-time the results of experiments and the state of the hardware, organise and reproduce the obtained results, and assess the performance of the produced models.

Chapter 6
Results

This chapter will present the results of the individual experiments performed. All program bases and created models are enclosed (see Appendix A).

6.1 YOLO detector

The aim of this part of the project was to create a trained YOLO classifier that will be able to detect individual bees on the acquired experimental data. YOLOv5 [43] system was chosen for this experiment. Version 5 is one of the latest iterations of the algorithm, providing a friendly environment for incorporating the algorithm into a custom Python-based system. In addition, this project offers a link to an online platform for visualising and managing machine learning experiments - Weights & Biases [82].

For training and validation, 40 images of 1024×1024, the original size of the source cutouts (see 5.5), were used. Experiments were run remotely on the RCI computing cluster [81] using a single NVIDIA Tesla V100-SXM2-32GB graphics unit. A total of 300 training epochs were performed, with the best mAP values on the validation dataset achieved at epoch 97 with a mAP@0.5 value of 90.82%. Further in the process, the detection quality stagnated (see Figure 6.1). In the case of F1 scores, the same break in the trend of improvement around the 100th epoch can be observed.

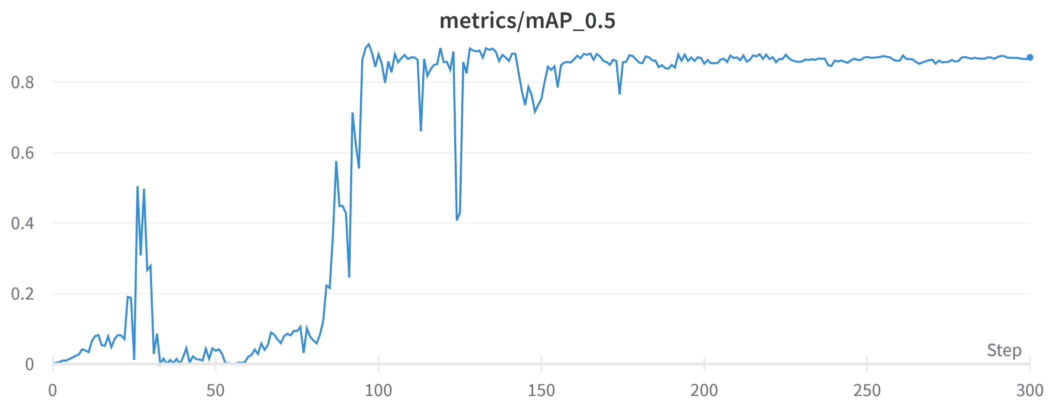

Figure 6.1. Values of mean average precision during the training process. Predictions with an IOU to a ground truth greater than 0.5 are considered valid.

A continuing downward trend can be observed in the training dataset on the loss function progression visualisation. However, both bounding box regression loss and objectness loss have stabilised on the validation data.

The created YOLO model was tested on a test dataset (see 5.5). The model achieved results shown in Table 6.1.

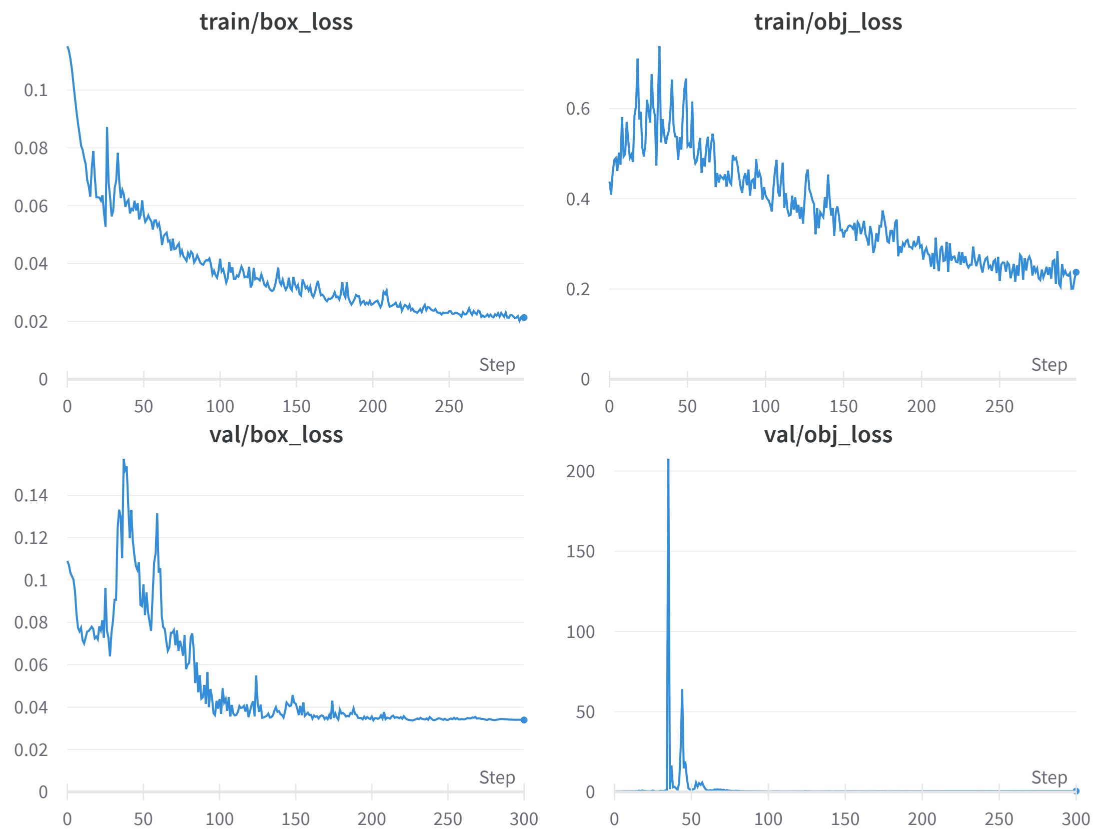

Figure 6.2. Loss function values on the training and validation datasets during the training process. Results on the training dataset in the upper half and results on the validation dataset in the lower half.

