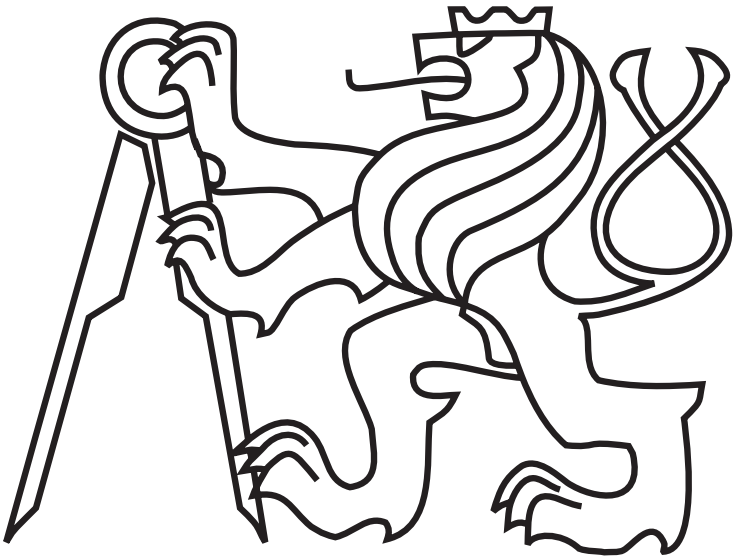
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

MASTER THESIS



Bc. Jiří Janota

Honeybee Comb Mapping Using a Robot

Department of Control Engineering

Thesis Supervisor: **Ing. Jan Blaha**

May, 2024

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Janota Jiří** Personal ID number: **483485**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

II. Master’s thesis details

Master’s thesis title in English:

Honeybee comb mapping using a robot

Master’s thesis title in Czech:

Mapování včelí plástve robotem

Guidelines:

1. Get to know the RoboRoyale observation system and the data generated by its main component [1,2].
2. Get to know the computer vision methods for classifying comb cell contents and states [3,4].
3. Get to know the robot vision methods used to create semantic maps of relevant environments.
4. Implement and integrate a method that can classify comb cell contents.
5. Merge the cells’ contents classification results into a spatially consistent model using filtering methods from mobile robotics.
6. Integrate the map into ROS tools to achieve map visualisation common in robotics.

Bibliography / sources:

[1] M. Stefanec, D. N. Hofstadler, T. Krajník, A. E. Turgut, H. Alemdar, B. Lennox, E. ¸Sahin, F. Arvin, and T. Schmickl, “A minimally invasive approach towards “ecosystem hacking” with honeybees,” Frontiers in Robotics and AI, vol. 9, 2022.
[2] K. Z ampachu, J. Ulrich, T. Rouc ek, M. Stefanec, D. Dvor a c ek, L. Fedotoff, D. N. Hofstadler, F. Rekabi-Bana, G. Broughton, F. Arvin et al., “A vision-based system for social insect tracking,” in 2022 2nd International Conference on Robotics, Automation and Artificial Intelligence (RAAI). IEEE, 2022, pp. 277–283.
[3] T. S. Alves, M. A. Pinto, P. Ventura, C. J. Neves, D. G. Biron, A. C. Junior, P. L. De Paula Filho, and P. J. Rodrigues, “Automatic detection and classification of honey bee comb cells using deep learning,” Computers and Electronics in Agriculture, vol. 170, p. 105244, 2020.
[4] N. Rathore, P. K. Tyagi and D. Agrawal, "Semi-automatic Analysis of cells in honeybee comb images," 2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), Bhopal, India, 2023, pp. 1-4.

Name and workplace of master's thesis supervisor:

Ing. Jan Blaha Department of Computer Science FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **29.01.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until:
by the end of summer semester 2024/2025

Ing. Jan Blaha	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master'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 master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Declaration of authorship

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague on..... .....

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Jan Blaha, for his invaluable guidance, support, and countless discussions about the ways of science. My sincere thanks go to the members of the Chronorobotics Laboratory for their dedication to the RoboRoyale project, without which this thesis would not have been possible. In particular, I am grateful to Tomáš Krajník for providing me with the opportunity to work on such an exciting project.

I want to thank my parents, Zuzana and Jiří, and my sisters, Ludmila and Eliška, for their love and endless support, which allowed me to pursue my studies. I would like to thank our dog, Honey, for his unwavering companionship and my friends, who always cheer me up.

Abstract

Honeybees play a crucial role in our ecosystem, acting as irreplaceable pollinators, directly affecting the global food web and biodiversity of flowering plants. Consequently, many researchers focus on understanding the behavioral strategies and dynamics of the colonies to support their health and growth. The RoboRoyale project aims to strengthen the colony through careful interactions with the honeybee queen using miniature robots introduced to the hive. To assess the effect of these interactions and the colony's health, it is important to monitor the honeybee comb and track its development in time, which is impossible with current biology techniques. In this work, we developed an algorithm for automated long-term comb mapping) using scans of the honeybee comb collected by a moving camera. Using computer vision methods and techniques from visual mapping, we build a spatially consistent semantic map that describes the individual cells and their contents. We do this in a living colony and, therefore, have to solve problems of partial observability, irregular and sparse observations and high levels of noise. The resulting map provides us with a detailed description of the hidden state of the comb in time, including predictions of the future states. We believe that our work could prove important in gathering data and insights needed for the efforts of formulating a complete model of the honeybee colony.

Abstrakt

Včela hraje klíčovou roli v ekosystému, kde působí jako nenahraditelný opylovač a přímo ovlivňuje globální potravinové řetězce a biologickou rozmanitost kvetoucích rostlin. V důsledku toho se výzkum zaměřuje na pochopení chování a dynamiky kolonií, s cílem podpořit jejich zdraví a růst. Projekt RoboRoyale má za cíl posílit kolonii prostřednictvím opatrné interakce s včelí královnou pomocí miniaturních robotů uvnitř úlu. Pro posouzení účinku těchto interakcí a zdraví včelstva je důležité monitorovat včelí plástev a sledovat její vývoj v čase, což doposud užívané techniky v biologii neumožňují. V této práci jsme vyvinuli algoritmus pro automatické dlouhodobé mapování plástve pomocí skenů včelí plástve z pohyblivé kamery. Pomocí metod a technik počítačového vidění a vizuálního mapování stavíme prostorově konzistentní sémantickou mapu, která popisuje jednotlivé buňky a jejich obsah. Mapování probíhá v živém úlu, a proto musíme řešit problémy částečné pozorovatelnosti, nepravidelných a řídkých pozorování a vysoké míry šumu. Výsledná mapa nám poskytuje podrobný popis skrytého stavu plástve v čase, včetně předpovědí budoucích stavů. Věříme, že naše práce by se mohla ukázat jako užitečná při shromažďování dat a poznatků potřebných pro formulaci kompletního popisu včelí kolonie.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Western honeybees (*Apis mellifera*) are crucial to our ecosystem as they serve as vital pollinators to the ecosystem. Their pollination efforts directly influence the global food web while significantly contributing to the biodiversity of flowering plants by facilitating their reproduction. Their significance becomes even more apparent in light of the ongoing decline observed in native pollinators, as well as managed honeybee populations [1]. Hence, monitoring and maintaining healthy and strong bee colonies is essential for both the balance of the natural ecosystem and food security in our society.

To assess the health of a honeybee colony and understand its dynamics, it is crucial to monitor the entire honeybee comb and track its development over time. However, manually conducting such monitoring without disrupting the honeybee colony is infeasible, and to our knowledge, it has not yet been fully automated. In this thesis, we aim to design an original system capable of mapping the contents of individual cells within the comb in time, using composite scans of honeybee comb collected in the 2023 observational season as part of the EU Horizon 2020 RoboRoyale project.

The challenges of such a problem correspond to the task of robotic mapping—the data originate from a moving camera, exhibit properties of sensoric observations, and the observed environment develops in time. First, we need to find objects of interest (comb cells) in the images and, using unreliable odometric information, localize them in metric coordinates of the hive. Then, we have to identify the states of the objects, adding semantic information to the map, for which we have to design a sensor (cell type classifier). Due to the conditions inside the hive, e.g., IR lighting, occlusions, or debris, the sensoric observations are noisy. We employ a temporal filtering method not only to enhance the robustness of our methods but, more importantly, to provide sound information on the system (bee colony) temporal development to biology researchers.

To understand the motivation behind our work, the section "RoboRoyale Project" briefly summarizes the honeybee-related research in the context of modern technologies and introduces the RoboRoyale project in detail. As our problem is complex, an extensive description is provided in the third section, "Problem Statement", which unveils the specifics of the system setup used for data collection, formally states the problem, and outlines our approach to the mapping. The fourth section, "Related Work", explores studies related to our mapping task, specifically focusing on semantic mapping and detecting and classifying honeybee cells, providing theoretical foundations for our work. In the fifth section, "Methods", we present the details of individual components of the semantic mapping pipeline, namely honeybee cell detection, spatial mapping, honeybee cell classification, and temporal filtering. The sixth section, "Testing of Pipeline Components", provides an evaluation of all system components, together with a description of datasets and the annotation processes. Finally, the seventh section, "System Capabilities", presents the results of the honeybee comb mapping, together with an example of the application of our work in biology.

2 RoboRoyale Project

Numerous studies have focused on unraveling honeybees' behavioral strategies and self-organization mechanisms[1], aiming to understand these aspects to support their overall well-being and alleviate the negative effects of the pollination crisis. Specifically, advancements in fields like robotics and artificial intelligence have opened up new avenues in bee research with the deployment of systems for animal-robot interaction directly in natural ecosystems [2, 3]. Additionally, researchers [4, 5] have proposed using fixed-camera systems equipped with marker-based vision systems to observe the activities of honeybee queens and individual bees, respectively. Likewise, in another study [6], an alternative marker-less approach was introduced for achieving the same objective. Beyond honeybee monitoring, these systems hold the potential for supporting nature conservation and protection of natural ecosystems [7, 8].

In addition to the significant ecological role, honeybees possess a remarkable ability to sense chemical odors. Studies have demonstrated that they can be trained, as opposed to dogs, within a matter of hours to detect various chemicals such as explosives or drugs [9]. The training process is demonstrated in Fig. 1. Ultimately, as discussed in studies [9, 10], organisms like honeybee colonies as a whole could even be utilized as biosensors for environmental monitoring.

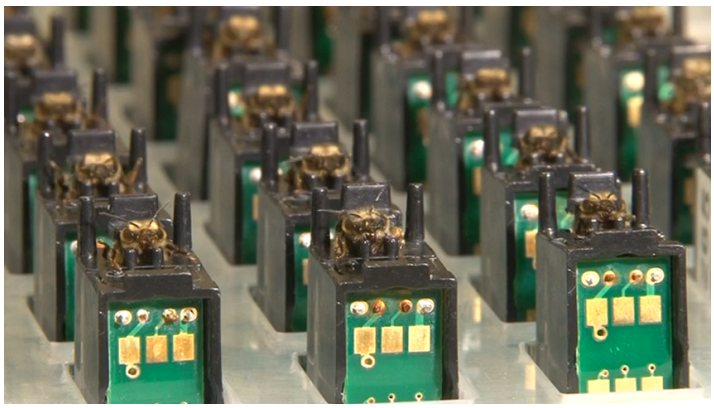


Figure 1: "Bomb-ble bees: Insects to sniff out explosives". Courtesy of [11].

The EU Horizon 2020 project "RoboRoyale" follows in the footsteps of past research endeavors, aiming to bolster honeybee colonies. It aims to support the efficiency and growth of honeybee colonies through a robotic system designed to interact with the honeybee queen, which is practically responsible for the reproduction of the whole colony. The core idea involves introducing biomimetic miniature robots into the hive, which could effectively replace the honeybee queen's courtyard responsible for nursing and feeding her (see Fig. 2). Through careful interactions with the queen, including providing a higher protein inflow and guiding her to optimal regions for egg laying, the project seeks to increase pollen collection and strengthen the colony. [12]

For this purpose, a vertical gantry robotic system, "AROBA", was designed to observe the colony with a focus on the honeybee queen, as described in [13] (currently in the second

[1]It should be noted that there is although still a lack of research addressing questions such as: "Do bees communicate using buzzwords?".

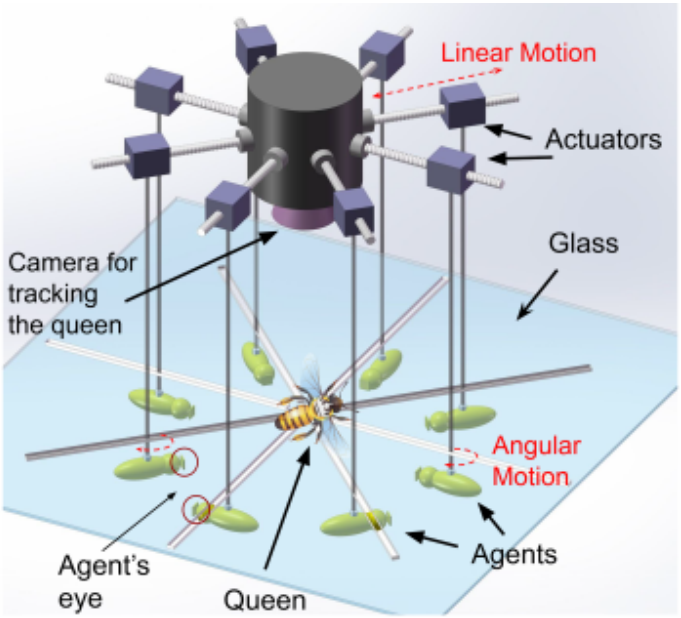


Figure 2: Vision of the RoboRoyale system. Courtesy of [12].

round of reviews). This system is equipped with a high-resolution moving camera and can also accommodate the robotic agents [14]. Additionally, apart from tracking the honeybee queen, the robotic system has the capability to observe the hive at different levels of detail by capturing images at any location within it. (see details in Sec. 3)

3 Problem Introduction

In our robotic system, the honeybee comb is sequentially scanned using the robotic system equipped with a moving camera. This process produces a grid of partially overlapping image tiles, each captured at a different location, as depicted in Fig. 3. By employing image stitching and registration methods (details in Sec. 5.2), individual image tiles can be combined to reconstruct an image of the entire comb (see Fig. 4). Using data from the scanning, we aim to create a semantic map of the honeybee comb and update it as new scans become available, uncovering areas previously occluded by bees and actualizing the map as the colony develops. In this section, we start by describing the system setup and then outline our approach to creating such a map of the comb.

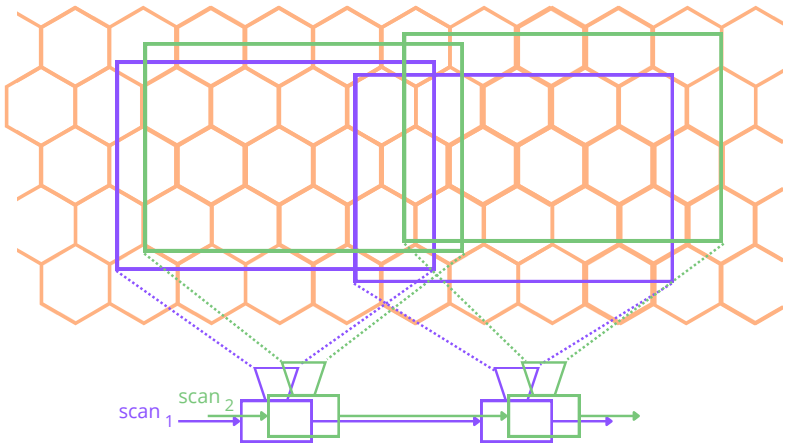


Figure 3: Honeybee comb scanning process. The system sequentially scans the map, producing a partially overlapping image tiles grid. Ideally, the image tiles are always taken at the same locations.

3.1 System Setup

The system comprises one observation hive with two vertically stacked combs of standard size 420 mm × 220 mm, covered with glass panels. The observation system was located indoors at the University of Graz to accommodate the hardware and shield the glass-covered hives from sunlight, with a plastic tube connecting the hive to the outdoors for the bees to fly out. All the observations were then made under a near-infrared LED light with wavelengths from 750 nm to 850 nm, which is invisible to the honeybees to avoid disrupting the colony. For a long-term close-up recording without human involvement, a vertical gantry robotic system named "AROBA" was designed and developed (currently in the second round of reviews [13]), which operates a camera around the hive.

3.1.1 Vertical Robot Gantry System

This robot consists of two independent ball screw drive systems for actuation in the horizontal and vertical directions. Each system is driven by a stepper servo motor coupled with

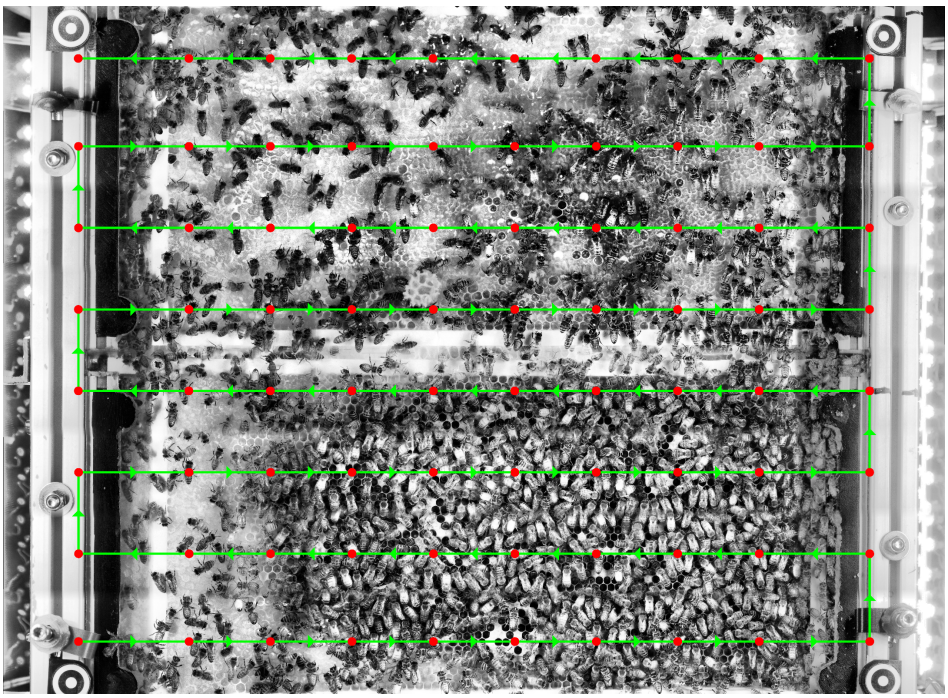


Figure 4: Reconstructed image of the entire honeybee comb from a grid of partially overlapping image tiles using image stitching and linear blending for smooth transitions between images. The illustration also depicts the snake-like trajectory of the robotic system (green).

one ball screw drive for horizontal actuation and two synchronized ball screw drives for vertical actuation to improve stability and precision of the vertical motion. Both systems are stabilized using two linear guides. The developed system in its working environment, along with a schematic diagram and the CAD design, is shown in Fig. 5. Each servo motor can be controlled by position and velocity commands to reach any position in the hive, using embedded controllers that communicate with the main management unit via a serial port. As the system operates close to the hive and to guarantee quality images of the comb, a custom driver minimizing vibrations was developed. A full coverage of the observation hive is achieved by two such mechanisms working in parallel, each recording at one side of the comb.

3.1.2 Vision System

The primary objective of the robotic system is to track the honeybee queen's trajectory for behavior investigation. Additionally, it is tasked with collecting detailed images of the hive to facilitate further investigation of the comb. Hence, the mechanism is equipped with a high-resolution camera See3CAM_CU27, based on the Sony IMX 462 sensor with Kurokesu L087 and Kurokesu L085 lenses, which is mounted on its end effector. The camera provides images with a resolution of 1920 px × 1080 px at a rate of 30 Hz and features controllable zoom and focus. Specifically, it can capture images with a resolution ranging from 16 to 67 μm per pixel, which is sufficient for capturing objects of interest such as honeybee eggs measuring about 1 mm × 0.3 mm. For tracking of the honeybee queen was developed a marker-based vision system [4]. The communication of the vision software with the main management unit

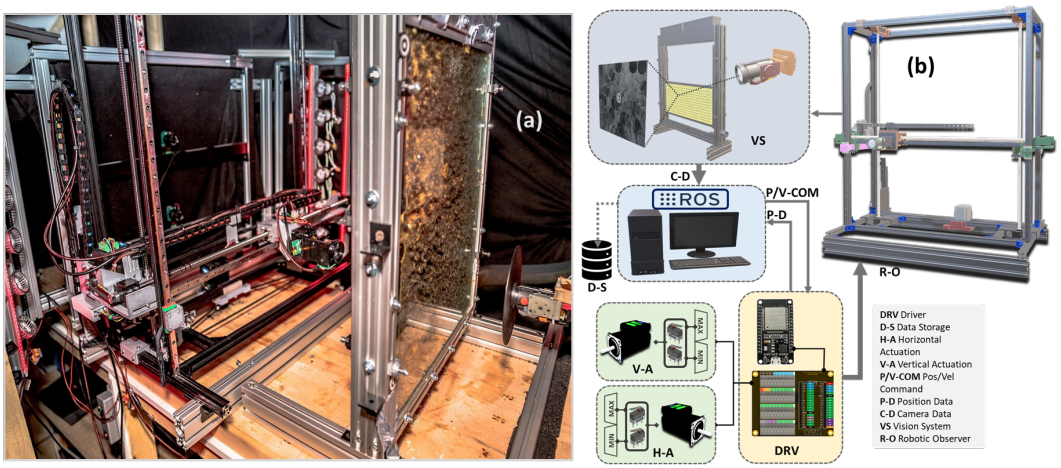


Figure 5: The autonomous observation system for tracking the honeybee queen. The figure (a) shows the system in its real workspace. Courtesy of [13]. In (b) is the CAD design and schematic diagram of the system.

is established via Robot Operating System (ROS) [15].

