



**Universität  
Zürich<sup>UZH</sup>**

## MASTER THESIS

Department of Informatics – Informatics and Sustainability Research  
Faculty of Business, Economics and Informatics

---

# Automated, Non-Invasive Integration of Pollen Metrics into the BeeLivingSensor Platform

---



*Author:*

**Anna Jancso**

13-700-638

*Supervisors:*

Prof. Dr. Lorenz Hilty

Dr. Clemens Mader

Daniel Boschung

Submission Date: July 10, 2021

## Abstract

Honeybees are one of the most important pollinators in our world and are therefore vital to biodiversity and economy. Over the past decades, bee populations have been dwindling. Although the causes are known, the exact impact of each cause is still unclear. The *BeeLivingSensor* project aims to shed light on this intricate question by aggregating data around bees in an automated, non-invasive fashion.

The goal of this thesis was to integrate two pollen metrics into our web-based platform that serve as an indicator for biodiversity and bee health: the number of pollen and the floral diversity of the pollen. For the first metric, we constructed a fully scalable end-to-end system for tracking pollen using existing computer vision solutions. For the second metric, we developed an algorithm for matching plants that bloom in a certain area at a given time. We evaluated both metrics in terms of accuracy, showing that the pollen counting mechanism needs improvement, whereas the plant-matching already delivers usable results. We also measured the speed for computing the pollen counts, demonstrating that we can process the daily amount of data of two hives on a strong GPU machine.

## Zusammenfassung

Honigbienen gehören zu den wichtigsten Bestäubern auf unserer Welt und sind deshalb unverzichtbar für die Biodiversität und die Wirtschaft. In den vergangenen Jahrzehnten haben die Bienenpopulationen abgenommen. Obwohl die Ursachen bekannt sind, sind die Auswirkungen jeder Ursache noch unklar. Das *BeeLivingSensor*-Projekt beabsichtigt Aufschluss über diese komplexe Frage zu geben, indem es Daten rundum Bienen auf automatisierte, nicht-invasive Weise aggregiert.

Das Ziel dieser Arbeit war zwei Pollenkennzahlen in unsere webbasierte Plattform zu integrieren, die als Indikator für Biodiversität und Bienengesundheit fungieren: die Pollenanzahl und die florale Vielfalt der Pollen. Für die erste Kennzahl haben wir zur Verfolgung des Pollen ein voll skalierbares End-zu-End System konstruiert unter Verwendung von Computervisionlösungen. Für die zweite Kennzahl haben wir einen Algorithmus entwickelt, der Pflanzen zu einem gegebenen Zeitpunkt in der Gegend abgleicht. Wir haben beide Kennzahlen hinsichtlich ihrer Genauigkeit evaluiert. Dabei konnten wir zeigen, dass der Pollenzählmechanismus noch verbesserungswürdig ist, während der Pflanzenabgleich schon verwertbare Resultate liefert. Überdies haben wir die Geschwindigkeit zur Berechnung der Pollenanzahl gemessen. Hier konnten wir darlegen, dass wir die täglichen Datenmengen von zwei Bienenstöcken auf einer starken GPU Maschine verarbeiten können.