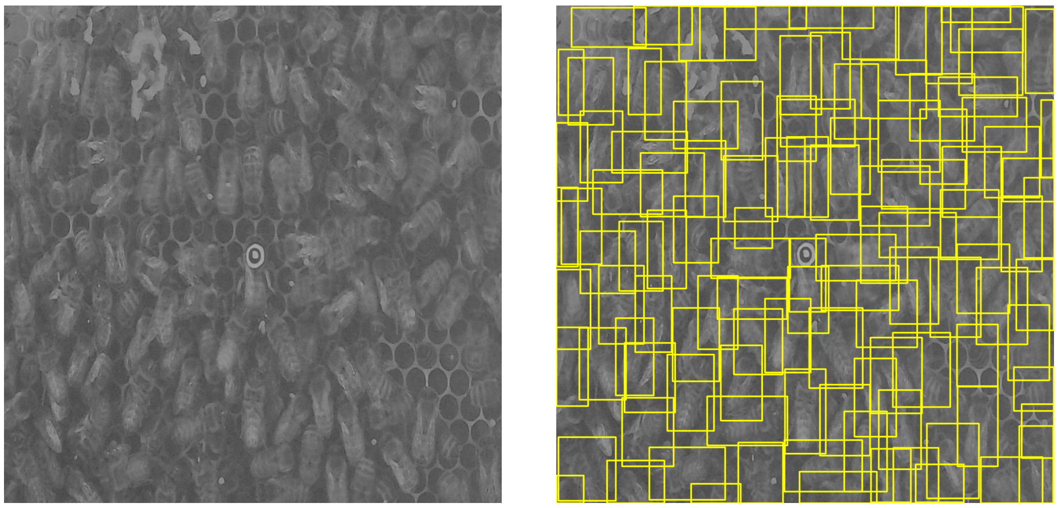

Figure 6.3. Sample of YOLO classifier output detections.

Precision	Recall	F1	mAP@0.5	time	memory
92.5%	73.3%	81.8%	81.6%	92.41 ms	2249 MiB

Table 6.1. Resulting performance statistics of the YOLO model on the test dataset. Time represents the average time for inference of one frame; memory represents the amount of RAM used by the program, both on the master computer of the experimental setup.

6.2 Markerless bee detector

This part of the project aimed to establish a detection model for the Markerless Tracking system [58]. The model was created using a dataset consisting of 40 consecutive images centred on the queen (see 5.5). The training process was performed on the RCI cluster [81] using a single NVIDIA Tesla V100-SXM2-32GB GPU. The model is built upon the 1000th epoch of the pre-trained model, which is extended by an additional 300 epochs.

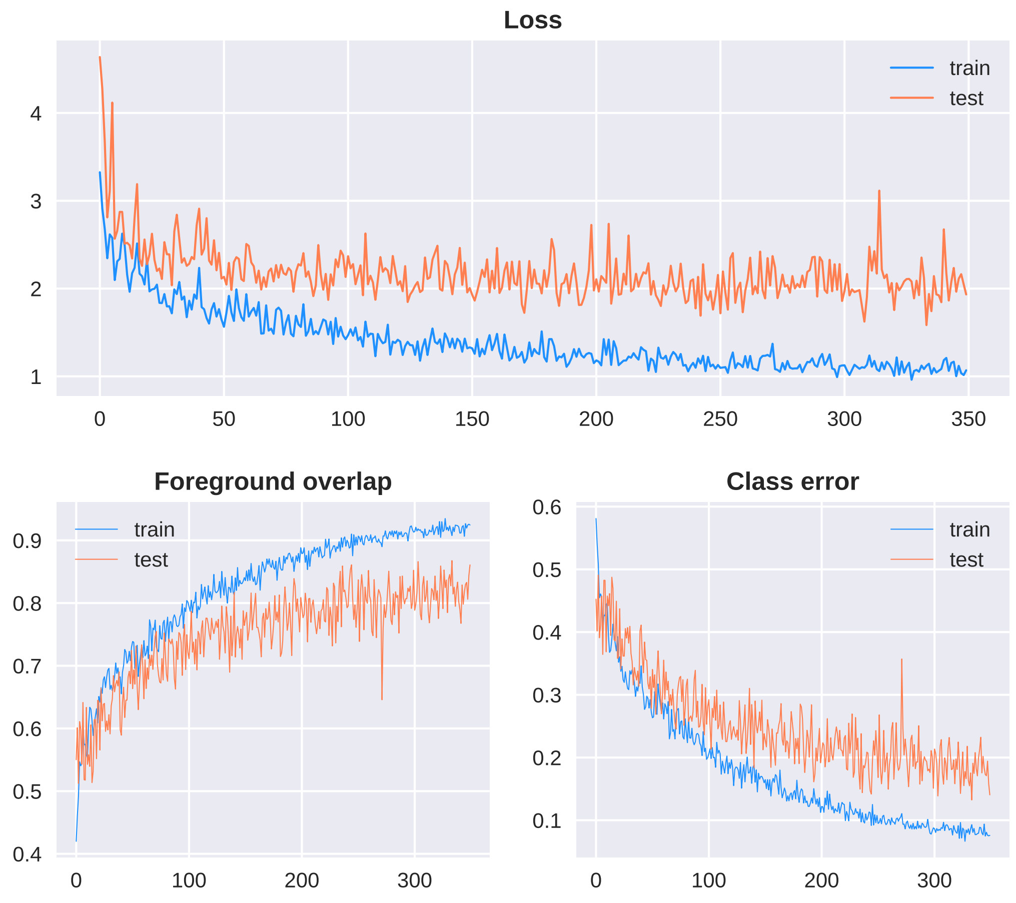

Figure 6.4. Evolution of the values of selected parameters during training of the Markerless detection model. The x-axis represents training iterations.

While the overall loss function on the validation set decreased mainly in the first fifty iterations, the class error showed continuous improvement throughout the training period. Angle error tended to oscillate in the area around the value of 0.35. During experiments with a more significant number of iterations (up to 3000), the monitored parameters tended to diverge, except for the background overlap, indicating over-fitting. Therefore, the enhancement of 300 iterations of the pre-trained model has been chosen for the final model.

The created Markerless model was tested on a test dataset (see 5.5). The model achieved the following results:

Precision	Recall	F1	time	memory
79.3%	88.1%	83.3%	358.97 ms	964 MiB

Table 6.2. Resulting performance statistics of the Markerless detection model on the test dataset (mean values). Time represents the average time for inference of one frame; memory represents the amount of RAM used by the program, both on the master computer of the experimental setup.