3.1.3 Odometry Calculation

For our experiments, it's important to know the location of the robotic manipulator's end effector within the hive. Since motor position information can be inconsistent over time, the AROBA system employs additional transformation of the motor positions into hive coordinate frame to improve manipulator position accuracy. It utilizes markers placed in each corner of the hive, as depicted in Fig. 4. The calibration process involves the robotic manipulator navigating to the centers of these markers and storing the motor positions for each. Subsequently, a homography matrix is calculated to project the motor position onto the coordinate frame of the hive. The manipulator is then navigated in the coordinate frame of the hive.

3.1.4 Produced Data

In this work, we use the dataset presented in [13], collected by the AROBA system. The dataset was collected from 20th September 2023 to 19th October 2023. It consists of tracks of the honeybee queen in the comb and sequentially collected scans of the honeybee comb in the format of ROS Rosbags. The honeybee comb scans were captured in two resolutions varying in the level of zoom, which we denote as "unzoomed" and "zoomed" images. On side 0 of the hive, the unzoomed images have a resolution of 67 μm per pixel, and the zoomed ones have a resolution of 34 μm per pixel. On side 1 of the hive, the unzoomed images were collected with a resolution of 67 μm per pixel, and the zoomed images with 25 μm per pixel. In total, 1837 scans of the honeybee comb were collected.

It is important to mention that during data collection, we identified an issue in the data-producing software, resulting in inconsistent coordinates within the hive coordinate frame, leading to variations in the honeybee comb scans' positions. We were not able to fix the

odometry information in the collected data, which are used for the mapping. However, the issue was later resolved and is not present in the new setup of the system.

Additionally, we used three honeybee comb scans collected on 23rd October 2023, which were collected with the new system setup without significant odometry inconsistencies, to evaluate image stitching in Sec. 6.4.

3.2 Problem Statement

The input to the mapping process is a sequence of honeybee comb scans, denoted as $(S_i)_{i=1}^{n}$. Each scan S_i is associated with two attributes $(t, side)_i$, where t denotes the time at which the scan was taken, and $side$ indicates whether the scan was taken from side 0 or side 1 of the comb. A single scan S_i is composed of a grid of partially overlapping image tiles $(I_j)_{j=1}^{Nside}$, which are collected in a sequential snake-like pattern, as shown in Fig. 4. The total number of image tiles N_{side} depends on the side of the hive from which the scan was taken. Specifically, for side 0, there are $N_0 = 90$ tiles, and for side 1, there are $N_1 = 80$ tiles. Each image tile I_j is further associated with a metric position $\mathbf{p}_j = (x, y)_j$ in the coordinate frame of the comb, representing the camera's position at the time the picture was taken.

The objective is to generate a sequence of semantic maps $(M_i)_{i=1}^{n}$ given the sequence of scans $(S_i)_{i=1}^{n}$, so that each M_j reflects the information from the scans $(S_i)_{i=1}^{j}$.

3.2.1 Map Representation

In the map of the comb, we combine a spatial topometric map with semantic information. Our topometric map M is represented as an undirected graph $G = (V, E)$, where $V = \{c_1, ..., c_n\}$ denotes the set of all cells in the map and $E = \{e_1, ..., e_m\}$ the set of edges. Each cell is connected by an edge only to its neighboring cells. The edges represent estimated spatial relations between the cells. Additionally, each cell c_i is described by five attributes $(\mathbf{p}, r, \mathbf{d}, \mathbf{s}, t)_i$, where $\mathbf{p}_i = (x, y)_i$ represents estimated metric position along the horizontal and vertical axes, r_i represents estimated radius of the cell, $\mathbf{s}$ is the belief over possible states of the cell and its content, $\mathbf{d}_i$ is its visual descriptor and t_i is the last timestamp the cell was observed.

3.2.2 Chosen Approach

Within the comb, various types of cells exist—open cells, which may contain eggs, larvae, pollen, nectar, or be empty, and capped cells, housing brood or honey. Ideally, we would apply instance segmentation on the comb images and use it to create the map. However, distinguishing between the different cell types in comb images can pose a challenge, even for human annotators, mainly because capped cells usually lack easily distinguishable borders. As a result, generating annotations, for instance, object segmentation becomes a complex and nearly intractable task. Nevertheless, capped cells maintain their state until they are uncapped, allowing us to focus solely on identifying open cells and extrapolating assumptions about the state of capped cells until they are uncapped again. Therefore, our approach is to perform object-based semantic mapping with cells as the objects.

The map is constructed using detections of individual cells and their classification based on the cell content. More specifically, we establish correspondences between the detections over time and employ temporal filtering to estimate the state of individual cells using the output of the classification as sensor observations. The mapping process will be detailed in Sec. 5.

4 Related Work

In this section, we begin by providing a concise overview of the biology of honeybees in subsection 4.1, highlighting aspects relevant to our work. Subsequently, in subsection 4.2, we delve into the methodologies surrounding semantic mapping and its fundamental components. Specifically, we discuss techniques for visual spatial mapping, strategies for acquiring semantic information, common representations of semantic maps, and methodologies for ensuring temporal coherence. Finally, in subsection 4.3, we explore existing studies focusing on detecting honeybee cells, followed by examining methods for classifying cells based on their contents.

4.1 Honeybee Biology

Honeybees are social insects that live in hives with a structured social order. Typically, a colony consists of three main types of adult bees: one honeybee queen, thousands of worker bees, and hundreds of drones, each fulfilling a unique role. The queen's primary task is to lay eggs and regulate the hive's population, while drones are solely responsible for mating with the queen. Worker bees are responsible for maintaining the hive and caring for the queen and brood. [16]

Despite these role differences, all bees undergo the same development stages: egg, larvae, pupa, and adult. However, the duration of each stage varies (see Tab. 1). The process begins with the queen laying eggs in comb cells. Unfertilized eggs develop into drones, and fertilized eggs, depending on the nutrition provided, become workers or queens. After three days, the eggs hatch into larvae. The larvae are fed for 5-7 days before the cell is capped and the brood enters the pupal stage. Finally, the adult bee emerges from the cell after 8-14 days. [16, 17]. It's important to acknowledge that not all brood successfully reaches adulthood. Throughout each developmental phase, there's a possibility of brood cannibalism, where worker bees consume the developing brood [18], or the brood may fail to develop due to freezing or starvation. In addition to brood cells, the comb contains cells dedicated to storing food resources such as pollen, nectar, and honey, vital for the colony's development.

Table 1: Development stages of honeybee (Apis mellifera) with approximate development times (in days) for queen, workers, and drones [19].

	Egg	**Larva**	**Pupa**	**Total**
Queen	3 days	5 days	8 days	16 days
Worker	3 days	5 days	13 days	21 days
Drone	3 days	7 days	14 days	24 days

4.2 Semantic Mapping

Robotic mapping is a process of creating a representation of an environment that a robot perceives. This is crucial for the robots to operate autonomously without collisions and effectively perform their tasks. The common approach in mobile robotics has been to construct

metrics maps with spatial information about the environment, which robots can use to navigate themselves safely through it [20]. However, with the advancements in robotics comes the trend of integrating service robots into domestic environments, which creates a need for object and scene understanding beyond mere geometry, crucial for bridging the gap in human-robot interaction [21]. For instance, in robotic tasks such as particular object manipulation or guiding individuals within an environment, the robot must understand different types of objects (e.g. cups, chairs, beds), locations (e.g. bathroom, bedroom, kitchen), and even relations between them [22]. This is addressed in semantic mapping by introducing high-level semantic information into the geometric representation of the environment.

Before delving into the topic, it's essential to clarify what a semantic map is. Several works contributed to the definition of a semantic map [21, 23, 24]. We will adopt the formulation from [24], where the authors defined it as follows: *"A semantic map for a mobile robot is a map that contains, in addition to spatial information about the environment, assignments of mapped features to entities of known classes. Further knowledge about these entities, independent of the map contents, is available for reasoning in some knowledge base with an associated reasoning engine."*

The typical approach to semantic mapping is to build the semantic map on top of a metric one [21, 22]. Initially, spatial mapping utilizes data from diverse sensors to construct a geometric map of the environment. Simultaneously, semantic information acquisition techniques, such as place categorization and object detection and recognition, are employed. Subsequently, the acquired semantic information is integrated with the geometric map.

This subsection will begin by discussing methods of visual spatial mapping (sec. 4.2.1). Following that, specific approaches to semantic information acquisition will be outlined (sec. 4.2). Next, will be covered representations of semantic maps (sec. 4.2.3) and temporal coherence in semantic mapping (sec. 4.2.4). Finally, methods of image registration applicable in the context of this work will be presented (sec. 4.2.5).

4.2.1 Visual Spatial Mapping

As mentioned, semantic mapping typically relies on spatial mapping, which constructs a geometric map of the environment [21]. Spatial mapping poses a chicken-and-egg problem with the robot's localization, a challenge commonly addressed through Simultaneous Localization and Mapping (SLAM). SLAM aims to simultaneously create a map of an unknown environment while estimating the robot's position within it. This is a widely researched topic in robotics, with algorithmic approaches varying depending on the sensors used. As this work focuses on camera-based mapping, this section will briefly mention merely visual SLAM (VSLAM) methods, which use visual sensors such as monocular, RGB-D, or a stereo camera as a main input to the system.

The VSLAM techniques can be split based on image registration methods (see 4.2.5) into two categories: direct and feature-based (indirect). The direct methods, such as DTAM [25] or LSD-SLAM [26], utilize the raw camera images to minimize the photometric errors. Such an approach can be especially beneficial in texture-less environments. The feature-based methods, on the other hand, rely on repeatable key-point features that are detected and matched between the images based on their descriptors. These methods can be further categorised

based on the type of features used, for example, the well-known ORB-SLAM2 [27] utilizes low-level ORB feature detector. Some researchers use a combination of low-level features (points) with middle-level ones like planes [28, 29], which can lead to more robustness in texture-less environments. Several studies have also proposed using semantic information for the spatial mapping itself as it can bring valuable information and robustness. More specifically, the usual approach is to utilize objects as high-level features in VSLAM. For instance, in SLAM++ [30], the authors performed object-based VSLAM by matching 3D models of known object instances in the environment. Similarly, [31] proposed a VSLAM method, QuadricSLAM, that does not require prior knowledge of 3D object models. Instead, they utilize neural network for object detection and represent objects using dual quadrics.

4.2.2 Semantic Information Acquisition

Part of semantic mapping involves acquiring semantic information, which is then incorporated into the spatial map. The approaches to semantic mapping can be categorized based on the process of acquiring this information [21, 32]. We adopt the categorization from [32] that proposes two categories: human-assisted and automatic sensor-based information acquisition.

Human-Assisted Information Acquisition Human-aided methods involve collecting semantic information with human input, facilitating the robot's understanding of the environment. For example, in [33], artificial signs with encoded semantic information are placed in the environment, and the robot automatically extracts it using optical character recognition (OCR). Other studies, like [34] and [35], utilize human-robot interaction for information extraction through dialogue and verbal utterances, respectively. In [36], verbal interaction is extended by interaction with the use of a laser pointer.

Automatic Sensor-Based Information Acquisition The second category encompasses techniques where robots autonomously gather semantic information from perception without requiring human-robot interaction. When prior knowledge about objects in the environment is available, semantic information can be obtained using a database of predefined objects [24, 30, 37, 38]. However, creating such a database can be challenging. Consequently, many studies opt for object detection using extensive datasets instead. For example, in [39], the authors utilized SIFT-based object recognition, collecting training samples through internet image search engines. With recent advancements in deep learning, researchers often rely on trained object detectors, such as the Single-shot Multi-box Detector (SSD) utilized in [40] or the Faster R-CNN object detector in [41], both trained on COCO dataset [42]. In [43], a full deep-learning approach was adopted, using YOLOv3 for object detection and a convolutional long short-term (ConvLSTM) recurrent neural network for visual odometry.

In contrast to methods focused on identifying specific objects in the environment, alternative approaches concentrate on place categorization and scene interpretation. Some studies address this challenge by categorizing images into semantic place categories. For instance, in [44], the CENTRIST visual descriptor [45] was applied for this task. Another method, as demonstrated in [46], involved using a convolutional neural network for place classification, complemented by a one-vs-all classifier to recognize places not covered in the training set. Additionally, researchers also solve the problem by means of semantic segmentation. Early works

utilized classification and clustering methods for the segmentation and labeling of 2D grid maps [47, 48, 49]. Similar techniques were adapted for the segmentation of 3D point clouds in studies like [50] and [51], where planes in the point cloud were extracted and classified. Moreover, some studies employ Random Forests to segment RGB-D images for the semantic labeling of 3D maps [52, 53]. Furthermore, inspired by the success of deep learning in computer vision, recent works have embraced convolutional neural networks for semantic segmentation [54, 55]. Additionally, certain methods utilize information about the objects in the environment to reason about the scene based on provided conceptual knowledge [35, 56, 57].

4.2.3 Map Representation

Following the completion of spatial mapping and semantic information acquisition, the next challenge in semantic mapping involves integrating the semantic data with the spatial map. The methods employed often depend on the specific techniques for acquiring the semantic information. Most of the works introduced in the previous subsection attach the semantic information to the geometric representation simply by labeling the created 2D [44, 48, 49], or 3D [40, 50, 51, 52, 53, 54, 55] spatial maps. Some approaches explore the feasibility of using object-centered maps, such as SLAM++ [30], which leverage semantic information even for constructing the spatial map. Similarly, in [41], the authors construct the map based on detected objects. Additionally, a few researchers utilize more sophisticated representations of semantic maps that adopt a hierarchical structure. For instance, in [56], they distinguish spatial hierarchy, representing the spatial map on different detail levels, and conceptual hierarchy, representing knowledge about the environment. A similar multi-layer representation with a spatial map, consisting of different abstraction levels and a conceptual map, was adapted in [35].

4.2.4 Temporal Coherence

In the realm of mobile robotics, mapping is typically augmented by filtering techniques to address the inherent uncertainty in state estimation and sensor measurements. Likewise, in semantic mapping, numerous studies leverage additional techniques to enhance the reliability of inferred semantic information over time. As stated in [21], even a simple 2D occupancy grid can be considered a semantic map, for which the well-known occupancy grid mapping algorithm, employing the Bayes filter, is commonly utilized [58]. Moreover, Bayes filtering seems to be well-suited for various applications in semantic mapping, as evidenced by its adoption in several works to ensure the temporal coherence [44, 46, 52, 53, 55]. In [40], the authors took a simpler approach, accumulating object classification confidences over time. Additionally, another reported approach in semantic mapping is to utilize Conditional Random Fields (CRF) [50, 54].

4.2.5 Image Stitching

Part of the visual spatial mapping in the context of our work is image stitching, a technique used to combine multiple images with overlapping fields of view into a single composite image. Akin to automated microscopy, our system produces a grid of overlapping images that must be registered to generate a cohesive representation of the observed scene. Typically, the camera

can be considered parallel to the observation plane, simplifying the image registration process solely to the estimation of horizontal and vertical translation. This overview primarily covers methods utilized for stitching a grid of overlapping image tiles related only by translation, with two prevalent approaches for image registration: feature-based and direct approach.

Feature-based image registration The feature-based approach relies on a feature detector to identify key points of interest, which are then described using descriptors. The key points are then matched between images based on their descriptor similarity. The final transformation is then estimated from the matched correspondences, most commonly using the Random Sample Consensus (RANSAC) algorithm. [59, 60, 61, 62, 63]

Numerous studies have explored the application of the feature-based approach for image stitching in the context of electron microscopy. A predominant choice for feature detection in these studies is the Scale-Invariant Feature Transform (SIFT) [59, 60, 62, 63], introduced by Lowe [64]. However, the particular choice of feature detector depends on the specific application. For example, [61] showed that in ceramic microscopy, using Speeded-Up Robust Features (SURF) and Principal Component Analysis (PCA) for descriptor dimension reduction led to better performance than SIFT. In [65], the authors introduced Virtual ALignment of pathology Image Series (VALIS), an automated pipeline for image registration with support for various feature detectors.

With the recent advances in deep learning, there is a growing trend in studies proposing the replacement of components in the traditional feature-based pipeline with machine learning (ML)-based methods. Particularly noteworthy is the work in [66], where authors introduced Superpoint, a CNN-based feature detector and descriptor. Additionally, [67] demonstrated the feasibility of using Graph Neural Networks for matching features. To address the challenge of separate neural networks for feature detection and matching, researchers in [68] introduced Detector-Free Local Feature Matching with Transformers (LoFTR). Notably, [63] showcased the significant potential of using LoFTR instead of SIFT in the context of electron microscopy and image tile stitching.

Direct image registration The direct approach to image registration aims to find the best match by maximizing a similarity metric across all possible shifts. Instead of using a computationally complex sliding-window approach, researchers often opt for more efficient techniques like Phase Correlation (PC), which leverages the Fourier Shift Theorem and is more resilient to noise and intensity variations. [69, 70, 71, 72, 73, 74, 75].

In [70], for instance, PC was applied to downscaled images, and a precise solution was obtained using a sliding-window approach with Normalized Cross-Correlation (NCC) on the original unscaled images. In another study [73], researchers evaluated a few highest peaks from PC and selected the best one based on NCC values. Additionally, authors in [76] combined feature-based and sliding-window approaches, identifying key points and using fixed-size windows around these features for precise translation estimation using NCC.

It's important to note that the presented direct approach mainly focuses on estimating translation between images. However, alternative techniques, such as the Fourier-Mellin transform, can extend these capabilities to include rotation and scale estimation, as demonstrated in [77] for underwater photo map stitching.

Global optimization The methods introduced for image registration are designed to estimate transformations between individual pairs of images. However, when dealing with a grid of image tiles, where each image typically overlaps with multiple adjacent tiles, the necessity for global optimization methods arises. These methods aim to minimize misalignment across all pairs of overlapping images, consequently mitigating the accumulation of image registration errors during the stitching process. While bundle adjustment, relying on 3D geometry, is a widely used solution in such situations, researchers focusing on estimating rigid transformations, as in automated microscopy, often opt for simpler approaches.

In the study by [59], the optimal rigid transformation was achieved by iteratively minimizing the square displacement of landmarks identified by SIFT. Another common strategy involves constructing a graph with image tiles as nodes and connecting edges between overlapping pairs of images. Many studies employ the minimum spanning tree algorithm to identify the optimal subset of edges that connect all tiles and minimize the resulting misalignment. Some researchers enhance this approach by weighting the edges with the number of feature points as weights of edges [76], or values such as Normalized Cross-Correlation (NCC) [73, 75], demonstrating great potential for an improvement. Additionally, [63] proposed to perform the optimization with graph-based 2D Simultaneous Localization and Mapping (SLAM) method GraphSLAM.

Another widely adopted technique is to construct an over-constrained system of linear equations and minimize the sum of all pairwise transfer errors through least squares [70, 71] or weighted least squares [69, 72, 74, 75], employing correlation values as weights.

4.3 Honeybee Cell Detection and Classification

The detection and classification of honeybee cells is a subject of ongoing research, primarily driven by concerns surrounding the Varroa destructor mite, an external parasite known to severely affect bee colonies by causing malformation and weakening of the colony while also transmitting viruses. In response, scientists are investigating alternative approaches, such as evaluating the removal rates of dead broods as a potential indicator of a colony's resistance to Varroa mites [78]. The assessment of colony hygienic behavior is typically conducted manually by beekeepers, demanding a considerable time and effort [79, 80]. Consequently, many studies aim to reduce the workload by automating the examination process.

4.3.1 Honeybee Cell Detection

The detection of honeybee cells has been explored through various methodologies, with a notable portion of studies focusing on images devoid of bees. In these instances, standard computer vision techniques are often utilized to detect the honeybee cells. For example, in [81], researchers employed the Canny edge detector to delineate contours within the image, subsequently utilizing a feature set for the classification of uncapped cells, allowing them to track uncapping events within recordings. Additionally, another study detailed in [82] utilized a convolution-based method for capped brood cell detection. Here, researchers generated circular masks corresponding to cell sizes and applied thresholding to differentiate between cell edges and interiors. Furthermore, in [83], authors proposed a methodology involving image thresholding and partitioning into so-called superpixels to detect individual cells.

Many researchers also leverage the circular shape and non-overlapping nature of honeybee cells. Typically, they employ preprocessing steps, such as applying Gaussian or Bilateral filters to suppress noise in the images, followed by image normalization using Contrast Limited Adaptive Histogram Equalization (CLAHE) or Automatic Color Enhancement (ACE). Subsequently, they commonly utilize the Circle Hough Transform (CHT) [84] for circle detection [85, 86, 87, 88, 89, 90]. Moreover, [88] extended this pipeline by integrating a U-Net, a convolutional neural network for binary segmentation, to filter out false positive cell detections outside the honeybee comb. Even though these methods produced promising results, the CHT requires application-specific parameter tuning and may encounter challenges under varying light conditions or in the presence of substantial noise.

Few studies have also ventured into detecting honeybee cells in images where bees are present. In such cases, the challenges posed by the dynamic nature of bees, including occlusions and varying lighting conditions, necessitate the use of more advanced techniques. A popular choice is to leverage deep learning for cell detection. For instance, the authors of [6] embraced a deep learning approach and trained a U-Net neural network to detect capped brood cells. Similarly, in [91], a convolutional neural network for segmentation capable of localizing centers of open cells not occluded by bees was used.

4.3.2 Honeybee Cell Classification

The classification of cells is implicitly included in the previously introduced cell detection methods designed for the detection of open or capped cells. This categorization is also adopted in [92], where the authors compared multiple classifiers, namely Minimum Distance Classifiers (MDC), Decision Trees (DT), and Support Vector Machines (SVM), for classification into three classes: occluded, closed and uncapped. Similarly, [82] classified capped cells using histogram comparison with labeled instances.

Beyond this classification approach, some studies have expanded the classification schema to encompass a broader range of cell categories based on their content. For instance, in [86], researchers developed commercial software capable of classifying cells into ten distinct categories, covering all stages of bee development, pollen, and nectar. However, we weren't able to find details regarding their methodology. More recent work, [88], defined seven categories of cells based on their content, including egg, larvae, brood, pollen, nectar, honey, and empty/other. They constructed a dataset annotated by experienced beekeepers and experimented with various convolutional neural network architectures for classification, yielding promising results. Additionally, in [91], the author focused on classifying stages of honeybee larvae from images. More specifically, two convolutional neural networks were tried. One was for regression of the larvae's age, which did not prove sufficient for this task, and the other one was for classification, which led to solid results, especially for classifying the age of older larvae.