6.3 ByteTrack tracker

The original implementation [53] was employed to implement the usability evaluation of the ByteTrack tracking system. The BYTE tracking algorithm was combined with the earlier developed YOLOv5 model to generate input detections.

The tracking system was subjected to performance testing on a test dataset with ground truth trajectories. The values of each tested metric were computed using the py-motmetrics library [78]. The IOU metric with a threshold value of 0.5 was adopted to compute the distance matrix. The resulting values of the monitored metrics are as follows:

ID-Precision	ID-Recall	IDF1	MOTA
69.5%	48.2%	57.0%	29.6%

Table 6.3. Resulting performance statistics of the ByteTrack tracking system on the test dataset.

6.4 Markerless bee tracker

In order to evaluate the quality of the tracking, the py-motmetrics library [78] and the distance matrix between ground truth detections and generated predictions based on the IOU metric with a threshold value of 0.5 were utilised again. By performing the experiment using the test dataset, the following values of the observed metrics were achieved:

ID-Precision	ID-Recall	IDF1	MOTA
77.0%	83.3%	80.2%	64.0%

Table 6.4. Resulting performance statistics of the Markerless tracker system on the test dataset.

6.5 Court bee detection

For further use in the project, it is desirable to create a system to decide which bees from the queen's surroundings are part of her court. Previous research suggests that court bees can also be distinguished from positional information [6]. These bees perform direct interactions with the queen. Thus, they are located in her vicinity. This part of the thesis presents a machine learning approach to distinguish court bees based on their position and pose.

The detections in the Markerless tracking system are used as input to the classifier. The expected size of the source image data is 80 pixels per bee length. Another requirement is that the queen is positioned in the centre of the image, which is achieved by the Whycon tracking system (see 5.4) and subsequent cropping.

In the first stage, the input observation vectors are calculated. The vector consists of two components for each bee: the Euclidean distance of the bee centre from the queen and the rotation angle from the detected queen body centre. The angle is from the interval $[0; 180]$, i.e., the deviation to either side is positive. Classification (4.1) is performed on this data in the next stage, where the list of observation vectors for each bee is equivalent to the set $\mathcal{X}$ and the set $\mathcal{K} = \{\text{court bee}, \text{worker bee}\}$.

In order to select the optimal classification method, a series of experiments were performed. First, feature vectors were extracted from all frames of the court detection dataset (see 5.5). These detections add the class "court bee" to the original format. The resulting data were highly unbalanced in favour of the off-court bee class. Therefore, the *NearMiss* under-sampling method from the imbalanced-learn library [75] was applied to the data. NearMiss falls into the prototype selection category of the under-sampling algorithms, so it does not generate any new data to represent the original samples. The method only reduces the number of samples from the over-sampled class to balance that number with the number of samples in the other classes. This method leaves only those points in the future training set that have the lowest average distance to the N closest samples from the other classes [83]. In the case of the court classifier, $N = 3$. In this way, it was possible to balance the data on the same number of samples in each class while the selected data is as close as possible to the future decision boundary.

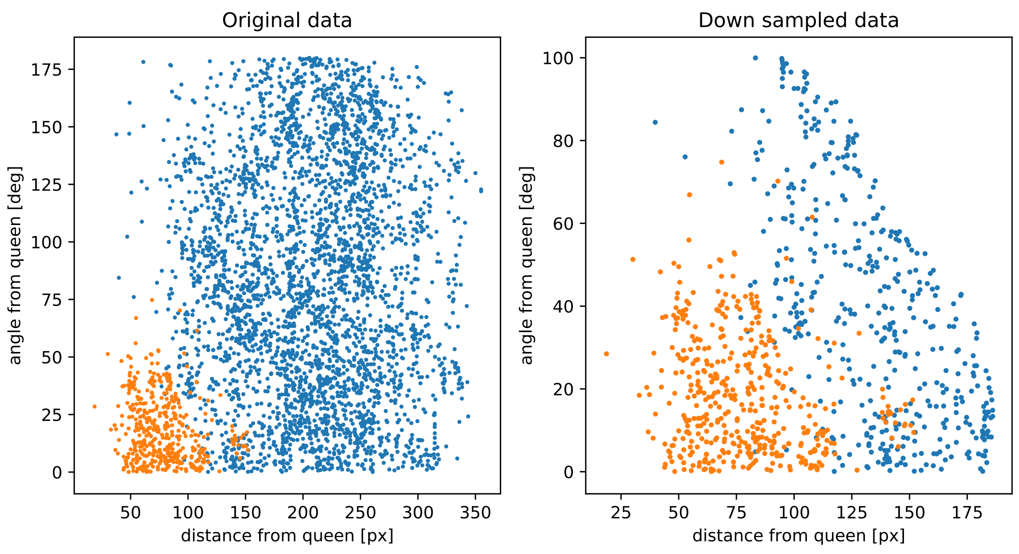

Figure 6.5. Legend: blue dots - bees outside the court; orange dots - bees included in the court; Left - original dataset with unbalanced class representation; Right- Dataset after reducing the number of bees in the over-sampled class using *NearMiss* method.

Three candidate methods were chosen for the classifier: k-nearest neighbours, SVM, and a fully connected neural network. A 5-fold cross-validation method was used to optimise the hyperparameters. The methods were tested in this manner with the selected hyperparameter values. The one that showed the best mean F1 score over all folds was selected for the final testing on the validation dataset.

For the k-nearest neighbours method, the optimised parameter was the number of neighbours. It was selected from values between one and 50. The best score was obtained for $N = 38$ with a mean score of 91.66%. For the SVM method, the regularisation parameter C was decided between values of $10^n, n \in -2, \ldots, 3$. The best score of 91.77% was achieved for $C = 100$. The number of neurons in each layer was tested for the neural network. The best F1 score of 94.10% was achieved for ten neurons in the first layer and 190 neurons in the second layer. Rectified Linear Unit (ReLU) was employed as the activation function, and stochastic gradient descent was applied as the optimisation method.

During the evaluation, the dataset was split into training and test subsets in a 4:1 ratio. Each method achieved the following scores on the test subset of the dataset:

Classifier	Precision	Recall	F1-score	Accuracy
KNN	96%	96%	96%	96%
SVM	96%	96%	96%	96%
NN	94%	94%	94%	94%

Table 6.5. Resulting performance statistics of the proposed Court detection classifiers.

6.6 ROS pipeline

The results of the experiments had to be integrated into an automated pipeline based on the ROS system. In this section of the paper, the computational nodes contributed by the author to this pipeline will be presented.

6.6.1 Court bee detection node

In order to extend the pipeline of automated data collection, a node implementing the Markerless detection system was created. The node's input is a 1024×1024 px image crop. The cutout contains the queen bee in its centre and covers a large part of its surroundings. The input image is expected on the topic `/cropped_image`. The `frame_id` (image metadata) has the following format:

```
/hive_<hive-num>/comb_<comb-num>/camera/crop_<x>_<y>   ,
```

where `hive-num` is the sequence number of the hive, `comb-num` is the sequence number of the honeycomb from which the image originated, x and y are the coordinates of the upper left corner of the cutout relative to the original image.

The original Markerless detection algorithm has been modified so that it can be applied to online data. The image is first scaled to 512×512 px to match the input size of the trained model. The model performs segmentation and subsequent detection of bee positions and orientations. Afterwards, the created court detection classifier is applied, which labels bees in and outside the court (see 6.5). The positions of the detected bees are plotted. The individual classes are distinguished by colour. The visualisation image is posted to the topic `/court_visualization`.

Detections of court bees are further processed and published on the topic `/court_poses` in message format rr_msgs/BeePositionArray. The message contains one BeePosition element for each bee in the court. The values `u` and `v` represent the position of the bee within the original image. `alpha` contains the orientation angle of its body in radians, starting on the positive x-axis and increasing counterclockwise.

The values `x` and `y` represent the position of the bee on the plane in metric coordinates, and `phi` represents the orientation of the bee after undistorting the image. A function to convert image coordinates to hive coordinates is used to calculate this positional information. First, the canonical coordinates are calculated using the `undistortPoints` function from the OpenCV library and the camera properties contained in the topic `/camera_info`. The resulting coordinates in the hive coordinate system are calculated as follows:

Using the homography matrix **H**, the points are converted into the coordinate system of the hive:

$$\begin{bmatrix} x_{hive} \\ y_{hive} \\ z_{hive} \end{bmatrix} = \mathbf{H} \cdot \begin{bmatrix} x_{can} \\ y_{can} \\ 1 \end{bmatrix}.$$

The homography matrix is obtained from the topic `/homography_matrix` and is being constantly recalculated from positions of the calibration marks in the corners of the comb. The resulting metric coordinates in the comb plane are obtained as follows:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{x_{hive}}{z_{hive}} \\ \frac{y_{hive}}{z_{hive}} \end{bmatrix}.$$

The angle `phi` is calculated by converting two points into the hive coordinate system, one at the centre of the bee and the other at the edge of its head. The angle of the line between these points is equal to `phi`.

6.6.2 YOLO detection node

A node that implements the developed YOLOv5 detection model was devised for testing and demonstration purposes. For this purpose, the library yolov5-pip [84] was utilised, which wraps the original implementation and exposes the code on a public package repository. The node listens similarly to the court detection node on the topic `/cropped_image`. The annotated image is published on the topic `/output_image`.

6.6.3 Twitter node

In order to be able to automatically share the results of the experiments with the public or log the system status on an easily accessible platform, rr_twitter_node was created. This node listens on the topic `/tweet` and `/tweet_testing`, respectively. The message is submitted in rr_msgs/Tweet format and contains either text-only or text accompanied by an image. Messages are tweeted on one of the project accounts, depending on the topic used. The tweepy library [85] was used for the implementation.

Chapter 7
Discussion

7.1 Selection of the methods

The methods introduced in Chapter 4 represent state-of-the-art approaches for detection and tracking. These systems also represent candidates for the selection of methods that were tested and extended for the final performance assessment at a later stage.

The YOLOv5 [43] algorithm was selected for the general detection experiment. According to the Papers With Code [86] portal, which compares current machine learning methods of various disciplines, at the time of writing, the YOLO family algorithms rank at the top of the performance rankings across the test datasets in the Real-Time Object Detection category. The YOLOv5 system was chosen for the experiments because it is one of the latest iterations of the YOLO algorithms and also offers a friendly framework for implementation within the Python environment.

Among the systems specifically focused on bee detection, the Markerless tracking system [58] was selected. According to the authors, the system offers comparable performance to bee tagging based techniques. Tagging methods were abandoned due to the need for continuous marker addition, associated colony disruption, and other disadvantages of using fiducial markers see 4.4.1.

The ByteTrack [53] method was chosen, as one of the newest methods for multi-object tracking. This system again ranks high in the [86] benchmarks, where ByteTrack achieved 80.3% MOTA on the MOT17 [87] dataset.

The methods used to assess the detection of courts were selected from a wide range of possible classifiers offered by the Scikit-learn [74] library. K-nearest neighbors [24] was chosen to represent non-parametric supervised learning methods, SVM [25] to represent linear classifiers. Fully connected neural networks were selected for comparison with both methods.

A number of arguments support the use of ROS. The experimental setup consists of many heterogeneous hardware components. The cooperation of these parts and the distribution of tasks between the different physical devices would be utterly complex without a central communication system. It is a genuine challenge to ensure the correct functionality of the individual programs separately and especially their cooperation. ROS allows circumventing these problems with a standardised solution.

The experimental setup is located in the building of the University of Graz in Austria. However, researchers from CTU need to have full remote control over the system. ROS offers the possibility to connect to a remote kernel, so local tools that operate with the remote system can be used. For example, one can monitor the real-time image from the cameras using the `RViz` tool or use other visualisation applications.

Several partners are working on the project. The topic system allows simultaneous work from multiple locations. At the same time, it offers a user-friendly way to check the functionality and status of the system outputs.

7.2 Comparison of systems

The experimental results of each method will be evaluated and placed in the context of other methods. Also, the choice of methods for the implementation of the ROS pipeline will be justified.

7.2.1 Detection

The systems in the comparison showed highly comparable results in terms of the primary metric of interest, the F1 score. At the same time, the systems show a significant difference in Precision and Recall. Thus, the values obtained indicate that YOLO excels in determining positive detections, unlike the Markerless method. In other words, the detections made by the YOLO model are 92.5% identical to the ground truth. However, it lags considerably in the number of correct detections relative to the number of bees in the image.