5 Methods

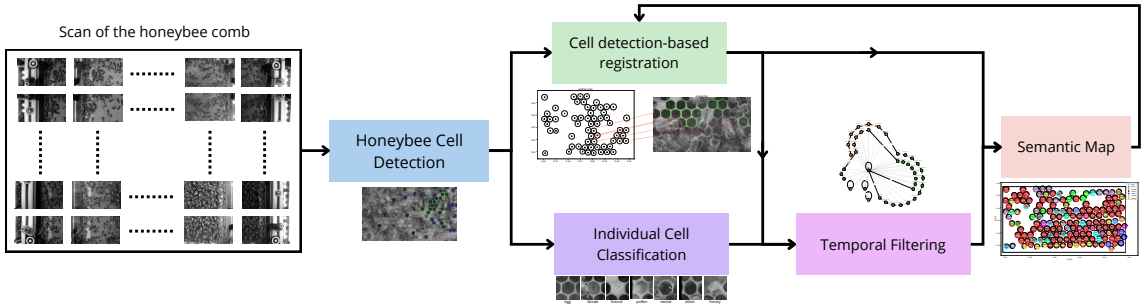


Figure 6: Our honeybee comb mapping pipeline

The semantic mapping process is outlined in Fig. 6. We follow the notation introduced in section 3.2. The input to the mapping includes the previous map M_{i-1} and sequentially collected scan of the entire honeybee comb S_i, comprising partially overlapping image tiles. At the beginning, an initial map M_0 is created using first $N_{\text{initial scans}} = 5$ comb scans, which are then not used further for the mapping (see section 5.2.3). Since localization is provided by the robotic system, as detailed in section 3.1.3, it is ommited from the pipeline.

When updating the map, initially, cell detection is applied to the collected image tiles (section 5.1). These detections are then fed into the spatial mapping module that establishes correspondences between the map and the detected cells using image registration techniques and updates the spatial map accordingly (see section 5.2).

Additionally, the cropped images of the individual cell detections are passed to the classification module (section 5.3), which categorizes the cells into one of seven observation categories based on their content: egg, larva, capped brood, pollen, nectar, honey, or unknown. The classifications, combined with the established correspondences between the map and the detected cells, are then filtered in time by a temporal Bayes Filter to estimate the state of the cells in the map (section 5.4).

5.1 Honeybee Cell Detection

In the task of honeybee cell detection, conventional methods often rely on the Circle Hough Transform (CHT) [84], as highlighted in Section 4.3.1. However, these methodologies typically operate on comb structures without bees. The presence of bees in the images further complicates the detection process. Few studies have specifically addressed the challenge of cell detection when they are heavily occluded by crawling bees in images, particularly utilizing neural networks for image segmentation (see section 4.3.1). In our work, we experiment with both the conventional CHT approach and a modern deep-learning paradigm employing neural network architectures for object detection. With the aim of supporting the networks' capability of detecting open cells not occluded by bees, we distinguish two distinct classes: fully visible cells and partially occluded cells. The detection is performed on individual image tiles of size 1920 px × 1080 px from the honeybee comb scans.

5.1.1 Circle Hough Transform

The Circle Hough Transform (CHT) [84] is an algorithm designed to detect circles with a predefined radius within images. Initially, it employs an edge detection algorithm, commonly the Canny edge detector. From edges, circle candidates are identified by voting, with local maxima selected as the circles. When the task involves detecting circles with varying radii, the process typically consists of two stages. In the first stage, potential circle centers are identified for each possible radius. Then, in the second stage, the optimal radius for each circle is determined through a voting process encompassing all possible radii for all detected circle centers.

Given the variability in exposure and illumination levels across comb images, which may present challenges to the Circle Hough Transform (CHT), we started by image preprocessing. Specifically, we experimented with two image normalization techniques: Histogram Equalization (HE) and Contrast Limited Adaptive Histogram Equalization (CLAHE). To filter noise in the images, we tried the application of both Gaussian and Bilateral Filters.

We use the implementation of the CHT in the Python library OpenCV [93], allowing specifying multiple radii of the circles. All detections of the cells from CHT are considered to be of class "fully visible cell". The specific parameters of this method were determined experimentally. Tab. 2 provides an overview of all parameters.

Table 2: Parameters of the Circle Hough Transform

Parameter	**Value**
minimum distance between circles	40
minimum radius of the circles	30
maximum radius of the circles	45
Canny edge detector threshold	40
accumulator threshold	18

5.1.2 Object Detection Neural Networks

The presented approach with CHT tends to produce false positive detections, primarily due to noise and the presence of bees in the images (see Sec. 6.1). To enhance performance, the false positives could be further filtered out, for example, by restricting the area of interest in the images, as in [88]. Another avenue would be to employ an additional classifier in an R-CNN style, utilizing the CHT as a region proposal algorithm. However, it's worth noting that this strategy could substantially impact achievable recall already at the region proposal stage. Thus, in this work, we opted for a deep-learning approach and trained two distinct architectures for object detection.

Faster R-CNN The first selected network is Faster R-CNN, a two-stage neural network designed for object detection tasks [94]. The architecture is illustrated in Fig. 7. It should be noted that the particular numbers of channels in the figure for the ROI heads do not comply

with our case as different number of classes is used. The network consists of a backbone network with a feature pyramid network (FPN) for feature extraction, a region proposal network (RPN) for generating regions of interest (ROIs) from anchors, and regression and classification heads. These heads process the region proposals by classifying them and refining their bounding boxes.

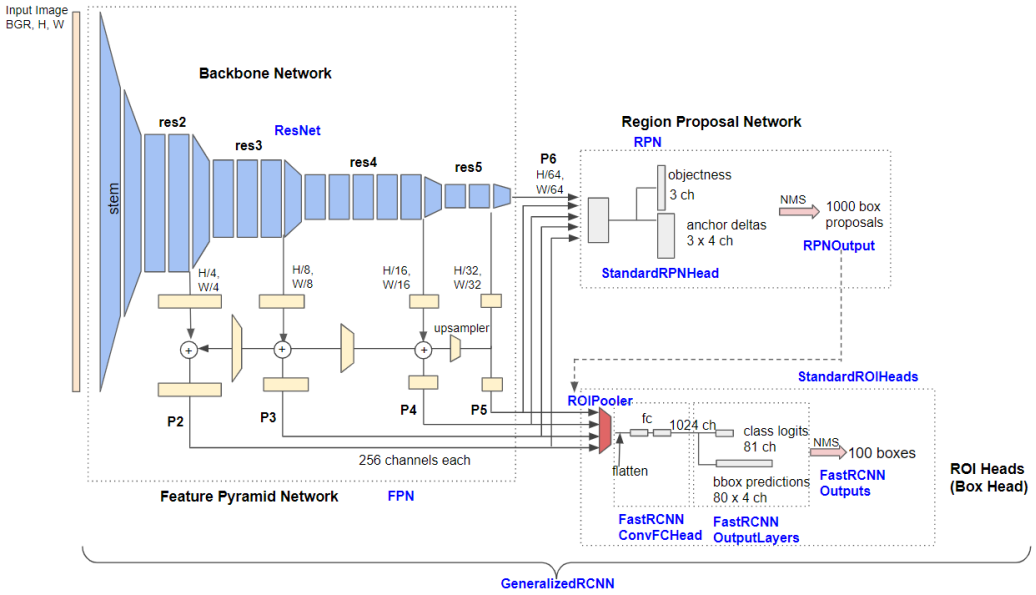


Figure 7: Faster R-CNN architecture with additional feature pyramid network (FPN). Courtesy of [95]

.

We utilized the implementation of Faster R-CNN from the Python package TorchVision [96]. Experiments involved employing two backbones, ResNet-50 and ResNet-18 [97], both employing additional feature pyramid network (FPN) and pre-trained on the COCO dataset [42]. While we maintained default network parameters, we also explored alternative settings that produced similar results. Similarly, we used the loss functions predefined in the TorchVision package for the training. Smooth L1 loss was utilized for bounding box regression in both the RPN and regression head. For object classification in the RPN, binary cross-entropy loss was applied to distinguish between object and non-object classes. The classification head employed categorical cross-entropy loss to classify objects into three classes: background, fully visible cell, and partially occluded cell.

YOLOv5 The second architecture we chose is YOLOv5, a single-stage neural network for object detection, selected for its faster inference compared to Faster R-CNN. Notably, it differs by omitting a separate network for region proposals. Instead, it features a backbone for feature extraction, a neck for extracting feature pyramids from the backbone's feature maps, and head networks for predicting object class, confidence objectness scores, and bounding boxes. The overall architecture of the network is shown in Fig. 8.

We opted for the small version YOLOv5s6 and utilized the original implementation from the Python Ultralytics package [99]. Similarly to Faster R-CNN, we maintained default parameters and utilized a backbone pre-trained on the COCO dataset [42]. Likewise, we use

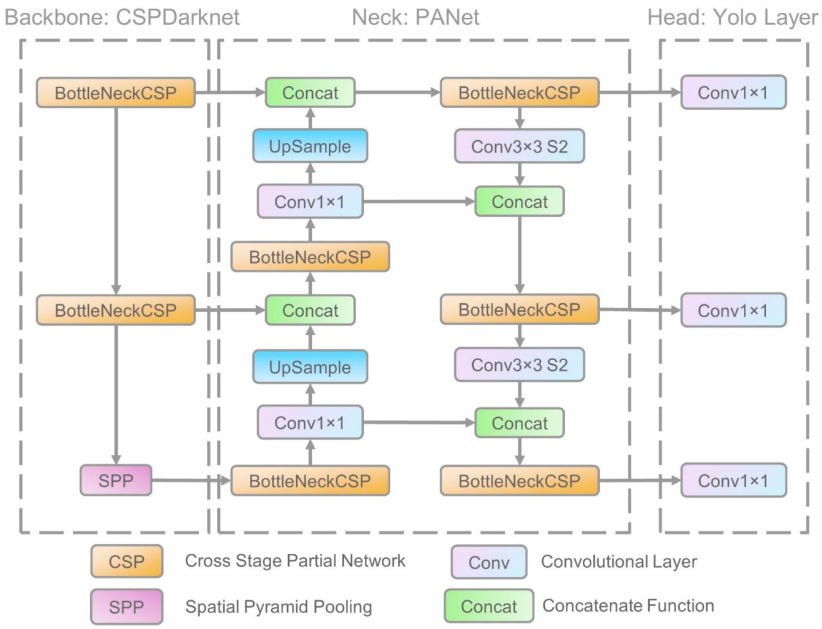


Figure 8: YOLOv5 architecture. Courtesy of [98]

the loss functions, which were predefined in the original implementation. For the regression head, CIoU loss was employed. The classification head, distinguishing between object and non-object classes, employed binary cross-entropy loss. Additionally, the original implementation utilized binary cross-entropy loss as well for the head, which classifies objects into fully visible and partially occluded cells, to address the fact that objects may generally belong to more than one class.

Training Process The training process was similar for both architectures. Initially, the pre-trained backbone was kept fixed while the remainder of the network was trained. Once the network's performance plateaued on the validation data, the backbone was unfrozen, and the entire network was fine-tuned. Additionally, to enhance the models' ability to generalize and accommodate variations in exposure and illumination within comb images, we employed image augmentations. These included random adjustments to brightness and contrast, defocusing, random gamma correction, and horizontal and vertical flips. Notably, these augmentations were specifically generated each time an image was utilized during training. The dataset used for the training is described in detail in Sec. 6.1.

5.2 Spatial Mapping

In accordance with Sec. 3.2, the spatial map is represented as a graph composed of cells defined by their position, radius, and visual descriptors. This subsection provides an in-depth explanation of the spatial map creation process. The mapping initiates by generating an initial map, denoted as M_0, derived from the first $N_{\text{initial scans}} = 5$ scans of the entire honeybee comb (Sec. 5.2.3). Subsequent scans serve to update this initial map. To create the initial map and perform map updates (detailed in Sec. 5.2.4), correspondences between detections in the scans

and also between the scans and the existing map need to be established.

We experimented with two distinct methods to achieve robust matching of the cells. Initially, we implemented two image registration methods to refine the alignment of the collected comb images within the incoming scan (Sec. 5.2.1). However, in the end, we decided not to utilize this component in the final pipeline, as it did not lead to satisfactory results. Instead, we employ feature-based image registration using the cell detections and their visual descriptors extracted from the comb images (Sec. 5.2.2). This method can serve both to align the image tiles in the scan and to establish correspondences between the cells in the map and detections.

5.2.1 Image-based Registration

In section 4.2.5, it was discussed that in the field of automatic microscopy, a similar grid of overlapping images is typically collected. To generate a stitched image from these tiles, the prevailing approach involves aligning the images through image registration methods. While our primary objective is not to create a seamlessly stitched image of the entire comb, achieving precise alignment of the image tiles within the scan can enhance odometry information and consequently improve the mapping process. We evaluated the performance of both the correlation-based and feature-based approaches, as each has different advantages and disadvantages. The comb images exhibit variations in illumination and exposure, and the overlaps differ significantly in content due to bee movement, presenting challenges for the correlation-based approach. Conversely, the feature-based approach may encounter difficulty due to the lack of unique features in the images, as both cells and bees generate highly repetitive patterns.

Correlation-based Approach The correlation-based method evaluates all potential translations between the images to determine the best alignment based on the maximal value of a similarity metric. To streamline this process, we utilized odometry information as prior knowledge, restricting deviations from it in both the horizontal and vertical axes. The similarity metric is then computed only on the estimated overlapping regions of the images. To mitigate the effects of varying exposure and illumination, we applied image normalization and histogram equalization techniques. For the similarity metric, we selected the Normalized Cross-Correlation Coefficient (NCC), defined as follows for two image tiles I_1 and I_2 with means $\bar{I_1}$ and $\bar{I_2}$:

$$NCC = \frac{\sum_{x,y}(I_1 - \bar{I_1})(I_2 - \bar{I_2})}{\sqrt{\sum_{x,y}(I_1 - \bar{I_1})^2}\sqrt{\sum_{x,y}(I_2 - \bar{I_2})^2}} \tag{1}$$

Feature-based Approach In the feature-based approach, we utilize SIFT [64] for keypoint detection and description. Similar to the correlation-based approach, we apply image normalization and histogram equalization. Likewise, we incorporate odometry information as priors on translation, limiting possible deviations from it in both the horizontal and vertical axes. Since the images do not vary in scale, we only match two keypoints if the ratio of their scales is less than 1.5. To determine the translation between images, we adopt a histogram voting method, selecting the translation with the most votes.

Global Optimization To tackle misalignment issues among all pairs of image tiles, we employed least squares optimization, often reported in other studies (see Sec. 4.2.5). We established a graph $G = (V, E)$, where V denotes the positions of individual image tiles $\mathbf{p}_i = (x, y)_i$, and E represents a set of edges connecting neighboring image tiles. The estimated translations $\mathbf{d}_{ij}$ between image pairs were then used to formulate an over-constrained system of linear equations, with the position of the first tile anchored to the origin $(0, 0)$. This results in the following problem, which can be solved using least squares optimization:

$$\min_{\forall i:\, \mathbf{p}_i} \sum_{(i,j)\in E} \sum_{c\in\{x,y\}} \big((\mathbf{p}_{i,c} - \mathbf{p}_{j,c}) - \mathbf{d}_{ij,c}\big)^2 \quad \text{s.t.} \quad \mathbf{p}_1 = (0, 0). \tag{2}$$

While there is an option to weigh the contribution of the individual estimated translations, for example, by confidence indicated by cross-correlation values, we do not utilize this measure due to its sensitivity to the randomness of bees' motion in our case.

5.2.2 Cell Detection-based Registration

The image registration methods discussed earlier face challenges unique to our application, such as dealing with repetitive patterns or variations in images due to the movement of bees. Hence, we opted for an alternative approach where cell detections serve as features. This subsection will first describe the method of obtaining descriptors for the cell detections, followed by a detailed explanation of the image registration algorithm. It's worth noting that all visualizations in this section and all the thresholds mentioned are based on images with a resolution of 25 μm per pixel.

Neural Network for Image Similarity Evaluating the similarity between images is essential in computer vision for tasks such as image retrieval, object tracking, or facial recognition. Our task is to assess the similarity of images of individual cells. One effective method for this is contrastive learning with the Siamese network, which learns to map images into a high-dimensional embedding space where similar images are closer together while dissimilar ones are farther apart. The term "Siamese network" originates from the concept that images in pairs are processed through identical networks with shared weights. In our case, as visualized in Fig. 9, we adopt triplet loss, introduced in [100], to learn the notion of similarity and dissimilarity. The triplet loss function takes a triplet of images (A, P, N) as an input, where A is an anchor image, P is a positive sample visually similar to the A, and N is a negative sample, dissimilar to both A and P. The triplet loss function is defined in Eq. 3, where d represents the distance metric, f denotes the neural network function, and $margin$ specifies the minimum desired distance between positive and negative pairs.

$$\mathcal{L}_{\text{triplet}}(A, P, N) = \max(d(f(A), f(P)) - d(f(A), f(N)) + margin), 0) \tag{3}$$

We utilize the XResNet-18 neural network architecture, which is based on ResNet-18 [97] with additional tweaks introduced in [101]. The architecture is visualized in Sec. 5.3 in Fig. 14. However, for the image similarity, we reduce the size of the network by using a smaller number

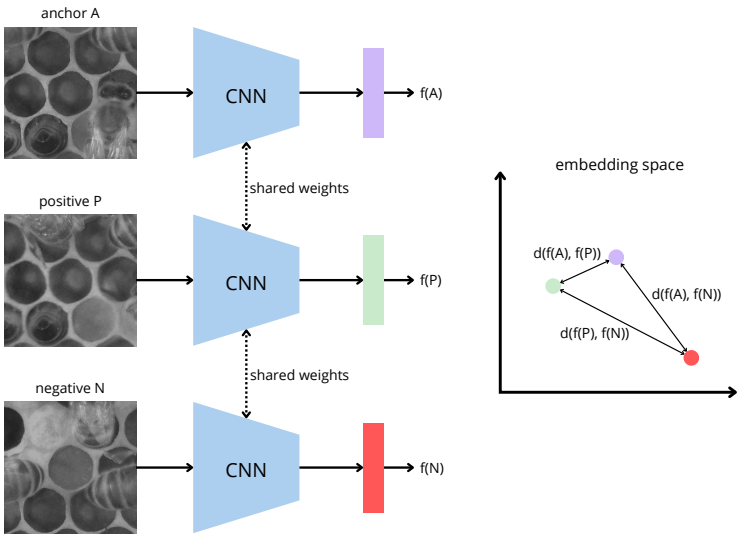


Figure 9: Obtaining the embedding vectors for a triplet comprising anchor, positive and negative images. On the right side is the desired result, where the similar images are closer while the dissimilar are farther apart.

of channels, starting with 8 channels in the input stem, resulting in a compact network with approximately 711 thousand trainable parameters. Moreover, the output of the final layer of the network is a 64-dimensional embedding vector. The used distance metric was cosine distance, defined for two vectors $\mathbf{u}$ and $\mathbf{v}$ as:

$$d_{cosine}(\mathbf{u}, \mathbf{v}) = 1 - \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} \tag{4}$$

Comparing the similarity of individual cells alone may not provide adequate discrimination, especially since not corresponding open cells tend to exhibit high similarity. However, even when non-corresponding individual cells share similarities, their surrounding neighborhoods often vary. Therefore, rather than directly comparing images of cells, we compare images of cells along with their neighborhoods, as depicted in Fig. 9, considering both individual cell features and their contextual information.

Similarly to training the object detection networks, we incorporate image augmentations such as random adjustments of brightness and contrast, defocusing, and random gamma correction to address variations in exposure and illumination. We also employ vertical and horizontal flips to prevent overfitting. Additionally, to encourage the network to focus on the neighborhood of the cells, we randomly mask the contents of the cells with a square mask full of zeros. This augmentation could help the network focus on contextual information surrounding the cells rather than solely relying on individual cell features.

Image Registration When matching the detections between images, the problem can be tackled by calculating the centers of the detections and employing methods for alignment of the resulting two 2-dimensional point clouds. We can use the fact that the images, and thus also the point clouds, are related only by translation in horizontal and vertical axes. One

possible approach would be to use the Iterative Closest Point (ICP) algorithm. However, such a method may fail in cases where there is only a small number of correspondences between the detected cells in both image tiles, and hence, we resort to a sampling algorithm instead.

We denote the centers of the detections in the pair of images as $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. The algorithm iteratively samples a center $\mathbf{c}_2$ from $\mathcal{C}_2$ that is close enough to some center from $\mathcal{C}_1$. The threshold for the distance is determined by the maximum allowed deviation from the odometry, denoted as $G_{\text{threshold}}$. For the images with a resolution of 25 μm per pixel, used for the comb mapping, this threshold is set to a default value of 140 pixels, which is about 3.5 mm.

If there are multiple centers from $\mathcal{C}_1$ close to the sampled center $\mathbf{c}_2$, the following process is performed consecutively for each such $\mathbf{c}_1$. First, the translation between $\mathbf{c}_1$ and $\mathbf{c}_2$ is calculated. Next, this translation is applied to all centers in $\mathcal{C}_2$, resulting in a set of translated centers denoted as $\mathcal{C}_2'$. Subsequently, correspondences are established between the translated $\mathcal{C}_2'$ and $\mathcal{C}_1$. For each center in $\mathcal{C}_2'$, the algorithm selects the closest center from $\mathcal{C}_1$ as its correspondence if it is within a distance of $L_{\text{threshold}}$, which is set to 50 pixels for zoomed images, about 1.3 mm.

Finally, a criterion function is calculated for the established correspondences. If this criterion value is better than the previous best value, it is used to update both the best criterion value so far and the associated best correspondences. The pseudocode for the registration process is provided in Algorithm 1.

We experimented with a few suitable criterion functions. The first one relies solely on the spatial structure and uses the number of established correspondences in the process as the criterion value, which is then maximized. However, to make the registration more robust, we decided to also employ visual descriptors of the cells. By utilizing these descriptors, we are able to assess the confidence of the registration, enabling us to reject potential faulty matches based on that.

Image Similarity Criterion for Matching Image Pairs When matching pairs of image tiles, we calculate the cosine distances between the established correspondences. The criterion for the whole registration is the average of the individual cosine distances, which we aim to minimize. If the value of the criterion for the final established correspondences exceeds 0.5, the registration is considered failed and is rejected. This rejection threshold was chosen based on the distribution of cosine distances between anchor and positive samples versus anchor and negative samples, calculated using the validation dataset used in the training process of the network for image similarity, depicted in Fig. 10.

We can express the probability $p(\text{class} \mid d)$ as in Eq. 5, where d is the cosine distance and $\text{class} \in \{0, 1\}$ refers to either positive or negative class. If we assume that the prior probabilities $p(\text{class} = 0)$, $p(\text{class} = 1)$ are identical, the problem of classifying cell correspondence to positive or negative class simplifies to solely taking the argument of maxima of the probabilities $p(\text{class} = 0 \mid d)$ and $p(\text{class} = 1 \mid d)$. Instead of taking the argument of maxima directly, we apply a threshold on the cosine distance d. As can be seen in Fig. 10, a reasonable value for the threshold is around 0.5.

$$p(\text{class} \mid d) = \frac{p(d \mid \text{class}) \cdot p(\text{class})}{p(d \mid \text{class} = 0) \cdot p(\text{class} = 0) + p(d \mid \text{class} = 1) \cdot p(\text{class} = 1)} \tag{5}$$

Input: $\mathcal{C}_1$, $\mathcal{C}_2$, (optionally descriptors) $\mathcal{D}_1$, $\mathcal{D}_2$
Output: *best_correspondences*
Initialize *candidates* as an empty list;
for *each* $\mathbf{c}_2$ *in* $\mathcal{C}_2$ **do**
  if *there is a* $\mathbf{c}_1$ *closer to the* $\mathbf{c}_2$ *than* $G_{threshold}$ **then**
    add $\mathbf{c}_2$ to the list *candidates*
  end
end
Randomly shuffle the list *candidates*;
Initialize *best_correspondences* as None;
Initialize *best_criterion_value* as ∞ or $-\infty$ based on the criterion;
for *each* $\mathbf{c}_2$ *in candidates or until maximum number of iteration is reached* **do**
  for *each* $\mathbf{c}_1$ *closer to the* $\mathbf{c}_2$ *than* $G_{threshold}$ **do**
    Compute the translation $\mathbf{t} \leftarrow \mathbf{c}_2 - \mathbf{c}_1$;
    Translate all $\mathcal{C}_2$ using this *translation*;
    $\mathcal{C}_2' \leftarrow \{\mathbf{c}_2 + \mathbf{t} | \mathbf{c}_2 \in \mathcal{C}_2\}$;
    Initialize *correspondences* as an empty list;
    for *each* $\mathbf{c}_2' \in \mathcal{C}_2'$ **do**
      if *there is a* $\mathbf{c}_1$ *closer to* $\mathbf{c}_2'$ *than* $L_{threshold}$ **then**
        add the original $\mathbf{c}_2$ with the closest $\mathbf{c}_1$ to the list *correspondences*
      end
    end
    Calculate value V of criterion given the list *correspondences*;
    if *value* V *is better than the actual best_criterion_value* **then**
      Update *best_correspondences* to *correspondences*;
      Update *best_criterion_value* to the V;
    end
  end
end

Algorithm 1: Establishing correspondences between pair of images and associated cell detections

Image Similarity Criterion for Matching Image to Map For matching an image to a created map, we calculate the cosine distances for the established correspondences between the cells in the map and in the image. To enhance the matching process, we further designed three different weights to weigh the cosine distances of each correspondence. The threshold for potentially rejecting faulty matches is once again set to 0.5, as described in the previous paragraph.

First, for each correspondence, we calculate a combined weight w as the sum of three individual weights: $w_{\text{detection reliability}}$, $w_{\text{landmark dissimilarity}}$, and $w_{\text{historical consistency}}$. The weights are then normalized over all correspondences by dividing each weight w by the sum of all weights so they sum up to one. After normalization, the criterion function is calculated as a weighted sum of the cosine distances between correspondences and is again minimized.

The first weight $w_{\text{detection reliability}}$ represents the reliability of each detection. To determine this, for each detection, we identify map cells in its relaxed neighborhood, defined by the radius

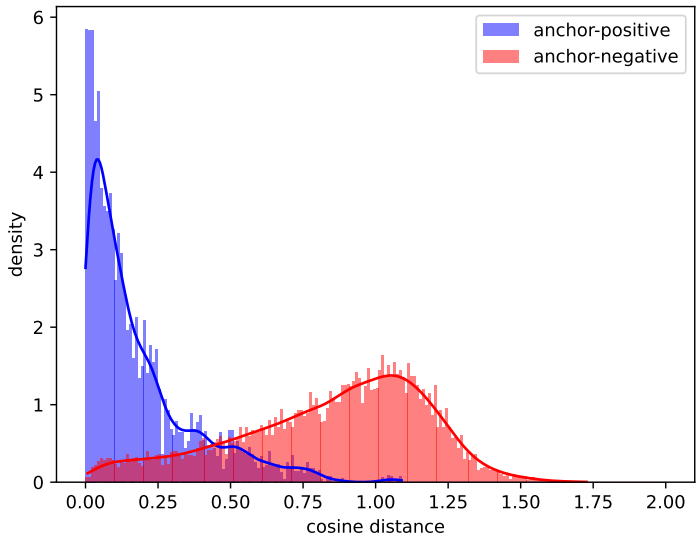


Figure 10: The distribution of cosine distances between anchor and positive samples and anchor and negative samples was calculated using the validation data used in the image similarity network training process. The distributions were estimated using kernel density estimation.

of $G_{\text{threshold}} + 250$ pixels. We then calculate cosine distances between the actual detection and map cells in its neighborhood and identify the first and second smallest cosine distances d_1 and d_2, respectively. If these distances are similar, it means that there may be multiple good matches for the detection in its neighborhood, and hence, the weight should be smaller. The weight is then defined as follows:

$$w_{\text{detection reliability}} = 1 - \frac{d_1}{d_2} \tag{6}$$

The second weight $w_{\text{landmark dissimilarity}}$ represents the dissimilarity of map cells (landmarks) with their neighbors. Similarly, for each landmark, we identify map cells in its relaxed neighborhood and calculate cosine distances c_i between the landmark and other map cells in the neighborhood. We then adjust these distances such that any cosine distance greater than one is set to one and take the mean of these adjusted distances. If the mean is close to one, indicating that the landmark could be unique in its neighborhood, the weight should be larger. The weight is calculated as follows:

$$w_{\text{landmark dissimilarity}} = 1 - \frac{1}{N} \sum_{i \in \text{neighborhood}} \min(c_i, 1) \tag{7}$$

The third weight $w_{\text{historical consistency}}$ represents the consistency of cosine distances between the matches. For each map cell, we calculate the mean and variance of the cosine distances between previously matched detections and the map cell. To quantify the consistency, we compute a z-score based on the actual cosine distance and the associated mean and variance with the map cell. From this z-score, we calculate the corresponding p-value using a survival function of normal distribution. This p-value indicates the likelihood of observing the actual

cosine distance, or larger, given the mean and variance of the historical cosine distances for the map cell. For example, if there was a low variance in distances between matched detections, the p-value will be small if it deviates a lot from the average one.

5.2.3 Making Initial Map

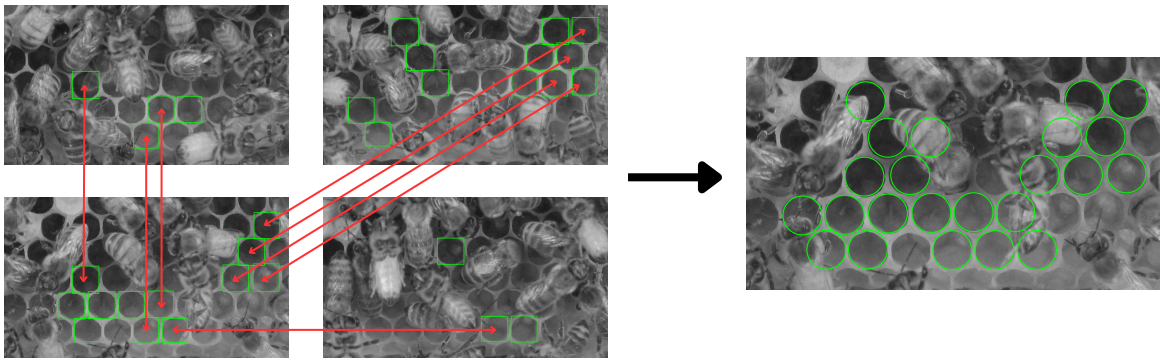
As previously mentioned, the input to the process of creating an initial map is a sequence of scans of honeybee combs. The objective of this process is to establish correspondences between cell observations in the comb scans and subsequently estimate the positions, radii, visual descriptors, and states (see Sec. 5.4 for more details) of identified unique cells. This could be approached, for instance, by simply calculating the metric positions of all detected cells and subsequent clustering of the detections. However, in the case of imprecise odometry, such an approach may fail, and thus, we designed a more robust technique for creating the initial correspondences.

Initial Map Per Tile The process begins by establishing correspondences between all comb scans for each individual image tile, as depicted in Fig. 11. Initially, we utilize the image registration method described in the previous subsection (see Sec. 5.2.2), employing a criterion for matching image pairs. These correspondences provide the basis for estimating translations between the image pairs. Subsequently, we anchor the position of the first image tile in the sequence to its metric position and apply a global optimization technique, introduced in Sec. 5.2.1, to refine the position of other image tiles. Once the images are aligned on top of each other, we recalculate the metric positions of the detections and group all detections, whose mutual distance is at most $L_{\text{threshold}} = 50$ pixels, into clusters, thereby forming an initial map for each image tile. Additionally, for each cell in the tile map, we compute a visual descriptor as the mean of all associated visual descriptors, thus representing the map for image tiles as a set of cells with corresponding visual descriptors. We maintain a list of all detections associated with the cells in the tile map.

To mitigate potential false positive detections of cells, we employ a Non-Maximum Suppression (NMS) with an Intersection over Union (IoU) threshold of 0.3. In cases of overlap, we retain cells with a higher number of detections.

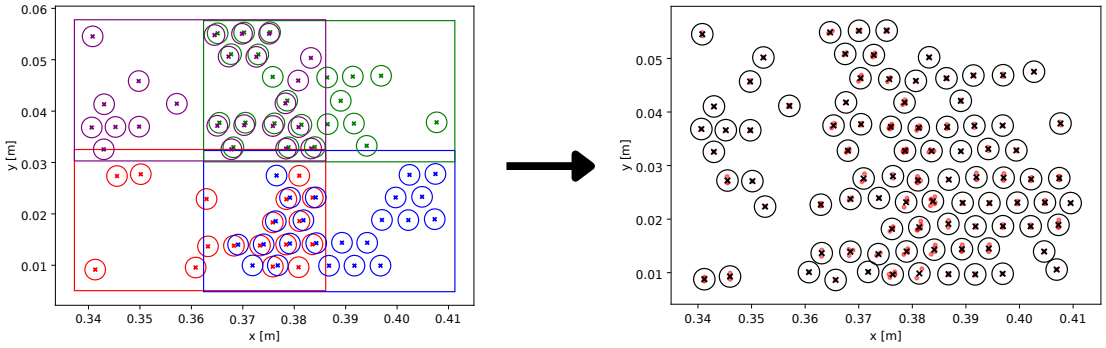
Combining Initial Per Tile Maps To generate the comprehensive initial map of the entire comb, we combine the initial maps created for individual tiles, as illustrated in Fig. 12, which showcases this process using only four image tiles for simplicity. Employing once more the registration method introduced in Sec. 5.2.2 and leveraging the criterion for matching image pairs, we establish correspondences among neighboring image tiles. This procedure directly yields a list of unique cells within the map. Subsequently, for each unique cell within the list, we compute all pertinent attributes before incorporating it into the final initial map. Additionally, we employ the NMS, as mentioned before, to filter out possible overlapping cells in the map. The resulting representation of the initial map for four image tiles is visualized in Fig. 12.

Each cell c_i in the map is characterized by five attributes: $(\mathbf{p}, r, \mathbf{d}, \mathbf{s}, t)_i$. The metric position $\mathbf{p}_i = (x, y)_i$ is computed as the mean of all detections associated with the cell, where each detection with its metric position is treated as an independent measurement. Similarly, the



a) pairwise correspondences b) initial map for the tile

Figure 11: Creating an initial map for one specific tile using images collected across four honeybee comb scans. In **a)** are the correspondences between all pairs of images. In **b)** is the final initial map, visualized on top of one of the images.



a) initial maps for individual tiles b) final initial map

Figure 12: The visualization illustrates the process of creating the initial map from per-tile initial maps. For simplicity, only four image tiles are considered. In **a)**, the initial maps of individual tiles are visualized. In **b)**, the final initial map (black) is presented alongside all detections used for estimating the final positions of the cells in the map (red).

radius r_i is determined as the mean over all detections. The 64-dimensional visual descriptor $\mathbf{d}_i$ is calculated as the mean over all visual descriptors of the cell. The state $\mathbf{s}_i$ is computed using a temporal filter, which will be introduced in Sec. 5.4. Additionally, the timestamp t_i indicates when the cell was last observed.

5.2.4 Updating Map

After the initial map is created, subsequent honeybee scans are utilized to update the map. For this purpose, we employ the image registration method presented in Sec. 5.2.2, using the criterion for matching images to the map. The cell detection-based image registration of the image and map is visualized in Fig. 13. If the image is not rejected due to a potentially faulty match, the correspondences are directly used to update the cells in the map. Detections

without correspondences are added to the map as new cells.

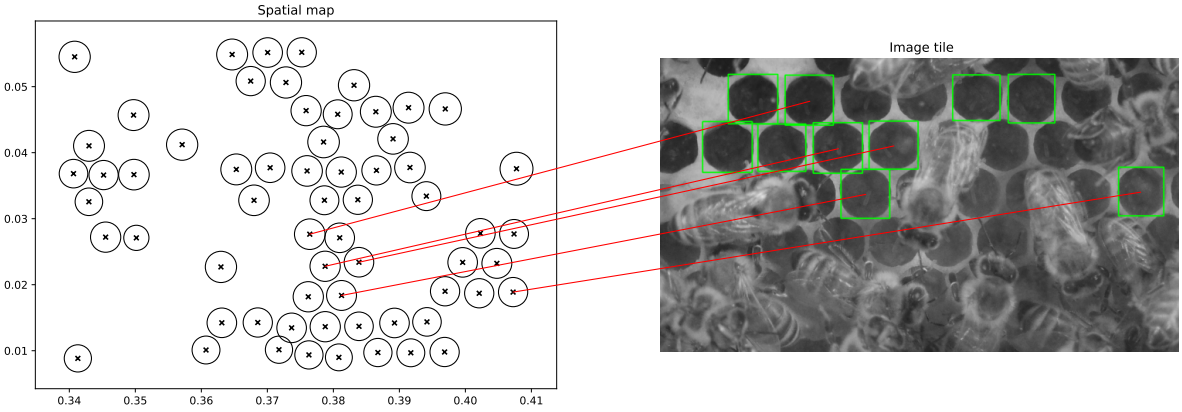


Figure 13: Matching an image tile to the spatial map

In cases of false positive detection or a faulty match that was not rejected, it may occur that cells in the map overlap. To ensure spatial consistency, we implement a Non-Maximum Suppression (NMS) with an Intersection over Union (IoU) threshold of 0.3 and retain the cells with the highest number of detections.

The attributes of each map cell c_i, namely its position $\mathbf{p}_i$ and radius r_i, are updated through incremental calculation of the average. To update the visual descriptor $\mathbf{d}_i$, we employ a form of exponential moving average (EMA), wherein the previous descriptor is weighted by 0.9 and the current descriptor by 0.1. We adopt this approach to enhance robustness, particularly in cases where a faulty match is not rejected. The state of the cell is updated through a temporal filter, which is described in Sec. 5.4.

5.3 Honeybee Cell Classification

When creating the semantic map on top of the metric one, we need a way of extracting the semantic information, which is, in our case, the content of the cells. Several works have addressed the classification of honeybee cells based on their content (see Sec. 4.3.2), which usually differ in the number of classes. We adopt the cell categorization as outlined in [88], with seven classes: egg, larva, capped brood, pollen, nectar, honey, and empty/other. In [88], the authors employed and compared multiple Convolutional Neural Network (CNN) architectures for cell classification, using networks pre-trained on the ImageNet dataset and fine-tuned them using a manually curated dataset comprising over 71k cell images. Following the state-of-the-art in image classification, we tackle cell classification using deep CNNs as well.

The input to the classification are individual image tiles and detections of the cells, the individual cells are then cropped from the image tiles, resized to 128 px × 128 px and fed to the network for classification. Specifically, we employ a rather small XResNet-18 architecture, illustrated in Fig. 14 with about 2.8 million trainable parameters. This architecture is a variant of ResNet-18 [97], incorporating several additional enhancements detailed in [101]. For training the network, we used the categorical cross-entropy loss. Given the relatively

small size of our dataset (see Sec. 6), we initially pre-trained the network on the dataset from [88] and we fine-tuned the network using our specific data.

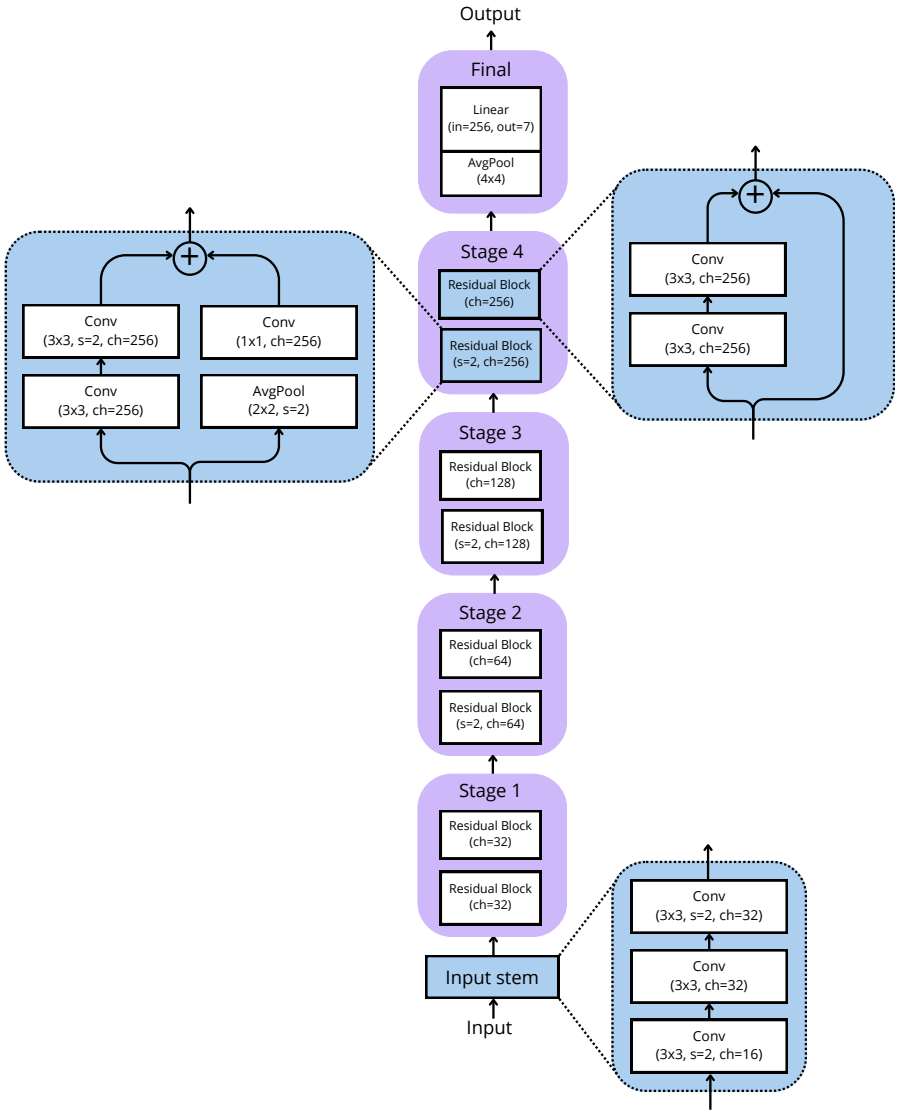


Figure 14: XResNet-18 architecture for cell classification

5.4 Temporal Filtering

To improve our understanding of the cell states and ensure temporal consistency, we can utilize the knowledge of possible biological processes occurring in honeybee colonies, such as the development stages of bees, which were described in Sec. 4.1. The goal is to estimate the posterior distribution of a state $S_t \sim p(S_t \mid z_{1:t})$ at time t for each cell c in the map, given all past observations of the cell's class $z_{1:t}$. Note that we understand cell state to be defined by the content, whether it's used for storage (pollen, nectar) or young bee development, not by the classes of observations discussed above in Sec. 5.3, so $support(S) \neq support(Z)$.

The Bayes filter simplifies the filtering process by the Markov assumption, which states that the current state encapsulates all relevant information for predicting the future state. In mathematical terms, this can be expressed by the following equations: $p(Z_t \mid S_t, z_{1:t-1}) = p(Z_t \mid S_t)$ and $p(S_t \mid S_{t-1}, z_{1:t-1}) = p(S_t \mid S_{t-1})$. This assumption allows for the recursive calculation of the posterior distribution $p(S_t \mid z_{1:t})$ at time t from the corresponding posterior $p(S_{t-1} \mid z_{1:t-1})$ at time $t-1$ and the most recent measurement z_t. We adopt the formulation of the Bayes filter from [58]. The Bayes filter algorithm comprises two steps: prediction (eq. 8) and measurement update (eq. 9), where $P(S_t \mid S_{t-1})$ is referred to as a transition model, $p(Z_t \mid S_t)$ is a sensor model:

$$p(S_t \mid z_{1:t-1}) = \sum_{s_{t-1}} p(S_t \mid S_{t-1} = s_{t-1}) \cdot p(S_{t-1} = s_{t-1} \mid z_{1:t-1}) \qquad \text{prediction} \quad (8)$$

$$p(S_t \mid z_{1:t}) \propto p(z_t \mid S_t) \cdot p(S_t \mid z_{1:t-1}) \qquad \text{measurement update} \quad (9)$$

5.4.1 States

As mentioned before, the cell state should reflect the content of the cell. Following the categorization established in Sec. 5.3, we distinguish seven categories of cells: egg, larva, capped brood, pollen, nectar, honey, and empty/other. The duration of each developmental stage of a young bee is in biological studies typically expressed in days, as introduced in Sec. 4.1, which is why we cannot simply use the observable classes. If we did, then the likelihood of the egg state transitioning to the larva state would be dependent on the age of the egg, which would violate the Markov assumption. Moreover, the bee development process is practically deterministic, so averaging the transition into stationary probabilities would create uncertainty, which would not be found in reality. Therefore, we define states separately for each day within each developmental stage. This inherently gives us the temporal resolution of the filter to be in days. We also need to consider that durations of larva and brood stages differ between female worker bees and male drones, so we introduce two hidden alternative paths for them, which can be distinguished from observations only by different lengths. This results in a total of 41 states, which are visualized in Fig. 15. The initial state is a uniform distribution over the observable cell classes.