Despite the similar result, the Markerless detection method performed 1.5 percentage points better on the test dataset and was chosen for further use and implementation in the ROS pipeline(see 5.4). Another argument for choosing this method is the simultaneous detection of bee orientations, which is helpful for further analysis. Nevertheless, the developed YOLO model represents a promising alternative for the chosen method. The YOLO model shows a significantly higher speed in inference. On an experimental setup, the method shows a speed of around 100 ms per frame, theoretically allowing online inference at up to 10 FPS (frames per second). In contrast, the Markerless method requires more than three times the processing time. The method also seems to be more reliable in the case of overlapping bees, which is also due to the functionality of the Markerless method. Obtained segmentations are clustered in the case of a bee partially hidden behind another bee. The different parts of its body do not belong to the same cluster and are therefore not considered potential detections.

Method	Precision	Recall	F1	time	memory
YOLOv5	92.5%	73.3%	81.8%	**92.41 ms**	2249 MiB
Markerless	79.3%	88.1%	**83.3%**	358.97 ms	964 MiB

Table 7.1. Resulting performance statistics of both detection methods tested. Time represents the average time for inference of one frame and memory, the amount of RAM used by the program, both on the master computer of the experimental setup.

Compared to the results in [58], a decrease in detection performance can be observed. Recall during the experimental testing reached hear 99% and could thus be compared to the performance of human annotators (also 99%). However, the authors do not clearly describe how the detections were categorised (TP, TN, FP, FN). These performance checks were performed by manual evaluation, which does not allow an exact comparison with the method used here based on an IOU threshold of 0.5. Another possible reason may be the high noise and dimness of the data obtained from the experimental setup, which often obscures essential contours, making the task more challenging even for a human annotator.

7.3 Tracking

In the case of tracking methods, a fundamental difference between the performances of the two methods can be observed. In the case of the IDF1 metric, which focuses primarily on the accuracy of the trajectories produced, there is more than a twenty per cent difference. The MOTA metric, more weighted on detections' quality, shows even more than a thirty per cent difference. Thus, the values obtained suggest that even a state-of-the-art method for multi-object detection finds a challenge in a densely populated environment such as a honeycomb. The missing orientation information probably also plays a role in the erroneous assignment of identifiers and the concatenation of the tracklets produced. The Markerless method was selected for further use.

Method	ID-Precision	ID-Recall	IDF1	MOTA
ByteTrack	69.5%	48.2%	57.0%	29.6%
Markerless	77.0%	83.3%	**80.2%**	64.0%

Table 7.2. Resulting performance statistics of both tracking methods tested.

7.4 Court bee detection

SVM with regularisation parameter C = 100 and Radial Basis kernel Function (RBF) [25] was chosen as the method used to distinguish bees in and out of court. This method shows both F1-score and accuracy of 96%. The result is the same as for the k-nearest neighbours method; however, SVM was chosen due to less dependence on the training data in a smoother decision boundary and, therefore, potentially better generalisation to previously unseen data. For the metrics comparison see Table 6.5.

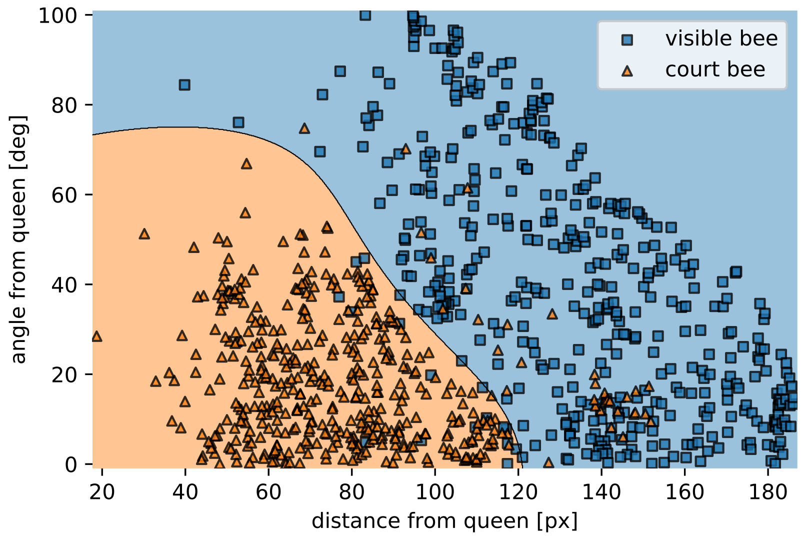

Figure 7.1. Decision boundary for court bees detection system based on SVM method.

7.5 Possible future improvements

The development of inference models and testing of selected methods revealed areas for potential improvement. These suggestions can serve as a starting point for further optional modifications of the mentioned methods for application within the project.

The recording setup offers room for improvement. As mentioned, some experimental data contain a significant amount of noise and are dark. Also, cameras use wide-angle lenses. Hence, the image is fairly distorted, affecting the detection quality at the honeycomb edges. The focal length used also creates a sharpening problem at the periphery. The difference between distances from the sensor to the centre of the honeycomb and between the sensor and comb boundary is significant. This problem has to be compensated by a higher aperture value and, therefore, greater depth of field but also worse light transmission. Since the experimental setup is limited by space, it is not the preferred solution to use lenses with long focal lengths and thus allow the aperture number to be reduced. At the same time, the lower aperture would probably result in worse contrast in details. The solution proposed is to increase the amount of light by adding additional infrared LED panels. Such an approach would allow reducing the sensor's sensitivity and, ultimately, noise caused by its high value. Adding more lights would also improve the uniformity of the illumination.

Another suggestion for a possible improvement of the quality of the detections is to increase the contrast between the bees and the background by using more fresh and bright honeycombs. Of course, frequent wax changes would be a highly disruptive element, but it would be good to at least experimentally test the impact of such a measure on system performance.

In the case of Markerless bee tracking, performance can potentially be improved by changing the input resolution. Currently, the image is reduced to half the size before an inference is performed to correspond to the weights of the pre-trained model, which operated with a bee length of 80 px. However, if the output resolution were changed to use the complete information of the input data, the model would have to be learned from scratch, requiring the expected hundreds of thousands of bee annotations [58] in the original image size. It should be noted that downsampling of the data was also applied in [58] to compensate for differences in the resolution of the cameras used. The results generated from the downsampled images did not show a significant decrease in detection performance.