5.4.2 Transition Model

The transitions between the states are rooted in the biological model of honeybee colonies. Beyond identifying which states are connected, it is crucial to set the probabilities of these transitions. However, tuning the parameters of the transition model is similar to answering the question "To bee or not to bee?", as there is a lack of sources addressing it. Additionally, these transition probabilities may be specific to the hive and dependent on the actual weather conditions and supplies of pollen, nectar and honey. Therefore, we set the probabilities based on our best knowledge, while acknowledging that the true probabilities may differ. Probabilities that we don't mention specifically, are calculated as a complement to the specified ones. All the possible transitions between the states are illustrated in Fig. 15.

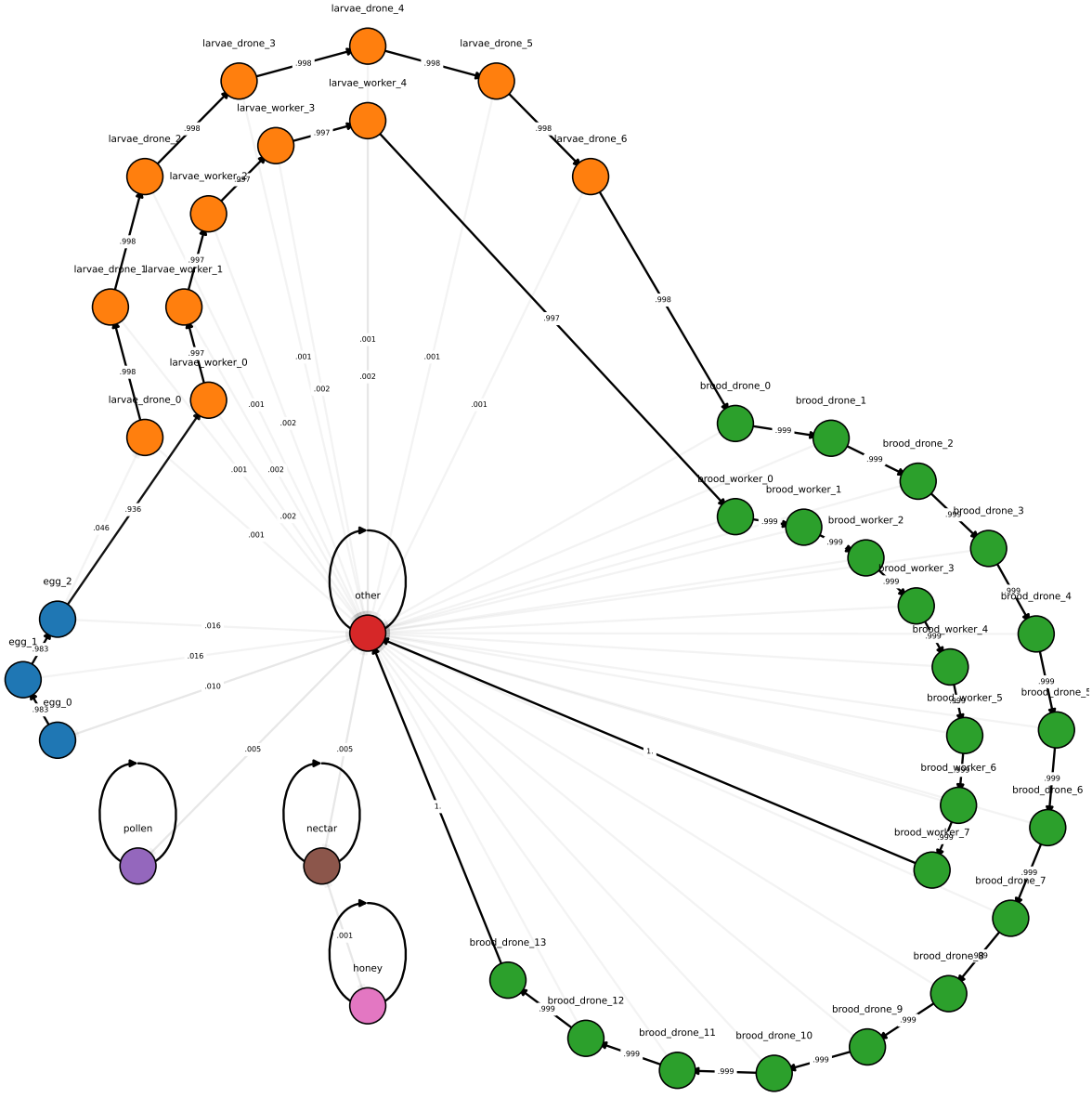


Figure 15: Visualization of possible states of the cells, together with the transition model between the states.

Brood Cells The last state of the egg development stage has a defined transition into the first state of the larva stages of worker bees and drones. The probabilities of the transitions are in the proportion of the number of worker bees and drones in the hive, which we set to a sensible value of $20 : 1$. Similarly, the last states of the larva stage for both worker bees and drones can transition into the first states of the corresponding capped brood stage. The states within each development stage are connected sequentially. For each state in each development stage, there is a possibility of brood cannibalism, thus, each such state is connected to the state other/empty. Additionally, the connection between the empty/other state and the state egg_1 is bidirectional, as there is a probability $p(\text{egg emergence}) = 0.05$ that a queen will lay

an egg in an empty cell.

Problem of Canibalism Bees exhibit a behavior where they remove their offspring, which is referred to as "brood cannibalism", but since it is hard to observe this behavior, there is not much literature about the rates of occurrence of this behavior, let alone how likely it is to happen at different stages and different days of the brood development. We set the probabilities that a brood cannibalism occur as $p(\text{egg canibalism}) = 0.05$, $p(\text{larva canibalism}) = 0.01$ and $p(\text{capped brood canibalism}) = 0.001$. However, when setting the probabilities of cannibalism for individual states that also contain the age information, we need to address that the probability of the removal at a certain age is conditioned on brood surviving to that age. For instance, the probability that egg_2 is removed is conditioned on the probability that egg_1 was not removed. We will now briefly describe the calculation of the probabilities of brood cannibalism for each day of the *class* development stage, where *class* is either egg, larva, or capped brood. The probability $p(\text{class canibalism})$ for state of *class* that forms N days long sequence of states can be expressed as:

$$p(\text{class canibalism}) = \sum_{i=1}^{\text{N}} P(\text{class canibalism} \mid \text{class}_i) \tag{10}$$

$$= \sum_{i=1}^{\text{N}} P(\text{class canibalism} \mid \text{class}_i) \prod_{j<i} (1 - P(\text{class canibalism} \mid \text{class}_j)) \tag{11}$$

We adopt a simplifying assumption that the probability of brood cannibalism is the same for each day of the development stage, i.e., it is equally likely on day 1 as day 2. We expect our system in the future to learn the properties of this process in greater detail from the observations. Using this assumption, we further simplify the equation. We can then numerically solve for it, which allows us to set the probabilities $p(\text{class cannibalism} \mid \text{class}_{day})$ accordingly.

$$p(\text{class cannibalism} \mid \text{class}_i) = c \quad \forall i \in \{1, \ldots, N\} \tag{12}$$

$$p(\text{class canibalism}) = \sum_{i=1}^{N} c \cdot (1 - c)^{i-1} \tag{13}$$

Storage Cells For storage cells, the pollen state also has a bidirectional connection with the empty/other state, as bees may start storing pollen in an empty cell with probability $p(\text{pollen emergence}) = 0.02$ and can also remove it with $p(\text{pollen depletion}) = 0.005$. Similarly, the nectar state has a bidirectional connection with the empty/other state for the same reasons, with $p(\text{nectar emergence}) = 0.02$ and $p(\text{nectar depletion}) = 0.005$. The honey state is only connected to the nectar state, reflecting the process of honey creation. The connection is bidirectional with $p(\text{nectar capping}) = 0.005$ and $p(\text{honey uncapping}) = 0.003$. The states of empty/other, pollen, nectar, and honey have a non-zero probability of remaining in their current state.

5.4.3 Sensor Model

We utilize the neural network for the classification of honeybee cells based on their content, presented in 5.3, as our sensor for state observations. For establishing the sensor model, we utilize the confusion matrix of the classification neural network calculated on the validation data, visualized in Fig. 16. The sensor model $p(z_t \mid S_t)$ is calculated by normalizing each row of the confusion matrix to sum up to one. Moreover, as a rather small dataset was used for the calculation of the confusion matrix (see Sec. 6), we add a 10% of uniformly distributed noise to the sensor model.

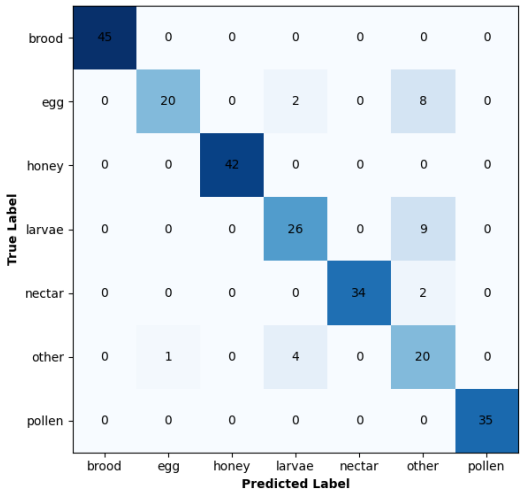


Figure 16: Confusion matrix of the classification neural network on validation data

5.4.4 Practical Details

Sensor Model When incorporating the class predictions from the classification neural network to the temporal filter as measurements, we can use the categorical distribution calculated as softmax of the network's output or a one-hot encoded vector of its argument of maxima. Based on experiments performed on the validation part of the classification dataset (see Sec. 6.2), we use the one-hot encoded vectors of an argument of maxima as measurements.

Additionally, as it will be discussed in Sec. 6.2, the dataset used for training the classification network is not entirely representative of all the cells that can be encountered in the honeybee comb. We estimated that we could not distinguish, and thus annotate, 20% of all the cells in the honeybee comb, which were consequently omitted in the dataset used for training the network. To address this, when deploying the temporal filter, we add an additional 20% of uniformly distributed sensor noise to the sensor model. In Sec. 6.2, we carry out an experiment that shows that this approach is reasonable.

Observations with Partial Time The temporal filter is designed for a 1-day frequency of observations. However, in reality, the frequency of observations usually differs as the scans are not performed at the same time every day and in the same amount, not to mention that the cells may not be observable in the scans if they are occluded by bees. Therefore, if there is

a smaller time interval than one day between the observation and the last time the prediction was performed for the cell, we only perform the measurement update step of the temporal filter. Similarly, if the time interval is greater than one day, we perform multiple prediction steps according to the number of whole days in the interval.

6 Testing of Pipeline Components

Evaluating the final map of the honeybee comb is a challenging task, as creating the ground truth for such a map is intractable with current data. Therefore, we evaluate all the components of the mapping pipeline, which were discussed in Sec. 5, separately, with the assumption that combining those components won't significantly affect their performance. Parts of subsections Sec. 6.1 and Sec. 6.4 are about to appear in [102].

6.1 Honeybee Cell Detection

In this section, we evaluate the proposed cell detection methods: Circle Hough Transform (CHT), Faster R-CNN, and YOLOv5 (see Sec. 5.1), which are the basis of our object-based mapping. First, we will introduce the dataset annotated to train the cell detection models and determine the optimal parameters of the CHT. Then, we summarize the results of these methods.

6.1.1 Dataset

The dataset was annotated in a semi-automatic manner and consisted of images with bounding boxes specified for each cell. We used the unzoomed images with a resolution of 67 μm per pixel. To ensure the diversity of the dataset, the images were randomly selected from all the collected scans over the span of 30 days (see Section 3.1.4).

Initially, we used the Segment Anything Model (SAM) [103] with the ViT-H model for image instance segmentation. Most parameters of SAM were left at their default settings, with the following exceptions: points per side were set to 35, the minimum mask area was set to 100, the crop Non-Maximum Suppression (NMS) threshold was set to 0.2, and the box NMS threshold was set to 0.2.

Subsequently, we refined the resulting masks by implementing a simple filter based on the area and circularity of the individual masks, and finally, we manually annotated the pre-filtered segmentation masks. As mentioned in section 5.1, we distinguish two categories of cells: fully visible and partially occluded. Using this annotation tool, we annotated a set of 260 honeybee comb images, which were split into training (200 images), validation (30 images), and testing (30 images) parts.

6.1.2 Results

The cell detection methods are evaluated using the testing part of the object detection dataset. We apply class-agnostic Non-Maximum Suppression (NMS) with an Intersection over Union (IoU) threshold of 0.3 to the output of object detectors (where applicable) to ensure that each cell is classified as either fully visible or partially occluded. The circles detected by the Circle Hough Transform (CHT) are all considered to be of class *fully visible cell*.

For each method and class, we calculate the Average Precision (AP) metric at an IoU threshold of 0.5, as well as the average AP over IoU thresholds in the interval $[0.5, 0.95]$ with a step size of 0.05. Additionally, we report precision (P) and recall (R) metrics at an IoU

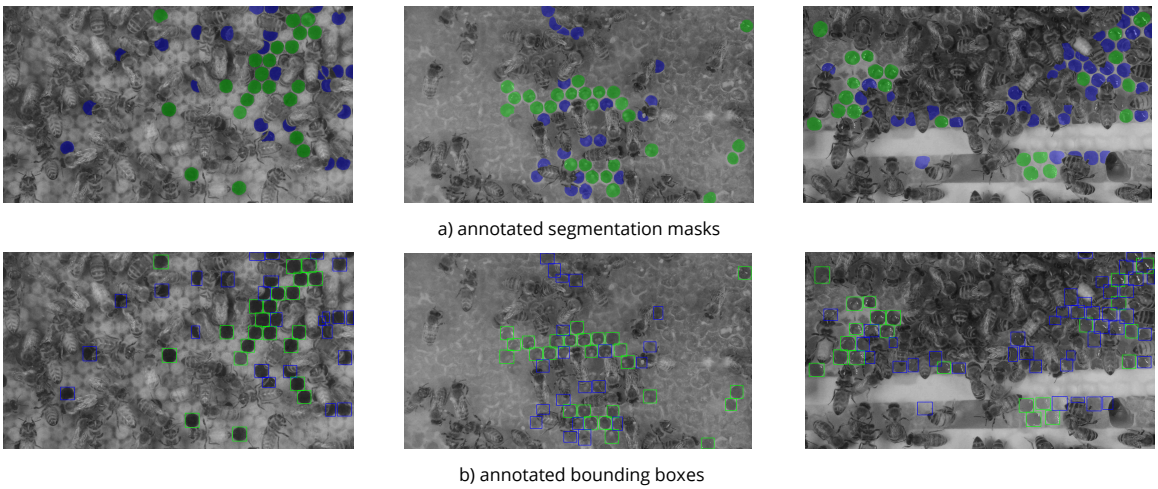


Figure 17: Samples from the testing part of the object detection dataset. Fully visible cells are highlighted in green, while partially occluded cells are highlighted in blue. **a)** shows the masks of individual cells, and **b)** shows the resulting bounding boxes.

threshold of 0.5 and a confidence threshold of 0.5. The results for the class *fully visible cell* are summarized in Table 3 and for the *partially occluded cell* in Table 4.

Table 3: Fully visible uncapped cell detection results

Method	**AP** [%]	**AP-50** [%]	**P** [%]	**R** [%]
CHT (Bilateral filter, CLAHE)	9.5	13.0	11.1	95.1
YOLOv5s6	86.6	94.0	85.1	93.9
Faster R-CNN (ResNet-18)	85.4	95.0	88.6	93.6
Faster R-CNN (ResNet-50)	**90.9**	**95.3**	**92.2**	**92.4**

Table 4: Partially occluded uncapped cell detection results

Method	**AP** [%]	**AP-50** [%]	**P** [%]	**R** [%]
CHT (Bilateral filter, CLAHE)	-	-	-	-
YOLOv5s6	64.2	83.3	87.6	68.8
Faster R-CNN (ResNet-18)	64.7	87.5	91.1	67.1
Faster R-CNN (ResNet-50)	**79.2**	**92.0**	**87.4**	**84.9**

It can be seen that although the Circle Hough Transform (CHT) has a high recall value, it is not suitable for cell detection in natural living colonies without additional filtering of the detections, as it produces a large number of false positives. The best results were achieved with Faster R-CNN using the ResNet-50 backbone, which outperformed all other methods. Faster R-CNN with the ResNet-18 backbone and YOLOv6 achieved similar results, but both struggled with the detection of partially occluded cells.

For the mapping of the comb, we ultimately decided to use only the zoomed data, as it captures the comb in more detail. Despite evaluating the detection models on unzoomed images, we visually inspected and confirmed that they perform well on downscaled zoomed images as well. Thus, it is acceptable to use these methods without further adjustments.

6.2 Honeybee Cell Classification

To acquire semantic information about the cells based on their content, we trained a classification neural network (see Sec. 5.3) and combined it with a temporal filter (see Sec. 5.4). This section first describes the dataset that was annotated for the classification. Subsequently, we evaluate the performance of the network and the effect of its combination with the temporal filter.

6.2.1 Dataset

To train the classification model of cells' content, we first used the dataset from [88] to pretrain the network. Additionally, we created a dataset for our specific setup comprising images of individual cells and their categorization based on the content: egg, larvae, brood, pollen, nectar, honey, and empty/other. As some of the contents, particularly honeybee eggs, are hardly visible in the unzoomed images, the dataset is created from the zoomed comb scans from both sides of the hive.

To make the annotation process more efficient, we first gathered sequences of observations for each cell. Moreover, we didn't have the possibility to let an experienced beekeeper annotate the data, and in many cases, the class of the cells was not easily distinguishable. Hence, creating sequences of cell observations also allows us to interpolate the class of the cell between observations in the sequence that we were sure about. For gathering the sequences, we used the method for establishing correspondences between images and map, which was described in Sec. 5.2.2, with the criterion of maximum number of correspondences. To ensure that the correspondences are correctly created, we supervised the method, and in case of an incorrect match, we manually corrected it.

We acknowledge that the distribution of cell types may not be uniform in the comb, however, the distribution usually differs based on weather conditions, season or even the specific hive that we observe. We wanted to prevent bias in the classification, as it may worsen the performance on other observation hives and throughout the seasons, thus, sensible choice was to create the dataset in a way that all classes are approximately similarly distributed in it. With this in mind, we selected the particular sequences for the annotation and divided them into the training, validation, and testing parts accordingly.

In the selected subset of sequences, we manually annotated all observations of the cells into the seven categories based on their content. We manually added the observations in which the cells were capped for completeness, as the object detectors don't detect these. In total, we annotated 146 sequences of cell observations, consisting of a total of 1,946 images of cells. The sequences were further split into training (73 sequences with a total of 988 cell images), validation (16 sequences with 248 cell images), and testing (57 sequences with 710 cell images) parts. Fig. 18 shows a few samples from the dataset.

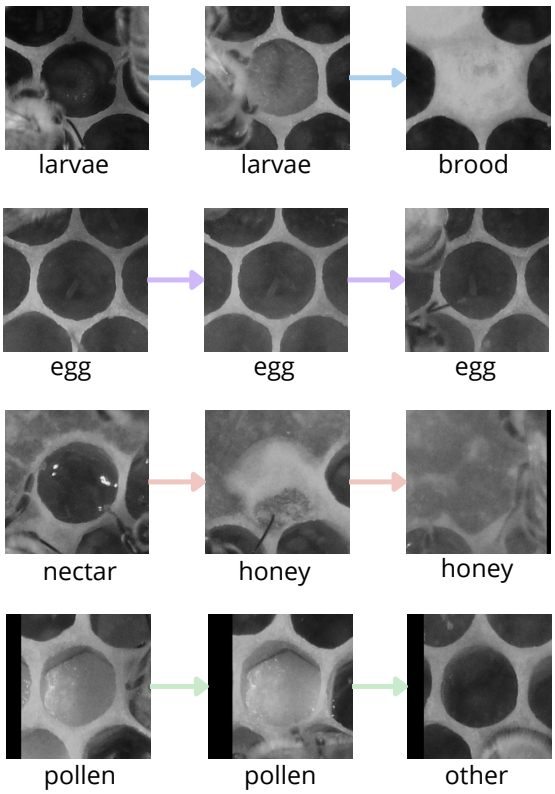


Figure 18: Samples from the dataset for cell classification. In each row are different observations of the same cell, ordered in time.

Problem with Annotation When annotating the cells, there were cases in which we didn't annotate the cell images, as we couldn't distinguish them with enough confidence. Consequently, these cells are not represented in the datasets we created, and thus, we can't expect the neural network to perform reliably on them, as they were not included in the training process. We randomly sampled 400 images of cells across all collected scans and estimated the proportion of data that we could not annotate to be 18.75%. We rounded this estimate to the value of 20% and used it when deploying the temporal filter by adding an additional 20% of uniformly distributed noise to the sensor model (described in Sec. 5.4).

6.2.2 Results

Due to the problem mentioned with the annotation, we carried out two experiments. First, we evaluated the model and its combination with the temporal filter on the testing part of the dataset, which comprises annotations that we were confident about. In the second experiment, we evaluate the effect of deploying the temporal filter on data that also contain, for us, undistinguishable cell images not included in the dataset.