As far as the detection of court bees is concerned, improvements in the computation of feature vectors are possible. Currently, the calculation is based on the distance and degree deviation from the centre of the queen's thorax. However, the data suggest that bees that come into contact with the queen's abdomen are also part of the court. Thus, the current method slightly discriminates bees that interact with this lower part of the queen's body against bees closer to the tracking mark. A possible solution would be to derive the queen's orientation and calculate the distance and deflection from either the body axis or the oval representing the body silhouette.

Chapter 8
Conclusion

The project is aimed to select and test methods capable of detecting and monitoring bees in the vicinity of the honeybee queen. In the next phase, a method for distinguishing bees belonging to the queen's court had to be developed, tested and the resulting classifier implemented within the experimental pipeline.

The theoretical basis for both the biological hypotheses in bee behaviour and the principle functionality of the selected state-of-the-art methods was presented. Based on the knowledge of the methods and the objectives of the thesis and the project, key performance indicators were established to select the method that best fits the requirements. A bee type classification system was designed based on geometric data and machine learning technology. The classification system was embedded in the selected detection method and integrated within a pipeline that caters for data collection and analysis in the context of the experimental setup.

The system developed in the thesis may find further applications. Based on the positions of the bees around the queen, statistical information can be obtained that will be used to understand better what happens in this particular region of the hive. For example, various hypotheses can be tested based on the data obtained. Furthermore, long-term observation data can be used to model the behaviour of bees in the vicinity of the queen. The model could also be continuously improved based on real-time positions obtained. In particular, the model would be used as a template of behavioural patterns for robotic agents, thus being able to be more readily accepted by the surrounding bees and thus blend in with the colony. Finally, information about bees' movement around the queen can be utilized by the controllers of the robotic agents to adjust their movements and actions, depending on the actions of the other bees.

The work also offers potential for improvements and extensions. It is possible to refine further the quality and generalization capability of the models created based on newly acquired data and ground truth annotations created by modifying the detections produced by the system. It would be advisable to attempt to accelerate and streamline the method for real-time use. In addition, the input resolution of the detection method could be increased, or the geometric model of the court detector could be enhanced, as described in the discussion.

References

[1] Thomas D Seeley. *The Wisdom of the Hive.* Harvard University Press, 1996. ISBN 9780674043404.

[2] Farshad Arvin et. al. *H2020 RoboRoyale project website.* `https://roboroyale.eu/`. Accessed April 17, 2022.

[3] Jill Atkins, and Barry Atkins. *The Business of Bees. An Integrated Approach to Bee Decline and Corporate Responsibility.* Taylor & Francis Group, 2016. ISBN 9781783534357. 1-108.

[4] *Rapid Assessment of Pollinators' Status. A contribution to the International initiative for the Conservation and sustainable Use of pollinators.* Food and Agriculture Organization of the United Nations. 2008.

[5] Thomas Schmickl et. al. *The Queen and her Robotic Court: A Minimally-Invasive Form of Ecosystem Hacking.* In: *Proceedings of International Science Fiction Prototyping Conference (SciFi-It' 2021).* Ghent, Belgium: 2021.

[6] Martin Stefanec et. al. A Minimally Invasive Approach Towards "Ecosystem Hacking" With Honeybees. *Frontiers in Robotics and AI.* 2022, 9 DOI 10.3389/frobt.2022.791921.

[7] David C Queller, and Joan E Strassmann. Eusociality. *Current Biology.* 203, 861-863. DOI 10.1016/j.cub.2003.10.043.

[8] Human Ageing Genomic Resources. *Honey bee (Apis mellifera) longevity, ageing, and life history.* `https://genomics.senescence.info/species/entry.php?species=Apis_mellifera`. Accessed April 17, 2022.

[9] P.K. Visscher, and R. Dukas. Survivorship of foraging honey bees. *Insectes Sociaux.* 1997, 44 1-5. DOI 10.1007/s000400050017.

[10] Laura Saade Haddad, Louie Kelbert, and A. J. Hulbert. Extended longevity of queen honey bees compared to workers is associated with peroxidation-resistant membranes. *Experimental Gerontology.* 2007, 42 601-609. DOI 10.1016/j.exger.2007.02.008.

[11] Kathryn E. Gardner, Thomas D. Seeley, and Nicholas W. Calderone. Do honeybees have two discrete dances to advertise food sources?. *Animal Behaviour.* 2008, 75 1291-1300. DOI 10.1016/j.anbehav.2007.09.032.

[12] Karl von Frisch. *The Dance Language and Orientation of Bees.* Harvard University Press, 2013. ISBN 9780674418776. `https://doi.org/10.4159/harvard.9780674418776`.

[13] John B Free. *Pheromones of Social Bees.* Peacock Press, 2000. ISBN 1904846106.

[14] Carla Mucignat-Caretta. *Neurobiology of chemical communication.* CRC Press, 2014. ISBN 9781466553415.

[15] K. Crailsheim, B. Gurmann, T. Schmickl, and B. Blaschon. Collective and individual nursing investment in the queen and in young and old honeybee larvae during foraging and non-foraging periods. *Insectes Sociaux.* 2003, 50 174-184. DOI 10.1007/s00040-003-0644-x.

[16] Jürgen Tautz. *The Buzz about Bees: Biology of a Superorganism.* 1. Aufl. ed.. Springer-Verlag, 2009. ISBN 9783540787273.

[17] Dervis Karaboga, and Bahriye Akay. A survey: Algorithms simulating bee swarm intelligence. *Artificial Intelligence Review.* 2009, 31 61-85. DOI 10.1007/s10462-009-9127-4.

[18] Thomas Schmickl, and Heiko Hamann. BEECLUST: A swarm algorithm derived from honeybees. *Bio-inspired Computing and Communication Networks. CRC Press (March 2011).* 2011,

[19] Jong-Hyun Lee, Hyun-Tae Kim, and Chang Wook Ahn. *Foraging Swarm Robots System Adopting Honey Bee Swarm for Improving Energy Efficiency.* In: *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication.* Association for Computing Machinery, 2012. ISBN 9781450311724.

[20] Farshad Arvin, Tomas Krajnik, Ali Emre Turgut, and Shigang Yue. *COSΦ: Artificial pheromone system for robotic swarms research.* In: *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS).* IEEE, 2015. 407-412. ISBN 9781479999941.