Results on Testing Dataset The multi-class classification is evaluated on the testing dataset using standard image classification metrics precision and recall, which are calculated

for each class separately and then averaged. These metrics are calculated for both the trained XResNet-18 classification neural network and its combination with the temporal filter (TF). In this experiment, we do not employ the additional sensor noise discussed in the previous subsection, as the testing dataset contains only annotations about which we were confident.

The results are presented in Tab. 5. It can be seen that the temporal filter (TF) slightly improved the classification results of the neural network on the testing dataset. However, it should be noted that the primary advantage of employing the temporal filter is not improving the classification metrics on the testing dataset but rather enhancing our understanding of the temporal changes that the cells undergo, as the temporal filter provides a detailed prediction of the development stage of the brood cells.

Table 5: Cell classification results

Method	**Precision** [%]	**Recall** [%]
XResNet-18	85.3	85.0
XResNet-18 + TF	**86.1**	**85.8**

Results with Simulated Expected Noise We expect the network to make mistakes on the type of images we could not annotate and, thus, on which the network wasn't trained. Here, we perform an experiment in which we evaluate the performance of the methods when they encounter these undistinguishable cell images. For that, we introduce artificial errors into the network predictions and test the effect of employing a temporal filter on the metrics of the network. Varying the error rate p_{noise} and setting the corresponding parameter of the temporal filter accordingly, we produce Fig. 19.

The results show that the temporal filter is capable of handling noisy measurements while maintaining reasonable precision and recall. As discussed previously, there was about 20% data that was not represented in the annotated dataset because even a human annotator could not label them. For this particular value the classification metrics remain around 80% when using the temporal filter, the uniform noise itself results in drop to 70%. At other values of noise, the temporal filter can sustain the performance of up to 15% over the non-filtered network.

6.3 Cell Image Similarity

To improve the robustness of matching image tile pairs and images to map, we trained a Siamese network that calculates the visual descriptors of individual cells from images of them with a small neighborhood (see Sec. 5.2.2). This section first introduces the process of creating the dataset to train the network. Afterward, we show the results of the image similarity network evaluated on validation data.

6.3.1 Dataset

For training the network for image similarity of cells and their neighborhood, we created a dataset of triplets (A, P, N), where A is an anchor image, P is a positive sample, which is

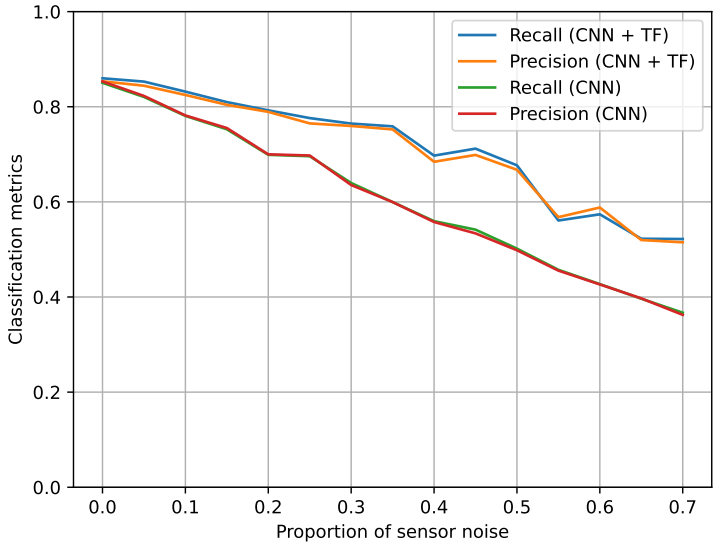


Figure 19: Classification results with simulated expected noise in data for proportions of sensor noise in the interval $[0, 0.7]$ with a step of 0.05.

an image of the same cell as A but observed at a different time and N is a negative sample, which is an image of different cell.

To handle that cell change in time, the image pairs were sampled randomly across all the collected scans. We used both the unzoomed and zoomed images. First, we sampled a random image tile from all scans, and then we sampled the same image tile or one of its neighbors from scans that were collected a maximum of two days ago or after. Subsequently, we again used the method for establishing correspondences between detections in a pair of images with the criterion of maximum number of correspondences introduced in Sec. 5.2.2. The established correspondences were manually checked, and if the match was incorrect, we did not add it to the dataset. Consequently, the dataset only has samples that were easily matchable for the method. However, as the triplets are sampled across whole images, this does not pose a problem. This way, we created a dataset of image pairs with correspondences between cell detections in the images. In total, we created 305 pairs of unzoomed images and 463 pairs of zoomed images, with the correspondences between them, which were further split into training (227 unzoomed image pairs, 383 zoomed image pairs) and validation (78 unzoomed image pairs, 80 zoomed image pairs) parts.

The triplets were then generated by sampling them from the pairs of images. For each image pair (I_1, I_2) and each corresponding detection (d_1, d_2) between the images in the pair, we first set the d_1 as the anchor and d_2 as the positive example and sampled N_{neg} negative examples from the detections in image I_2. Similarly, we then set the d_2 as the anchor and d_1 as the positive example and sampled N_{neg} negative examples from the image I_1; to mitigate overfitting, $N_{neg} = 4$ for the training dataset, for the validation dataset, $N_{neg} = 7$. This way, we obtained 19,631 triplets in the training dataset and 8,682 triplets of images in the validation part of the dataset. A sample from the dataset is shown in Sec. 5.2.2 in Fig. 9.

It should be noted that we did not go the extra mile to create a testing dataset for the image similarity on purpose. This choice is based on the fact that we do not care about the

specific precision of the neural network on a dataset of triplets. The primary use case of the network is for matching detections. Thus, we test the performance of the image similarity network on this specific task instead (see Sec. 6.5 and Sec. 6.6). However, to show that the network works as desired, we provide here the results of the evaluation on the validation data.

6.3.2 Results

We evaluate the neural network for image similarity using validation data. Since the dataset contains ground truth samples only for the positive class, we use accuracy as the metric. The accuracy indicates the proportion of cases in which $d(A, P) < d(A, N)$, where d is the cosine distance function, A is the anchor image, P is a positive sample, and N is a negative sample. The model's accuracy on validation data is $\mathbf{95.2}\%$.

6.4 Image-based Registration of Pair of Image Tiles

In the beginning, we experimented with traditional image-based methods for registration of image tiles within one scan, discussed in Sec. 5.2.1, to improve the result of image stitching of the honeybee comb. In the end, these methods were not utilized in our work. However, we report them for completeness and to show their limitation when used in a dynamic and buzzing environment with highly repetitive patterns. Furthermore, we will compare these methods with cell detection-based methods in Sec. 6.5.

6.4.1 Dataset

To evaluate the accuracy of both the odometry within one comb scan and the proposed image-based registration methods, we created a dataset of neighboring image tile pairs along with the corresponding translation information between them. The odometry information served as prior information for determining these translations, which were then further manually refined. For creating the dataset, we used the unzoomed images with a resolution of 67 μm per pixel.

As mentioned, there were some technical problems during the data collection, which resulted in imprecise odometry. Hence, we decided to evaluate the methods for both the system that produced imprecise odometry as well as for the system that didn't have this problem. We created two datasets, the first dataset *IS1*, consists of 82 pairs of image tiles with imprecise odometry information. The second dataset, *IS2*, comprising 176 pairs of neighboring image tiles, has more precise odometry information. A few samples from the *IS1* dataset are shown in Fig. 20.

6.4.2 Results

The image-based registration methods were evaluated using the dataset presented in the previous section. For each method, we calculate the mean translation error with standard deviation for both the horizontal and vertical axes. The parameter O_{dev}, which limits the search space of the methods, was set to the value of 20 pixels, which is about 1.3 mm. The results for the *IS1* dataset are summarized in Tab. 6.

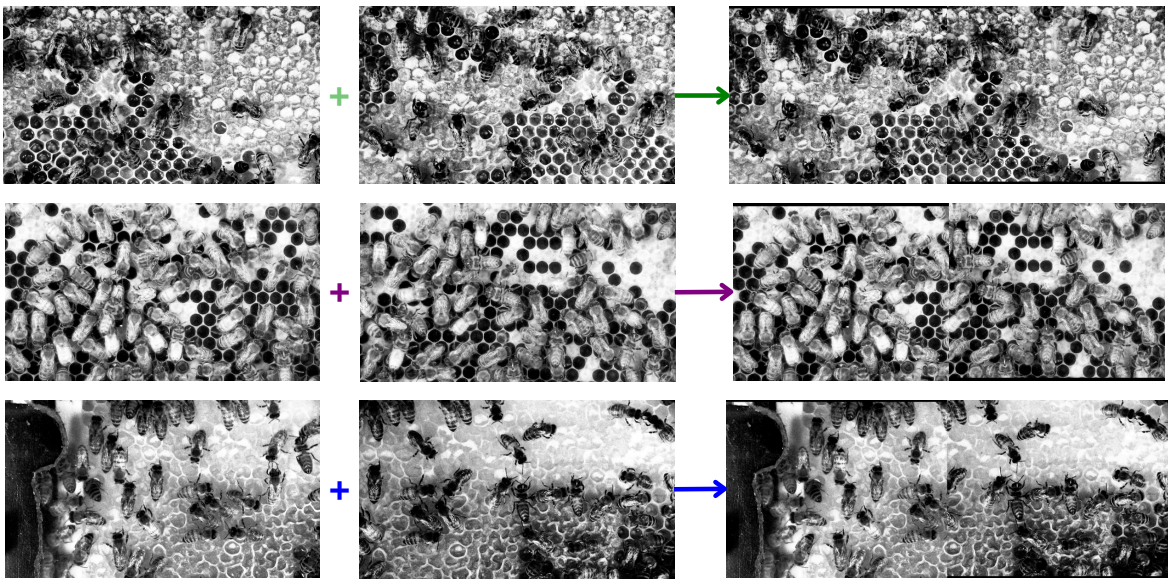


Figure 20: Samples from the *IS1* dataset for image registration. All presented images underwent normalization and histogram equalization. The individual images are on the left, and the visualized translations in the form of stitched images are on the right.

It can be seen that the odometry in the *IS1* dataset is relatively imprecise. The best result was achieved with the correlation-based approach. Surprisingly, its performance was better without employing global optimization. The feature-based approach performed worse, but the global optimization slightly improved its accuracy. Both methods, however, managed to decrease the odometry error.

Table 6: Image stitching results on *IS1* dataset

Method	**Glob. opt.**	**Error x-axis** [px]	**Error y-axis** [px]
odometry	-	10.24 ± 7.40	12.35 ± 7.94
NCC	-	**5.36 ± 5.72**	**4.33 ± 4.59**
NCC	least squares	7.23 ± 5.41	6.01 ± 5.09
SIFT	-	9.90 ± 9.08	10.63 ± 10.22
SIFT	least squares	8.72 ± 6.45	10.36 ± 8.63

The results for image-based registration evaluated on *IS2* dataset with more precise odometry are summarized in Tab. 7. It shows that in the case of precise odometry, the other methods could not achieve nearly as low an error as the odometry itself and even worsened it. However, in this case, the feature-based approach outperformed the correlation-based one. In both cases, employing global optimization led to better results.

Table 7: Image stitching results on *IS2* dataset

Method	**Glob. opt.**	**Error x-axis** [px]	**Error y-axis** [px]
odometry	-	**1.67 ± 4.04**	**1.85 ± 4.51**
NCC	-	9.59 ± 7.32	9.53 ± 7.09
NCC	least squares	7.67 ± 6.28	8.77 ± 5.30
SIFT	-	8.07 ± 5.23	8.97 ± 6.11
SIFT	least squares	**5.40 ± 4.27**	**5.69 ± 4.73**

6.5 Cell Detection-based Registration of Pair of Image Tiles

When creating the initial map, we need to establish detection correspondences between pairs of image tiles. Since the maximum deviation of odometry can be almost twice the size of a typical honeybee cell, robust registration methods are necessary. To match the detections between images, we can first align the images using the image-based registration methods, described in Sec. 5.2.1 and then match the detections with the nearest neighbor approach. Another method is to use the cell detection-based registration, introduced in Sec. 5.2.2, with the criterion of maximal correspondences or use the visual descriptors of the cells with the criterion for matching image pairs. In this section, we evaluate the performance of all the mentioned methods.

6.5.1 Dataset

The dataset created for the classification of cells (see Sec. 6.2) also includes sequences of images captured over time for 22 image tiles across the honeybee comb, along with correspondences of cell detections between all images in the sequence. This dataset can be used to evaluate the registration of image pairs.

The dataset comprises a total of 23,809 image pairs. The maximum deviation of the odometry in the dataset is approximately 8.0 mm, almost twice the size of a typical honeybee cell. However, in 97.5% of image pairs, the deviation is smaller, approximately $O_{\text{dev}} = 5.7$ mm. Considering the large odometry deviation leaves room for errors. Moreover, we expect significantly more precise odometry information with the new system setup that will be used in the future. Therefore, we consider image pairs with odometry deviations greater than $O_{\text{dev}} = 5.7$ mm as outliers and exclude them from the evaluation. The size of the dataset without these outliers, denoted as *IP1*, is 23,216 image pairs. A few samples from the dataset are shown in Fig. 21.

To demonstrate the effect of the maximum odometry deviation on results, we also create a dataset by filtering out all image pairs with odometry deviations larger than $O_{\text{dev}} = 4.1$ mm, which is a reasonable estimate of the worst case with the new setup. This filtered dataset, denoted as *IP2*, contains a total of 21,544 image pairs.

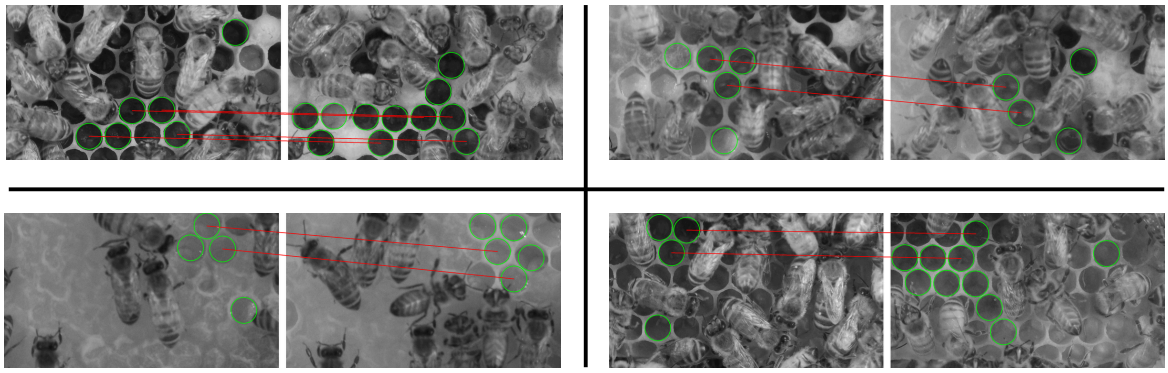


Figure 21: Samples from a dataset for cell detection-based image-image registration. Each pair comprises image tiles with detections (green) and correspondences between them (red lines).

6.5.2 Results

We evaluate the methods on the datasets described in the previous subsection. As we do not care about the specific pixel precision of the methods but rather about the fact that the methods produce correct correspondences between detections in the images, we use the number of correctly and incorrectly established correspondences and the number of rejected pairs as the metric.

First, we try the cross correlation-based image registration, denoted as "CC + NN", and the feature-based image registration with SIFT denoted as "SIFT + NN", which are used to align the pairs of images. We leverage the maximum deviation O_{dev} from the odometry as prior and limit the search space in a way discussed in Sec. 5.2.1. Subsequently, we perform nearest neighbor matching between the detections in the aligned image pairs, where we limit the maximum distance between the neighbors to $N_{\text{threshold}} = 2.2$ mm, which corresponds to about half of the typical size of a honeybee cell. If no such correspondences are found, the image pair is rejected.

We also evaluated the cell detection-based registration, presented in Sec. 5.2.2. The experiments were performed using the criterion of maximum number of correspondences, referred to as "spatial", and the criterion for matching image pairs with visual descriptors of cells, which we denote as "visual". The parameters of the methods, which were described in Sec. 5.2.2, were set as $G_{\text{threshold}} = O_{\text{dev}}$ mm and $L_{\text{threshold}} = 1.3$ mm. If no correspondences are found or the average cosine distance between correspondences, for the visual method, is greater than 0.5, the image pair is rejected.

The results of the methods evaluated on the dataset *IP1* are summarized in Tab. 8. It can be seen that the cell detection-based method with spatial criterion always finds correspondences between detections in the image pairs, although almost half of those are incorrect. The image-based registration methods with nearest neighbor matching have similar performance. However, as indicated by the number of rejected pairs, in many cases, these methods cannot provide correspondences between the detections in the images. We consider it the best-performing cell detection-based method with visual criterion, as it produced the largest number of correct correspondences while maintaining an acceptable number of rejected pairs

and incorrect correspondences.

The results of the methods evaluated on the dataset *IP1* with $O_{\text{dev}} = 5.7$ mm are summarized in Tab. 8. The cell detection-based method with the spatial criterion always finds correspondences between detections in the image pairs, although nearly half are incorrect. The image-based registration methods with nearest-neighbor matching have both a similar performance. However, these methods often fail to provide correspondences between detections, as indicated by the number of rejected pairs. The best-performing method is the cell detection-based approach with the visual criterion. It produced the highest number of correct correspondences while maintaining an acceptable number of rejected pairs and incorrect correspondences.

Table 8: Detection-based image pair registration results on *IP1* dataset

Method	**Correct**	**Incorrect**	**Rejected**
CC + NN	11,258	3,172	8,786
SIFT + NN	9,706	4,199	9,311
spatial	12,965	10,251	0
visual	**15,351**	**5,535**	**2,330**

In Tab. 9, the results on the dataset *IP2* with $O_{\text{dev}} = 4.1$ mm are shown. As expected, reducing the maximum odometry deviation results in fewer incorrectly established correspondences for all methods. The image-based registration methods with nearest neighbor matching continue to show similar performance, but the issue of rejected image pairs remains unsolved. The cell detection-based method with the spatial criterion shows significant improvement, producing fewer incorrectly matched image pairs. The best-performing method is the cell detection-based registration with the visual criterion, which, despite producing fewer correct correspondences than the same method with spatial criterion, resulted in half as many incorrect correspondences.

Table 9: Detection-based image pair registration results on *IP2* dataset

Method	**Correct**	**Incorrect**	**Rejected**
CC + NN	12,195	2,168	7,181
SIFT + NN	11,651	2,698	7,195
spatial	18,233	3,311	0
visual	**17,201**	**1,458**	**2,885**

6.6 Cell Detection-based Registration of Image to Map

Updating the map requires registration of the image tiles of the actual scan to the previous map, represented by a graph (see Sec. 5.2.4). As mentioned previously, due to imprecise odometry, the registration method must be robust to produce as few faulty matches as possible in cases where the correspondence is not straightforward. For this purpose, we designed a

cell detection-based registration method, described in Sec. 5.2.2. In this subsection, we first describe the created dataset for testing the image-map registration and then evaluate the performance of the designed method.

6.6.1 Dataset

Similarly to the previous section, we utilize a part of the dataset created for cell content classification, which contains sequences of images over time for 22 image tiles across the comb, together with correspondences between all images in the sequence. We use this dataset to create a previous map for each image tile and each individual image in their sequences, using the annotated correspondences. The resulting dataset comprises pairs of image tiles and their associated previous maps, created using all previous images in the sequence. In total, the dataset consists of 999 such pairs. The maximum deviation between the image tiles and maps is about 6.6 mm. For the same reasons discussed in Sec. 6.5, we consider pairs with deviations greater than $M_{\text{dev}} = 4.1$ mm as outliers and exclude them, resulting in a final dataset of 988 image-map pairs. A few samples from the dataset are depicted in Fig. 22.

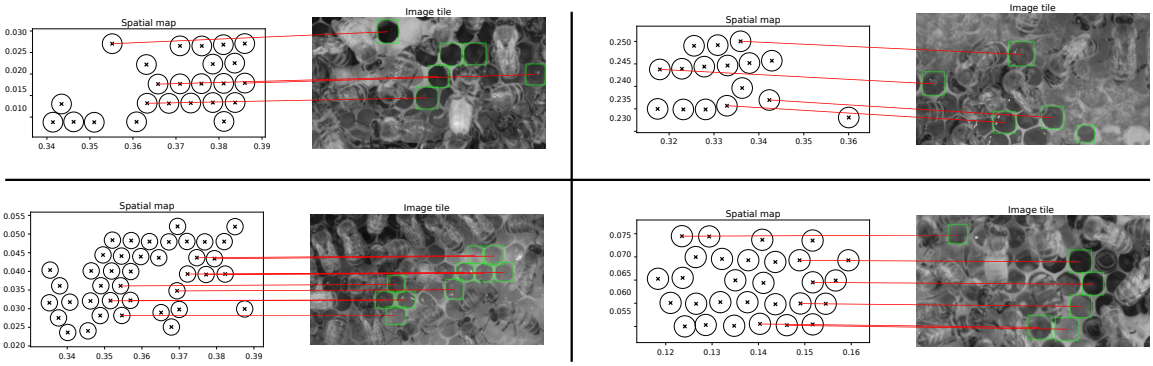


Figure 22: Samples from a dataset for cell detection-based image-map registration. Each pair comprises of an image tile with detections (green), previous map, and correspondences between them (red lines).

6.6.2 Results

For evaluation, we use the dataset introduced in the previous subsection. As the metric, we use the number of correctly and incorrectly established correspondences between the images and maps and the number of rejected pairs.

We assess the performance of three cell detection-based registration methods, each with different criteria for matching the images to a map, described in detail in Sec. 5.2.2. The first criterion, denoted as "spatial", is the maximum number of correspondences between the detections in the images and the maps. The second criterion, referred to as "visual", uses the visual descriptors of the cells and takes as the best match the one with the maximum mean cosine distance between the established correspondences. The third one, which we denote as "weighted visual", also uses the visual descriptors of the cells but employs three different weights, detailed in 5.2.2, and takes the best match based on a weighted sum of

the cosine distance between the created correspondences. If no correspondences are found or the average/weighted sum of cosine distances between correspondences, for the visual methods, is greater than 0.5, the pair is rejected. The parameters of the methods were set as $G_{\text{threshold}} = M_{\text{dev}} = 4.1$ mm and $L_{\text{threshold}} = 1.3$ mm.

The results are summarized in Tab. 10. It can be seen that all the methods perform similarly well. The largest number of correct correspondences was achieved using the cell detection-based registration with spatial criterion. However, this method also produced the most incorrect matches. The number of incorrectly established correspondences can be mitigated using visual and weighted visual criteria. The best-performing method is the one with the weighted visual criterion, producing the second-largest number of correct correspondences while creating the least number of incorrect ones.

Table 10: Detection-based image-map pair registration results

Method	**Correct**	**Incorrect**	**Rejected**
spatial criterion	955	33	0
visual criterion	933	22	33
weighted visual criterion	**947**	**20**	**21**

7 System Capabilities

In this section, we show the results of the honeybee comb mapping. Due to inconsistencies in the odometry information in the collected data, we do not create a map of the whole honeybee comb but only a part of it. We identified a part of the honeybee comb comprising six image tiles where the deviation of odometry was reasonable and created a map of this part. We used data collected between 9th October and 19th October, which consists of a total of 17 honeybee comb scans with zoomed images with a resolution of 25 μm per pixel. It should be noted that in the future, as the odometry inconsistencies were caused by a software bug, we expect the new system set up to have more consistent and precise odometry information. Therefore, we believe that creating a map of the whole comb will be a simple extension of our work.

The mapping process follows the description in Sec.5.2. For cell detection is used the Faster R-CNN with ResNet-50 backbone, which achieved the best results (Sec. 6.1). The cell classification is performed using the XResNet-18 model with a temporal filter. Correspondences between per-tile maps and between image tiles and the map are established using the cell detection-based methods using the XResNet-18 model with reduced size for visual description of the cells, as this method proved to be most suitable (see Sec. 6.5 and Sec. 6.6). The initial map is created using the first five honeybee comb scans.

This section will present technical details regarding the mapping process, together with various visualizations of the spatial and semantic map. Subsequently, we will discuss the biological application of our work.

7.1 Map Diagnostics

The statistics of the mapping process are summarized in Tab. 11. We report the number of registered image tiles in the map together with the number of registered cell detections, the number of rejected images and cell detections, the number of rejected images in which no cells were detected, and also the number of excluded cells from the map due to overlaps, which can indicate false positive cell detections or faulty matches between the map and the images. It can be seen that most of the images and detections were successfully registered to the map, and only a small portion of the data was rejected. The resulting map comprises a total of 158 cells.

Table 11: Map creation statistics

	Image tiles	**Detections**
Total registered	79	565
Total rejected	4	9
Total rejected without detections	19	-
Excluded cells	-	17

7.1.1 Spatial Map

The spatial map is a graph where each cell is represented by its positions in the hive coordinate frame. These positions are estimated from individual metric position measurements of the cell across different comb scans. The spatial map with all the measurements of cell positions is illustrated in Fig. 23. Apart from the estimated positions, we show 3-sigma covariance ellipses (where possible), indicating the error of position estimation.

The figure indicates that the covariance is more significant towards the center of the map. This is caused by the fact that the positions of cells close to the center are estimated from different image tiles that overlap. In contrast, the covariance is small in parts where the cell positions are estimated from measurements from a single image tile (on the sides of the map).

7.1.2 Semantic Map

The semantic map, shown in Fig. 24, is built on top of the spatial map. Apart from the metric positions of the cells and their estimated sizes, it contains the most recent semantic information about the classes of cells. If the cell was not observed for one day or more, the cell classes were predicted with the temporal filter.

The classes are visualized with the color of individual cells, and the last observations of the class are highlighted with the color of the cell edge. Confidences of the class predictions, which are in interval $[0, 1]$, are indicated by a (partial) black circle around the cells, where a full circle indicates confidence of 1. Apart from that, we add the "last seen" (LS) information, which is the number of days since the cell was last fully updated (with observation) by the temporal filter. We also present the entropy (E) of the cell states, which also indicates confidences of the cell state predictions.

Fig. 24 shows that cells near the center of the map were typically observed more recently than those at the edges. This is caused by the fact that cells in the center of the map can be observed in multiple image tiles. The last seen attribute of observations also indicates the prediction ability of the temporal filter for young bee cells. For instance, on the comb map's left side or upper side, there are larva cells that were not seen for many days. The temporal filter correctly predicts the actual class of the capped brood, which we do not detect. This points to the validity of our mapping approach based on open cell detections.

Class predictions are divided into seven categories (egg, larva, brood, other, pollen, nectar, and honey) and are determined by aggregating the corresponding states of the cells. As shown in Fig. 24, the class's confidence is generally close to or above 0.5. Conversely, entropy measures the certainty of cell states, indicating our confidence in the specific stages of young bee development (lower entropy indicates higher confidence). It can be seen that in some cases, while we may be confident in the class prediction, we may be less sure about the exact day within the development stages of young bees.

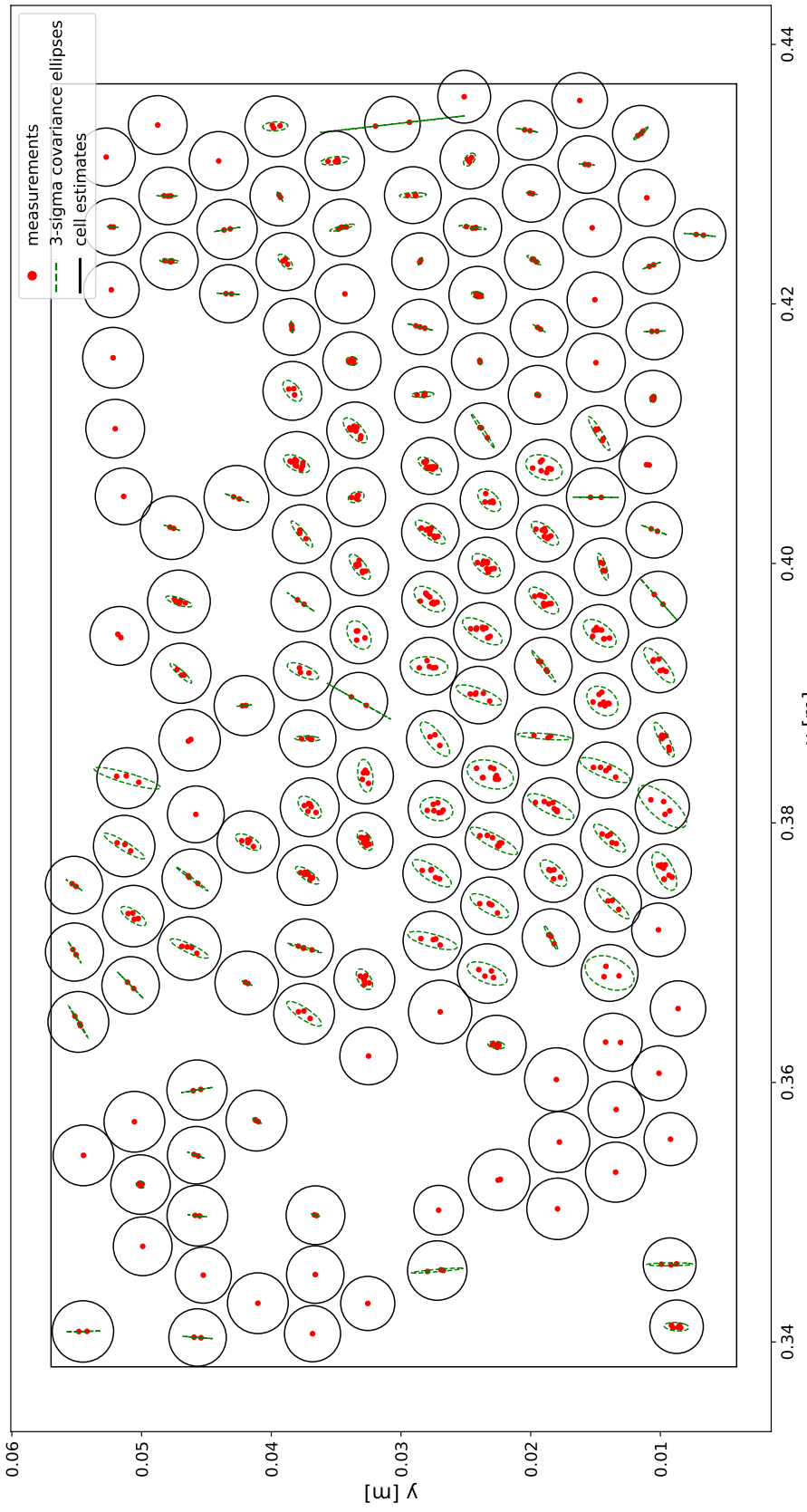


Figure 23: The resulting spatial map of the honeybee comb. Each cell is illustrated with a circle of its estimated size at the mean position in the hive coordinate frame. Individual position measurements are highlighted in red. Additionally, 3-sigma covariance ellipses are reported for each cell.

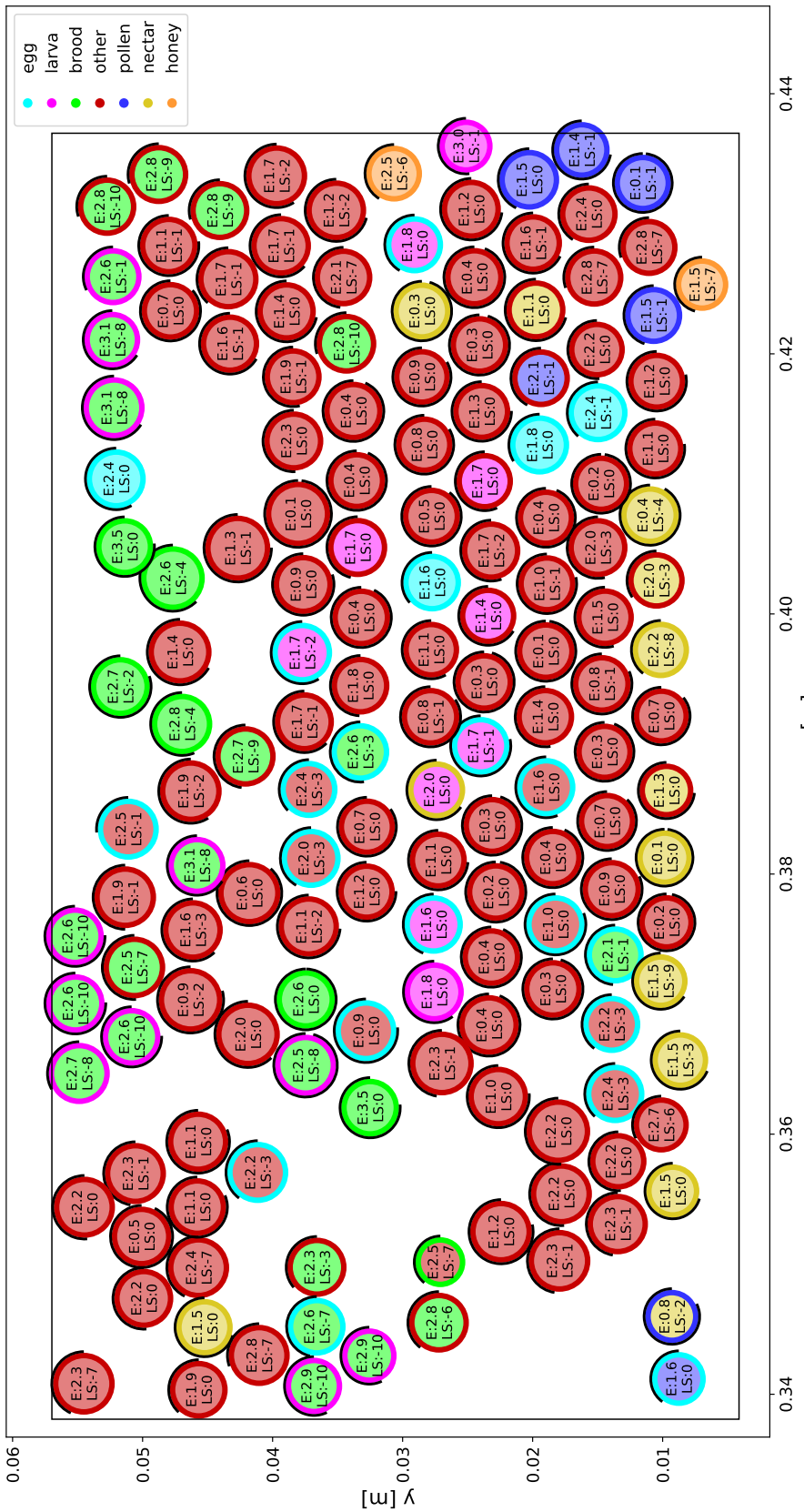


Figure 24: The resulting semantic map of the honeybee comb. Cell classes are depicted by the colors of the circles, while the edge color represents the last observation. The confidence level of each cell state is shown by a black (partial) circle on the cell's edge, where a full circle represents a confidence value of 1.0. Additionally, entropy (E) for each cell and "last seen" (LS), indicating the number of days since the cell was last fully updated by the temporal filter, are reported.

7.1.3 Visualization for Debugging Purposes

It is helpful to have the ability to supervise the system and inspect whether the mapping process produces reasonable outputs. For this purpose, when the map is updated with a new scan, the user is provided with the resulting map combined with the stitched images from the most recent scan and information about the number of cell observations and whether it was seen in the recent scan. If the cells were not observed for one day or more, we also apply the prediction step of the temporal filter to get an actual prediction of cell classes. Such visualization is shown in Fig. 25. It should be noted that the cells in the images may not perfectly align with the map of the comb, which is expected because of the process of estimating the cell positions independently based on position measurements across different scans.

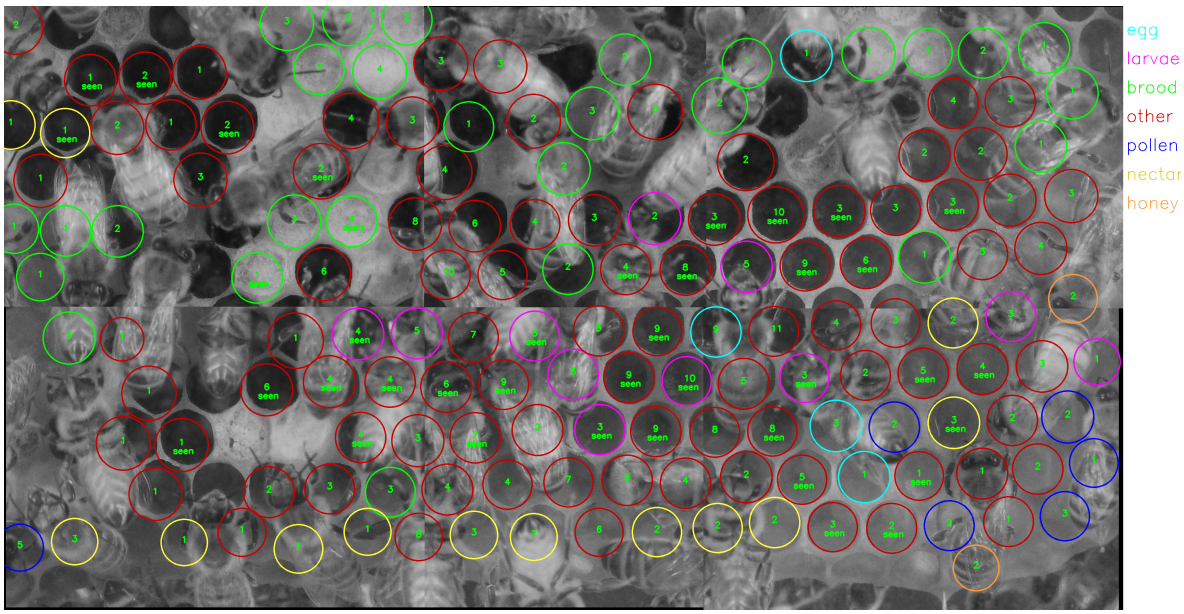


Figure 25: Visualization of the map with comb scan images for debugging purposes. Cell states are indicated by different colors. Additionally, the number of observations for each cell and whether they were seen in the actual scan are reported. Note that slight deviations in cell positions on top of the images are expected due to the cell position estimation process.

7.1.4 Integration into ROS RViz Tool

To show that we mean "buzziness", when it comes to mapping of the honeybee comb, we integrated the visualization of the map into Robot Operating System visualization tool (RViz). Each cell is published with its estimated metric position in the coordinate frame of the hive, together with its estimated size and predicted class of the cell. Additionally, we include the information "last seen" (LS), indicating the number of days since the cell was last observed with a full update of the temporal filter. This should be sufficient for a user to identify comb regions that could be preferred for visiting with a camera to collect more information. The visualization is shown in Fig. 26.

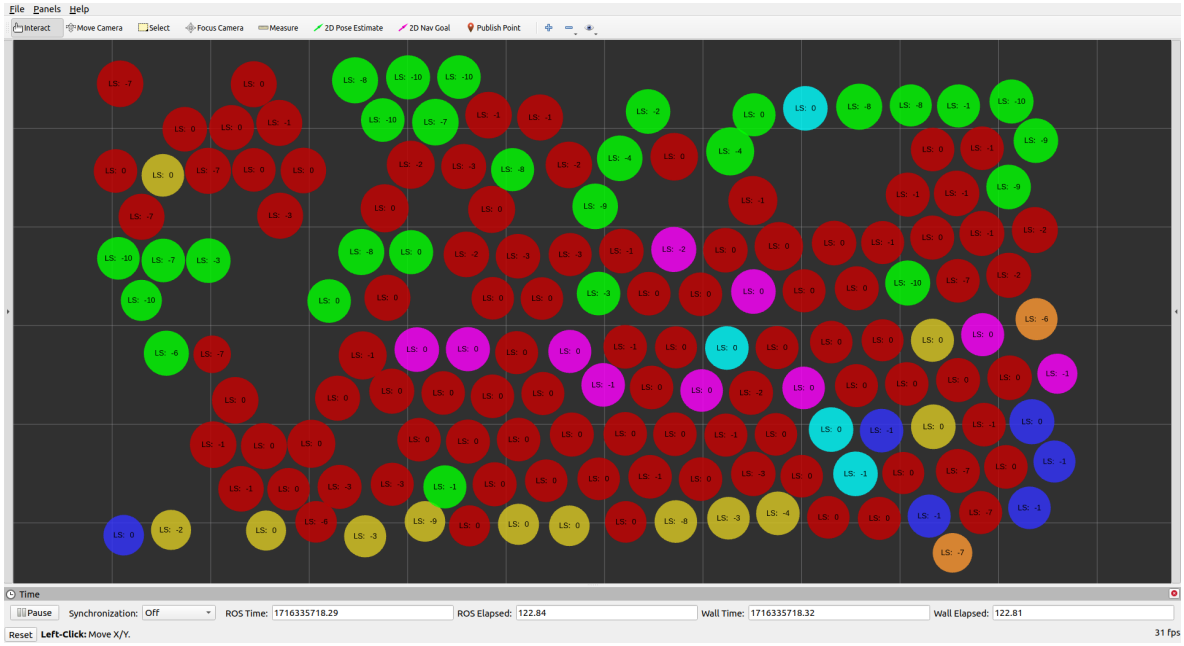


Figure 26: RViz visualization of the resulting map for the user with the estimated positions in the hive coordinate frame. States are highlighted with specific colors (light blue for eggs, purple for larvae, green for brood, red for other, blue for pollen, yellow for nectar, and orange for honey). Additionally, each cell has an associated "last seen" (LS) attribute, indicating the number of days since the cell was last observed with a complete update of the temporal filter. The resolution of the grid is 1 cm per grid cell.

7.2 Application in Biological Sciences

The number of cells in different classes can be used to assess the state and strength of the honeybee colony, which is crucial for evaluating the impact of interactions with the honeybee queen planned in the RoboRoyale project (see Sec. 2). This section presents a way of using the created comb map for such evaluation.

We use a sampling-based estimation to determine the expected value of the number of different cell classes in the map. We consider the model of the whole comb to be an independent collection of the cells and can, therefore, take samples of the whole comb by sampling individual cells according to our belief distribution over the states. From the samples, we can estimate the total number of cells in each class and the variation under our uncertainty. Since the number of cells in the map varies because the map is updated incrementally, we assume that the states of cells not yet observed are uniformly distributed across the cell classes.

The expected values for the number of cell types in time and standard deviations are visualized in Fig. 28, with 5000 samples used for each datapoint. In the beginning, when most of the cells were not yet observed, the number of cell types was spread almost uniformly. The numbers became more representative as new cells were added to the map. Similarly, the standard deviation is higher towards the beginning and decreases with more cells observed and registered to the map. However, it can be seen that for some types of cells, the standard deviation does not decrease significantly, which would be expected. We believe that this is

caused by the fact that most of the cells are not observed very frequently, as shown in the histogram of the number of observations per cell in Fig. 27.

Fig. 28 further shows the development of young bees in the hive. For instance, we can see larvae transition into brood cells between days 6 and 10. Similarly, between days 9 and 10, the number of egg cells decreases; this could indicate either egg cannibalism in the hive or false cell type observations by the classification neural network.

To showcase the potential of using the temporal filter, we also predicted the evolution of the expected number of various cells five days into the future, depicted in Fig. 28. The prediction indicates that the parameters of the temporal filter may not be adequately tuned as the number of nectar and pollen cells spontaneously increases, which is not expected to be frequent in the colony. However, we are aware of this problem, and in the future, we aim to tune the temporal filter to the colony representatively by learning the parameters from the data.