[21] Muhammad Salman, David Garzón Ramos, Ken Hasselmann, and Mauro Birattari. Phormica: Photochromic Pheromone Release and Detection System for Stigmergic Coordination in Robot Swarms. *Frontiers in Robotics and AI.* 2020, 7 DOI 10.3389/frobt.2020.591402.

[22] Mark S Nixon, and Alberto S Aguado. *Feature extraction & image processing for computer vision.* Third ed.. Elsevier/Academic Press, 2012. ISBN 0123965497;9780123965493;.

[23] Tao Qin. *Dual Learning.* Springer Singapore, 2020. ISBN 978-981-15-8883-9; 978-981-15-8884-6. Chapter: Machine Learning Basics. 11-23.

[24] N. S. Altman. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician.* 1992, 46 175-185. DOI 10.1080/00031305.1992.10475879.

[25] Corinna Cortes, and Vladimir Vapnik. Support-vector networks. *Machine Learning.* 1995, 20 273-297. DOI 10.1007/BF00994018.

[26] Boguslaw Cyganek. *Object Detection and Recognition in Digital Images.* John Wiley & Sons, Incorporated, 2013. ISBN 9781118618370.

[27] David G. Lowe. *Object recognition from local scale-invariant features.* In: *Proceedings of the IEEE International Conference on Computer Vision.* IEEE, 1999. 1150-1157.

[28] Navneet Dalal, and Bill Triggs. *Histograms of oriented gradients for human detection.* In: *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005.* 2005. 886-893. ISBN 0769523722.

[29] Paul Viola, and Michael J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision.* 2004, 57 137-154. DOI 10.1023/B:VISI.0000013087.49260.fb.

[30] Licheng Jiao, Ruohan Zhang, Fang Liu, Shuyuan Yang, Biao Hou, Lingling Li, and Xu Tang. New Generation Deep Learning for Video Object Detection: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*. 2021, DOI 10.1109/TNNLS.2021.3053249.

[31] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. *A Survey on Performance Metrics for Object-Detection Algorithms*. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, 2020. 237-242. ISBN 978-1-7281-7539-3.

[32] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. *Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation*. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014. 580-587.

[33] Yang Liu, Peng Sun, Nickolas Wergeles, and Yi Shang. A survey and performance evaluation of deep learning methods for small object detection. *Expert Systems with Applications*. 2021, 172 DOI 10.1016/j.eswa.2021.114602.

[34] Ross Girshick. *Fast R-CNN*. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015. 1440-1448.

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2017, 39 (6), DOI 10.1109/TPAMI.2016.2577031.

[36] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. *SSD: Single Shot MultiBox Detector*. In: *Computer Vision – ECCV 2016*. Cham: Springer International Publishing, 2016. 21–37. ISBN 978-3-319-46448-0.

[37] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C. Berg. DSSD : Deconvolutional Single Shot Detector. 2017, DOI 10.48550/ARXIV.1701.06659.

[38] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[39] Lihi Gur Arie. 2022. `https://towardsdatascience.com/the-practical-guide-for-object-detection-with-yolov5-algorithm-74c04aac4843`. Accessed May 7, 2022.

[40] Joseph Redmon, and Ali Farhadi. *YOLO9000: Better, Faster, Stronger*. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017. 6517-6525.

[41] Joseph Redmon, and Ali Farhadi. YOLOv3: An Incremental Improvement. 2018, DOI 10.48550/ARXIV.1804.02767.

[42] Gaudenz Boesch. *Object Detection in 2022: The Definitive Guide - viso.ai*. 2022. `https://viso.ai/deep-learning/object-detection/`. Accessed April 19, 2022.

[43] Glenn Jocher et. al. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference. 2022, DOI 10.5281/ZENODO.6222936.

[44] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You Only Learn One Representation: Unified Network for Multiple Tasks. 2021, DOI 10.48550/ARXIV.2105.04206.

[45] Subhash Challa, Robin J Evans, Mark R Morelande, and Darko Musicki. *Fundamentals of object tracking.* Cambridge University Press, 2011. ISBN 9781139008235.

[46] Yingkun Xu, Xiaolong Zhou, Shengyong Chen, and Fenfen Li. Deep learning for multiple object tracking: A survey. *IET Computer Vision.* 2019, 13 411-419. DOI 10.1049/iet-cvi.2018.5598.

[47] Jack Valmadre, Alex Bewley, Jonathan Huang, Chen Sun, Cristian Sminchisescu, and Cordelia Schmid. Local Metrics for Multi-Object Tracking. 2021, DOI 10.48550/ARXIV.2104.02631.

[48] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly.* 1955, 2 83-97. DOI 10.1002/nav.3800020109.

[49] Jonathon Luiten, Aljoša Ošep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. HOTA: A Higher Order Metric for Evaluating Multi-object Tracking. *International Journal of Computer Vision.* 2021, 129 548-578. DOI 10.1007/s11263-020-01375-2.

[50] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. *Simple online and realtime tracking.* In: *2016 IEEE International Conference on Image Processing (ICIP).* 2016. 3464-3468.

[51] Aditya Singh. *Top 5 Object Tracking Methods.* 2021. `https://medium.com/augmented-startups/top-5-object-tracking-methods-92f1643f8435`. Accessed April 23, 2022.

[52] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. *Simple online and realtime tracking with a deep association metric.* In: *2017 IEEE International Conference on Image Processing (ICIP).* 2017. 3645-3649.

[53] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. 2021, DOI 10.48550/ARXIV.2110.06864.

[54] Franziska Boenisch, Benjamin Rosemann, Benjamin Wild, David Dormagen, Fernando Wario, and Tim Landgraf. Tracking all members of a honey bee colony over their lifetime using learned models of correspondence. *Frontiers Robotics AI.* 2018, 5 DOI 10.3389/FROBT.2018.00035.

[55] Mei Han, Amit Sethi, Wei Hua, and Yihong Gong. *A detection-based multiple object tracking method.* In: *Proceedings - International Conference on Image Processing, ICIP.* 2004. 3065-3068.

[56] Tim Gernat, Vikyath D. Rao, Martin Middendorf, Harry Dankowicz, Nigel Goldenfeld, and Gene E. Robinson. Automated monitoring of behavior reveals bursty interaction patterns and rapid spreading dynamics in honeybee social networks. *Proceedings of the National Academy of Sciences of the United States of America.* 2018, 115 1433-1438. DOI 10.1073/pnas.1713568115.