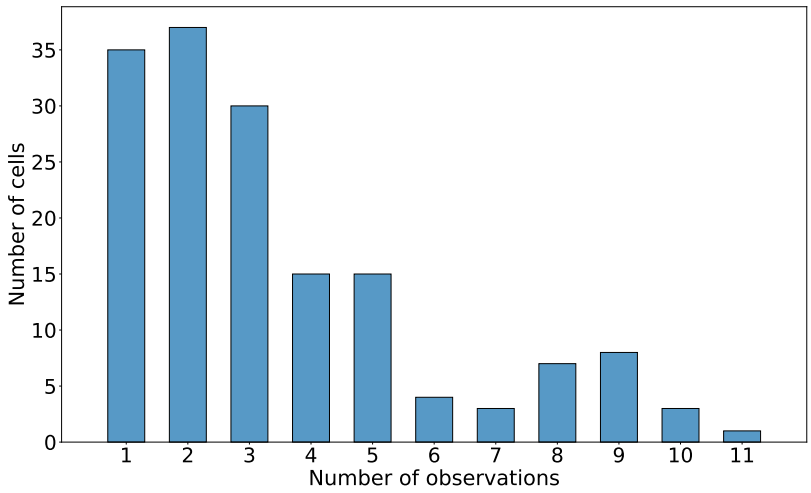


Figure 27: Distribution of observations collected for each cell. Due to clutter and crawling bees, we cannot detect every cell every time, and here we see that only a few had over ten observations in the 17 collected scans.

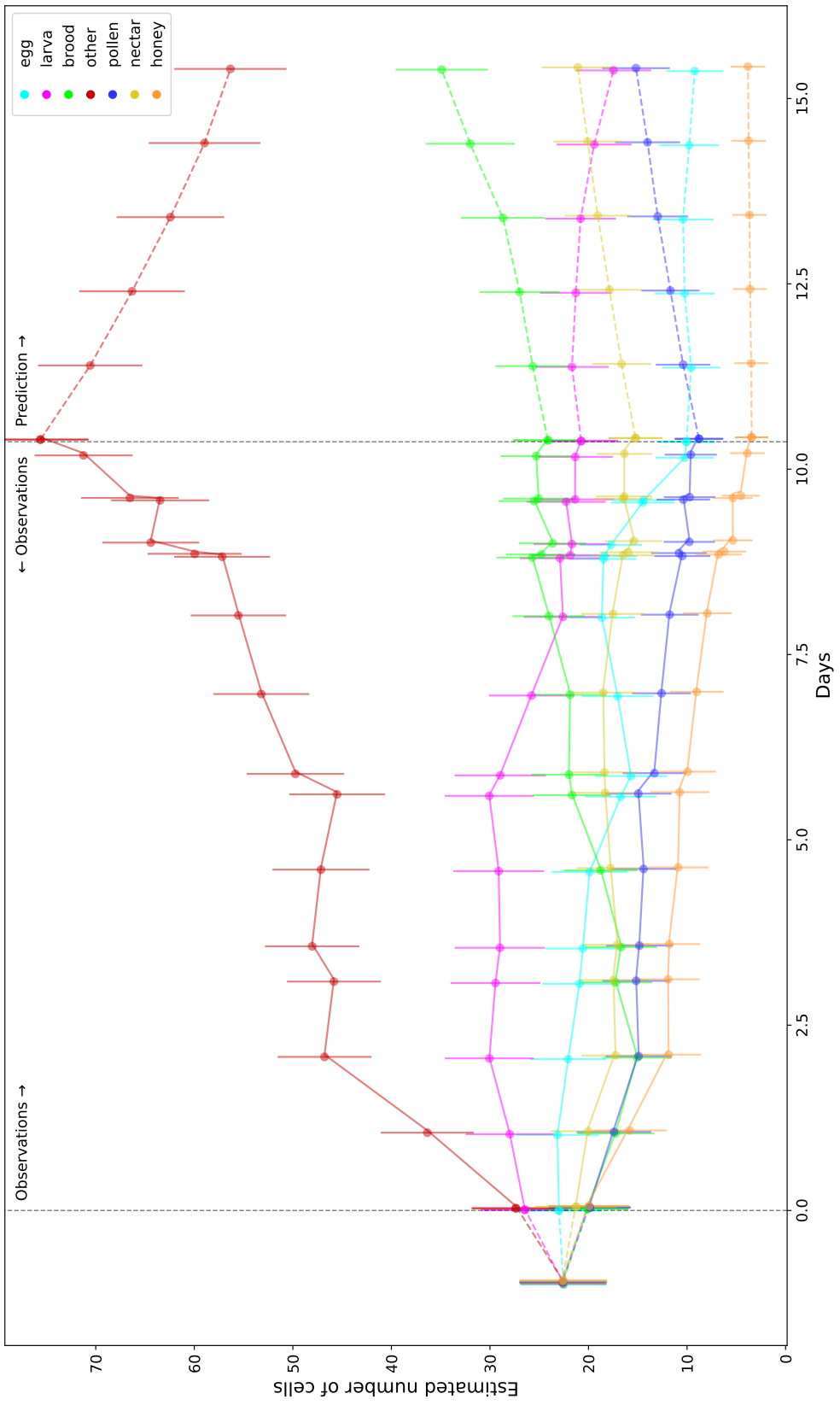


Figure 28: Aggregated estimates of the total number of cells of individual classes and their evolution in time. Based on a simple sampling scheme, we estimate the total numbers as mean sums of the sampled values. This way, we also get the corresponding standard deviations. Note that the total number of cells, i.e., the sum of all lines at each point in time, stays the same and is only redistributed between different classes. Before the first observations (day 0), we show estimate for purely uniform distribution of cell classes.

8 Conclusion

The aim of this work was to create a system for automated mapping of honeybee comb and its contents by using scans of the honeybee comb collected by a moving camera. The automated creation of a semantic map of the comb is necessary for the assessment of the state and strength of the colony in its natural environment. This is impossible by manual observation due to the scale and without disrupting the colony, which is why the automated system is important to the RoboRoyale project.

From the technical perspective, we had to solve the problem of just-in-time inference of the global state of a highly dynamic, partially observable system from past sparse and irregular observations. We developed the mapping algorithm using data recorded during the 2023 season on the first version of the robot and showed it integrated into the ROS visualization stack. Despite our efforts, we only succeeded in creating the map in a specific part of the hive. We could not make a complete map of the honeybee comb due to a software issue in the data collection, causing odometric errors in the recorded data, which we were not able to repair. In some parts of the hive, these were not so severe as to prevent our methods from working, and we have shown that these odometry inconsistencies have been eliminated in the new version of the system already collecting data in the 2024 season. Therefore, we are confident that creating a map of the entire honeybee comb will be a simple application of our methods on the newly collected data.

8.1 Summary of our Solution

The mapping is approached as object-based mapping with cells as the objects of interest. We tested multiple methods for open cell detection and showed that a common approach with Circle Hough Transform for cell detection is not sufficient when applied in images with the presence of bees. The best performance was achieved with the Faster R-CNN network with ResNet-50 backbone, fine-tuned on our own annotated dataset.

We compared multiple registration methods of image pairs and image-map pairs. We evaluated their performance when used in a highly dynamic and cluttered environment with repetitive patterns and their robustness in case of inconsistent odometry information that was present in the collected data. The best results were obtained with a hand-crafted registration using cell detections as features and XResNet-18 Siamese neural network, trained on our own annotated dataset, for a visual description of individual cells.

To extract semantic information about the cells based on their content, we employed the XResNet-18 classification neural network, fine-tuned on our annotated dataset. We combined it with the Bayes Filter for temporal filtering. The results indicate the capability to enhance the classification results and improve our understanding of the cell states by providing detailed predictions of the young bee development.

Partial results of the work on image-based registration and cell detection were accepted for conference publication [102] during the work on this thesis.

8.2 Future Work

In the future, we would like to test and eventually retrain our models on the data collected in the 2024 season, fully integrate our solution into the RoboRoyale system, and create a map of the entire honeybee comb.

We are aware of the problem of tunning the temporal filter, which is caused by the general absence of detailed long-term observations of honeybee colonies. Not to mention, these parameters depend on the specific hive, weather conditions, and time of the season. In the RoboRoyale project, we want to learn how to handle this beeness using data from future experiments.

The temporal filter itself could be extended by considering the spatial correlation of the cell types and refining the cell state predictions by recreating the most likely sequence. Another aspect that could be addressed is the discretization of time, as our observations happen in irregular time intervals.

In this work, we aimed to describe the temporal model of individual cells on the comb. In the future, we need to create a complete model of the colony and its development. We aim to use the estimated numbers of cell types in the hive to estimate the state of the whole system full of bugs, evaluating the storage and presumed development of future honeybee generations.

References

[1] Gretchen LeBuhn and Joshua Vargas Luna. Pollinator decline: what do we know about the drivers of solitary bee declines? *Current Opinion in Insect Science*, 46:106–111, 2021. Special Section on Pollinator decline: human and policy dimensions * Social insects.

[2] Rafael Barmak et al. A robotic honeycomb for interaction with a honeybee colony. *Science Robotics*, 8(76):eadd7385, 2023.

[3] Donato Romano, Maurizio Porfiri, Payam Zahadat, and Thomas Schmickl. Animal–robot interaction—an emerging field at the intersection of biology and robotics. *Bioinspiration & Biomimetics*, 19(2):020201, feb 2024.

[4] Kristi Žampachů et al. A vision-based system for social insect tracking. In *2nd International Conference on Robotics, Automation and Artificial Intelligence (RAAI)*, pages 277–283, 2022.

[5] Tim Gernat et al. Automated monitoring of honey bees with barcodes and artificial intelligence reveals two distinct social networks from a single affiliative behavior. *Scientific Reports*, 13(1), Jan 2023.

[6] Katarzyna Bozek et al. Markerless tracking of an entire honey bee colony. *Nature Communications*, 12(1), 2021.

[7] Martin Stefanec et al. A minimally invasive approach towards "ecosystem hacking" with honeybees. *Frontiers in Robotics and AI*, 9, Apr 2022.

[8] Mrudul Chellapurath et al. Bioinspired robots can foster nature conservation. *Frontiers in Robotics and AI*, 10, 2023.

[9] Jerry Bromenshenk et al. Bees as biosensors: Chemosensory ability, honey bee monitoring systems, and emergent sensor technologies derived from the pollinator syndrome. *Biosensors*, 5(4):678–711, Oct 2015.

[10] Wiktoria Rajewicz et al. Organisms as sensors in biohybrid entities as a novel tool for in-field aquatic monitoring. *Bioinspiration & Biomimetics*, 19(1):015001, Nov 2023.

[11] Joanna Bagniewska. Bomb-ble bees: Insects to sniff out explosives. YouTube, 2013. `https://www.youtube.com/watch?v=DMGbf6pGOUg`.

[12] Farshad Arvin. Roboroyale project. `https://roboroyale.eu/`.

[13] Jiří Ulrich et al. Long-term tracking of individual, collective and social behaviors in honeybees by cooperating robots. *Science Robotics*, in review.

[14] Rekabi-Bana et al. Mechatronic design for multi robots-insect swarms interactions. In *IEEE International Conference on Mechatronics (ICM)*. IEEE, 2023.

[15] Open Robotics. Robotic operating system. `https://www.ros.org/`.

[16] Alemayehu Gela. *Molecular Analysis during the embryo development of honeybees*. LAP LAMBERT Academic Publishing, 07 2014.

[17] T V Ramachandra et al. Beekeeping: Sustainable livelihood option in uttara kannada, central western ghats. *ENVIS Technical Report: 49*, 01 2012.

[18] Jerzy Woyke. Cannibalism and brood-rearing efficiency in the honeybee. *Journal of Apicultural Research*, 16:84–94, 01 1977.

[19] P. J. Gullan and P. S. Cranston. *Insects: An Outline of Entomology*. Wiley Blackwell, Chichester, 5th edition, 2014.

[20] Denis F. Wolf and Gaurav S. Sukhatme. Semantic mapping using mobile robots. *IEEE Transactions on Robotics*, 24(2):245–258, 2008.

[21] Xiaoning Han et al. Semantic mapping for mobile robots in indoor scenes: A survey. *Information*, 12(2):92, Feb 2021.

[22] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86–103, Apr 2015.

[23] Dagmar Lang and Dietrich Paulus. Semantic maps for robotics, 2014.

[24] Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, Nov 2008.

[25] Richard A. Newcombe et al. Dtam: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, 2011.

[26] Jakob Engel et al. Lsd-slam: Large-scale direct monocular slam. In *Computer Vision – ECCV 2014*, pages 834–849. Springer International Publishing, 2014.

[27] Raul Mur-Artal et al. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[28] Phi-Hung Le and Jana Kosecka. Dense piecewise planar rgb-d slam for indoor environments. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep 2017.

[29] Shichao Yang et al. Pop-up slam: Semantic monocular plane slam for low-texture environments. In *Proceedings of (IROS) IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1222 – 1229, October 2016.

[30] Renato F. Salas-Moreno et al. Slam++: Simultaneous localisation and mapping at the level of objects. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, 2013.

[31] Lachlan Nicholson et al. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, 4(1):1–8, 2019.

[32] Jonathan Crespo et al. Semantic information for robot navigation: A survey. *Applied Sciences*, 10(2):497, Jan 2020.

[33] Carl Case et al. Autonomous sign reading for semantic mapping. In *2011 IEEE International Conference on Robotics and Automation*, pages 3297–3303, 2011.

[34] J. Crespo et al. Relational model for robotic semantic navigation in indoor environments. *Journal of Intelligent & Robotic Systems*, 86(3–4):617–639, Jan 2017.

[35] H. Zender et al. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6):493–502, Jun 2008.

[36] E. Bastianelli et al. On-line semantic mapping. In *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–6, 2013.

[37] R O Castle et al. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4102–4107, 2007.

[38] Andrzej Pronobis and Patric Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *2012 IEEE International Conference on Robotics and Automation*, pages 3515–3522, 2012.

[39] David Meger et al. Curious george: An attentive semantic robot. *Robotics and Autonomous Systems*, 56(6):503–511, Jun 2008.

[40] Niko Sünderhauf et al. Meaningful maps with object-oriented semantic mapping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5079–5085, 2017.

[41] Zhen Zeng et al. Semantic mapping with simultaneous object detection and localization. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018.

[42] Tsung-Yi Lin et al. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[43] Aditya Singh et al. Efficient deep learning-based semantic mapping approach using monocular vision for resource-limited mobile robots. *Neural Computing and Applications*, 34(18):15617–15631, Apr 2022.

[44] Jianxin Wu et al. Visual place categorization: Problem, dataset, and algorithm. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4763–4770, 2009.

[45] Jianxin Wu and Jim M. Rehg. Centrist: A visual descriptor for scene categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1489–1501, 2011.

[46] Niko Sünderhauf et al. Place categorization and semantic mapping on a mobile robot. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5729–5736, 2016.

[47] Emma Brunskill et al. Topological mapping using spectral clustering and classification. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3491–3496, 2007.

[48] Stephen Friedman et al. Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In *International Joint Conference on Artificial Intelligence*, pages 2109–2114, 01 2007.

[49] Nils Goerke and Sven Braun. Building semantic annotated maps by mobile robots. 01 2009.

[50] Radu Bogdan Rusu et al. Model-based and learned semantic object labeling in 3d point cloud maps of kitchen environments. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3601–3608, 2009.

[51] Nico Blodow et al. Autonomous semantic mapping for robots performing everyday manipulation tasks in kitchen environments. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4263–4270, 2011.

[52] Alexander Hermans et al. Dense 3d semantic mapping of indoor scenes from rgb-d images. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2631–2638, 2014.

[53] Jörg Stückler et al. Semantic mapping using object-class segmentation of rgb-d images. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3005–3010, 2012.

[54] John McCormac et al. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.

[55] Xuanpeng Li and Rachid Belaroussi. Semi-dense 3d semantic mapping from monocular slam. *ArXiv*, abs/1611.04144, 2016.

[56] C. Galindo et al. Multi-hierarchical semantic maps for mobile robotics. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2278–2283, 2005.

[57] Shrihari Vasudevan and Roland Siegwart. Bayesian space conceptualization and place classification for semantic maps in mobile robotics. *Robotics and Autonomous Systems*, 56(6):522–537, 2008. From Sensors to Human Spatial Concepts.

[58] Sebastian Thrun et al. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[59] Stephan Saalfeld et al. As-rigid-as-possible mosaicking and serial section registration of large sstem datasets. *Bioinformatics*, 26(12):i57–i63, 2010.

[60] Albert Cardona et al. Trakem2 software for neural circuit reconstruction. *PLOS ONE*, 7(6):1–8, 06 2012.

[61] Kaiyue Li and Guangtai Ding. A novel automatic image stitching algorithm for ceramic microscopic images. In *2018 International Conference on Audio, Language and Image Processing (ICALIP)*, pages 17–21, 2018.

[62] Gayathri Mahalingam et al. A scalable and modular automated pipeline for stitching of large electron microscopy datasets. *eLife*, 11, 2022.

[63] Petr Šilling. Deep learning for image stitching. Master's thesis, Brno University of Technology, 2023.

[64] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, page 91–110, 2004.

[65] Chandler D. Gatenbee et al. Virtual alignment of pathology image series for multi-gigapixel whole slide images. *Nature Communications*, 14(1), 2023.

[66] Daniel DeTone et al. Superpoint: Self-supervised interest point detection and description. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop*, 2018.

[67] Paul-Edouard Sarlin et al. Superglue: Learning feature matching with graph neural networks. *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020.

[68] Jiaming Sun et al. Loftr: Detector-free local feature matching with transformers. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[69] Dirk Steckhan et al. Efficient large scale image stitching for virtual microscopy. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4019–4023, 2008.

[70] M. Emmenlauer et al. Xuvtools: Free, fast and reliable stitching of large 3d datasets. *Journal of Microscopy*, 233(1):42–60, 2009.

[71] Stephan Preibisch et al. Globally optimal stitching of tiled 3d microscopic image acquisitions. *Bioinformatics*, 25(11):1463–1465, 2009.

[72] Tolga Tasdizen et al. Automatic mosaicking and volume assembly for high-throughput serial-section transmission electron microscopy. *Journal of Neuroscience Methods*, 2010.

[73] Joe Chalfoun et al. Mist: Accurate and scalable microscopy image stitching tool with stage modeling and error minimization. *Scientific Reports*, 7(1), 2017.

[74] Dženan Zukić et al. Itkmontage: A software module for image stitching. *Integrating Materials and Manufacturing Innovation*, 10(1), 2021.

[75] Jeremy L Muhlich et al. Stitching and registering highly multiplexed whole-slide images of tissues and tumors using ASHLAR. *Bioinformatics*, 38(19):4613–4621, 08 2022.

[76] S. K. Chow et al. Automated microscopy system for mosaic acquisition and processing. *Journal of Microscopy*, 222(2):76–84, 2006.

[77] Heiko Bulow et al. Online generation of an underwater photo map with improved fourier mellin based registration. In *OCEANS 2009-EUROPE*, pages 1–6, 2009.

[78] Maria Alejandra Palacio et al. Evaluation of the time of uncapping and removing dead brood from cells by hygienic and non-hygienic honey bees. *Genet. Mol. Res*, 4(1):105–114, 2005.

[79] Vincent Dietemann et al. Standard methods for varroa research. *Journal of apicultural research*, 52(1):1–54, 2013.

[80] Lukas Jeker et al. Computer-assisted digital image analysis and evaluation of brood development in honey bee combs. *Journal of Apicultural Research*, 51(1):63–73, 2012.

[81] Uwe Knauer et al. Application of an adaptive background model for monitoring honeybees. 2005.

[82] Pedro Rodrigues et al. Geometric contrast feature for automatic visual counting of honey bee brood capped cells. *In EURBEE 2016: 7th European Conference of Apidology. Cluj-Napoca, Romania*, 2016.

[83] Amelia Carolina Sparavigna. Analysis of a natural honeycomb by means of an image segmentation. *Philica*, 2016(897), December 2016.

[84] J. Illingworth and J. Kittler. The adaptive hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):690–698, 1987.

[85] Lee Hung Liew et al. Cell detection for bee comb images using circular hough transformation. In *International Conference on Science and Social Research (CSSR)*, 2010.

[86] Benjamin Höferlin et al. Automatic analysis of apis mellifera comb photos and brood development. In *Association of Institutes for Bee Research Report of the 60th Seminar in Würzburg*, volume 44, page 19, 03 2013.

[87] Théotime Colin et al. The development of honey bee colonies assessed using a new semi-automated brood counting method: Combcount. *PLOS ONE*, 13(10), 2018.

[88] Thiago S. Alves et al. Automatic detection and classification of honey bee comb cells using deep learning. *Computers and Electronics in Agriculture*, 170:105244, 2020.

[89] Gianluigi Paolillo et al. Automated image analysis to assess hygienic behaviour of honeybees. *PLOS ONE*, 2022.

[90] Neha Rathore et al. Semi-automatic analysis of cells in honeybee comb images. In *2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, pages 1–4, 2023.

[91] Adrian Defèr. Classification of honeybee larval stages using cnns applied to image data. Master's thesis, Freie Universität Berlin, 2022.

[92] Uwe Knauer et al. A comparison of classifiers for prescreening of honeybee brood cells. In *The 5th International Conference on Computer Vision Systems*, 03 2007.

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

[94] Shaoqing Ren et al. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[95] Hiroto Honda. Digging into detectron 2 - part 1. `https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd/`, Jan 2020. Acessed: May 24, 2024.

[96] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. `https://github.com/pytorch/vision`, 2016.

[97] Kaiming He et al. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[98] Renjie Xu et al. A forest fire detection system based on ensemble learning. *Forests*, 12(2), 2021.

[99] Glenn Jocher. YOLOv5 by ultralytics. `https://github.com/ultralytics/yolov5/`, 5 2020.

[100] Florian Schroff et al. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015.

[101] Tong He et al. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[102] Jiří Janota et al. Towards robotic mapping of a honeybee comb. In *International Conference on Manipulation, Automation and Robotics at Small Scales (MARSS)*, to appear, 2024.

[103] Alexander Kirillov et al. Segment anything. *arXiv:2304.02643*, 2023.

A Declaration of Using AI Tools

During the preparation of this work, the author utilized ChatGPT to enhance readability and Grammarly for grammar corrections. After using these services, the author diligently reviewed and edited the content as necessary, assuming full responsibility for the final work.

B List of Attachments

Table 12: Attachment content

Name	**Description**
datasets/	structure for the annotated datasets
models/	structure for the trained models
src/	source codes of the proposed methods and experiments, including traning of the neural networks
README.md	description of the repository with links to datasets and trained models
env.yaml	conda environment definition