[57] Tin Kam Ho. *Random decision forests.* In: *Proceedings of 3rd International Conference on Document Analysis and Recognition.* IEEE Comput. Soc. Press, 1995. 278-282. ISBN 0-8186-7128-9.

[58] Katarzyna Bozek, Laetitia Hebert, Yoann Portugal, and Greg J. Stephens. Markerless tracking of an entire honey bee colony. *Nature Communications.* 2021, 12 DOI 10.1038/s41467-021-21769-1.

[59] 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, 122 10-18. DOI 10.1016/j.compag.2016.01.011.

[60] Francisco Romero-Ferrero, Mattia G. Bergomi, Robert C. Hinz, Francisco J.H. Heras, and Gonzalo G. de Polavieja. idtracker.ai: tracking all individuals in small or large collectives of unmarked animals. *Nature Methods*. 2019, 16 179-182. DOI 10.1038/s41592-018-0295-5.

[61] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing, 2015. 234–241. ISBN 978-3-319-24574-4.

[62] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. *Rethinking the Inception Architecture for Computer Vision*. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. 2818-2826.

[63] Mohammed Badawy, A.A. Abd El-Aziz, Amira M. Idress, Hesham Hefny, and Shrouk Hossam. A survey on exploring key performance indicators. *Future Computing and Informatics Journal*. 2016, 1 47-52. DOI 10.1016/j.fcij.2016.04.001.

[64] Treesfive. *Treesfive/calculate-iou*. 2019.
`https://github.com/Treesfive/calculate-iou`. Accessed May 5, 2022.

[65] *3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 1.0.2 documentation*.
`https://scikit-learn.org/stable/modules/model_evaluation.html`. Accessed May 5, 2022.

[66] Yoonseok Pyo, Hancheol Cho, I Ryuwoon, Jung I, and Taehoon Lim. *A Handbook Written by TurtleBot3 Developers Robot Programming From the basic concept to practical programming and robot application*. ROBOTIS Co.,Ltd., 2017. ISBN 9791196230715.

[67] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. *ROS: an open-source Robot Operating System*. In: *ICRA workshop on open source software*. 2009. 5.
`http://stair.stanford.edu`.

[68] Tomáš Krajník, Matias Nitsche, Jan Faigl, Tom Duckett, Marta Mejail, and Libor Přeučil. *External localization system for mobile robotics*. In: *2013 16th International Conference on Advanced Robotics (ICAR)*. 2013. 1–6.

[69] Peter Lightbody, Tomáš Krajník, and Marc Hanheide. *A versatile high-performance visual fiducial marker detection system with scalable identity encoding*. In: *Proceedings of the symposium on applied computing*. 2017. 276–282.

[70] Jiří Ulrich, Ahmad Alsayed, Farshad Arvin, and Tomáš Krajník. *Towards Fast Fiducial Marker with Full 6 DOF Pose Estimation*. In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2022. 723–730. ISBN 9781450387132.
`https://doi.org/10.1145/3477314.3507043`.

[71] Kristi Žampachů. *Visual analysis of beehive queen behaviour*. Bachelor Thesis, Czech Technical University. Prague, Czech Republic, 2022. To appear.

[72] Kasia Bozek, and Laetitia Hebert. *DenseObjectAnnotation: Tool to annotate objects in images.*
`https://github.com/oist/DenseObjectAnnotation`. Accessed May 7, 2022.

[73] G Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools.* 2000,

[74] F Pedregosa et. al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research.* 2011, 12 2825-2830.

[75] Guillaume Lemaître, Fernando Nogueira, and Christos K Aridas. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research.* 2017, 18 1-5.

[76] Sebastian Raschka. MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *Journal of Open Source Software.* 2018, 3 638. DOI 10.21105/joss.00638.

[77] J D Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering.* 2007, 9 90-95. DOI 10.1109/MCSE.2007.55.

[78] Christoph Heindl, Toka, and Jack Valmadre. *py-motmetrics: Benchmark multiple object trackers (MOT) in Python.* 2022.
`https://github.com/cheind/py-motmetrics`. v1.3.0. Accessed May 4, 2022.

[79] Guido Van Rossum, and Fred L. Drake. *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

[80] Google LLC. *Google Colaboratory.*
`https://colab.research.google.com/`. Accessed May 4, 2022.

[81] Research Center for Informatics. *RCI Cluster.*
`https://login.rci.cvut.cz`. Accessed May 4, 2022.

[82] Lukas Biewald. *Experiment Tracking with Weights and Biases.* 2020.
`https://www.wandb.com/`. Accessed May 3, 2022.

[83] Imbalanced-learn. *Imbalanced Learn - 3. Under-sampling.*
`https://imbalanced-learn.org/dev/under_sampling.html`. Accessed May 8, 2022.

[84] fcakyon. *fcakyon/yolov5-pip: Packaged version of ultralytics/yolov5 + many extra features.*
`https://github.com/fcakyon/yolov5-pip`. v6.1.2. Accessed May 6, 2022.

[85] Joshua Roesslein. *Tweepy: Twitter for Python!* 2020.
`https://github.com/tweepy/tweepy`. v4.7.0. Accessed: March 20, 2022.

[86] *The latest in Machine Learning | Papers With Code.*
`https://paperswithcode.com/`. Accessed May 4, 2022.

[87] Anton Milan, Laura Leal-Taixe, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A Benchmark for Multi-Object Tracking. 2016, DOI 10.48550/ARXIV.1603.00831.

Appendix A
Attached files

A.1 Data

Datasets and trained models can be downloaded from:
`https://owncloud.roboroyale.eu/s/d4IjpZfBFsjultM`.
The folder contains the following items:

- datasets
  - court_detection
  - markerless_bee_detection
  - testing_dataset
  - yolo_detection
- models
  - markerless_bee_detection
  - markerless_bee_tracking
  - court_model.joblib.ml
  - yolo_model.pt

A.2 Code Base

Code Base contains the individual projects on the basis of which the results presented were derived.
The folder contains the following items:

- CourtBeeDetectionExperiments
- DenseObjectAnnotation
- MarkerlessTracking
- rr_courtdetector
- rr_twitter
- rr_yolo_detection
- ByteTrack.ipynb
- YOLOv5.ipynb