## Acknowledgments

I would like to thank my supervisors, Prof. Dr. Lorenz Hilty, Dr. Clemens Mader and Daniel Boschung, for allowing me to work on such a challenging, yet intriguing and rewarding project. Not only did I learn a lot in technical respect, I also learnt a great deal about bees and plants, coming to appreciate their role in our world even more. Special thanks go to Daniel and Clemens who lead the project with such great enthusiasm and commitment. In particular, I want to express my sincerest gratitude to Daniel who continually supported and guided me throughout my thesis, be it through helping me with the tedious evaluation of the system or obtaining the necessary equipment and financial resources but also putting me in contact with the right people or simply giving me regular feedback and inputs to my work. Merci danibo for being such a great mentor! I am also grateful to all the people Daniel put me into contact with: Francis Cordillot with his expert knowledge on plants, Katharina Bieri who performed the pollen analysis and Prof. Dr. Ernst Hafen funding the costs for it, and the Info Flora and PhaenoNet team, with whom we had fruitful exchanges and who kindly provide their data for free. Last but not least, I want to thank my family for continuously supporting and motivating me throughout my studies.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Research Question . . . . .	1
<b>2</b>	<b>Background</b>	<b>2</b>
2.1	Botanic Basics . . . . .	2
2.2	BeeLivingSensor Project . . . . .	4
<b>3</b>	<b>Previous Technical Contributions to BeeLivingSensor</b>	<b>7</b>
3.1	AI Models & Trackers . . . . .	7
3.2	First Prototype of Platform . . . . .	9
<b>4</b>	<b>Performance of Pollen Counting</b>	<b>10</b>
4.1	Related Work . . . . .	11
4.2	Implementation . . . . .	13
4.2.1	Model-Service Abstractions . . . . .	13
4.2.2	Media Data Flow . . . . .	13
4.2.3	Media Process Scheduler . . . . .	16
4.2.4	Model Inference Pipeline . . . . .	16
4.2.5	Pollen Color Extraction . . . . .	17
4.2.6	Bee Tracking . . . . .	18
4.2.7	Pollen Tracking . . . . .	19
4.2.8	GPU Processing on Kubernetes . . . . .	20
4.2.9	Parallelization with Threads and Celery . . . . .	21
4.3	Accuracy Evaluation . . . . .	25
4.3.1	Results . . . . .	27
4.3.2	Discussion . . . . .	28
4.4	Speed Evaluation . . . . .	29
4.4.1	Results . . . . .	29
4.4.2	Discussion . . . . .	30
<b>5</b>	<b>Systems Providing Flora Data</b>	<b>32</b>
5.1	Methodology . . . . .	32
5.2	Plant Observation . . . . .	34
5.2.1	BeeKeepr . . . . .	34
5.2.2	Flora Incognita . . . . .	35
5.2.3	iNaturalist . . . . .	36
5.2.4	Info Flora . . . . .	39
5.2.5	PhaenoNet . . . . .	40
5.2.6	Pl@ntNet . . . . .	41
5.2.7	Discussion . . . . .	43
5.3	Pollen Color . . . . .	44
5.4	Food Abundance . . . . .	45

---

<b>6</b>	<b>Plant-Matching Algorithm</b>	<b>45</b>
6.1	Related Work . . . . .	46
6.1.1	Palynology . . . . .	46
6.1.2	Data-Driven Approach . . . . .	47
6.2	Implementation . . . . .	47
6.2.1	Database Schema . . . . .	47
6.2.2	iNaturalist Integration . . . . .	48
6.2.3	InfoFlora Integration . . . . .	49
6.2.4	Pollen Color Integration . . . . .	49
6.2.5	Pollen Abundance Integration . . . . .	49
6.2.6	Plant-Matching Algorithm . . . . .	49
6.3	Evaluation . . . . .	52
6.3.1	Results . . . . .	54
6.3.2	Discussion . . . . .	54
<b>7</b>	<b>Conclusion &amp; Outlook</b>	<b>58</b>
	<b>Bibliography</b>	<b>59</b>
	<b>Glossary</b>	<b>67</b>

## 1 Introduction

Over the past decades, the amount of flying insect biomass has dramatically decreased [33]. Many, primarily human-induced, causes have been identified [79], but how large their individual impact is and how they interrelate is still an open research question due to the large number of parameters involved and the sparseness of the data [85]. Honeybees, which belong to the flying insects, are major pollinators and vital for our ecosystem and economy. Therefore, it is important to understand what leads to their massive death.

BeeLivingSensor (BLS) is an ongoing citizen-science project led by the Informatics and Sustainability Research Group at the University of Zurich (UZH) and the Institute of Molecular Systems Biology at the Eidgenössische Technische Hochschule (ETH). Its goal is to understand how various factors such as biodiversity, weather, diseases, agriculture and other human activities influence the life and well-being of honeybee colonies. The main outcome of the BLS project will be an open, web-based platform where different stakeholders such as beekeepers, farmers and schools can help aggregate data about bees in a non-invasive manner. Cameras placed around beehives form the main source of data. They capture the movements and the interactions of bees and the pollen carried into the hive, which is correlated with data from the beehive itself (e.g. temperature, sound, weight, etc.) and with data from weather stations, agriculture and drones/satellites. The collected data will serve as a basis for researching the complex interplay of various parameters and taking effective measures based on these results.

### 1.1 Research Question

Biodiversity around a beehive is essential for the health of the bees. Good metrics for biodiversity are the pollen amount and variety. A typical bee colony requires approximately 17-34 kg pollen per year [19, 44] and a mixed pollen diet has been linked to improved bee health [3, 23, 22, 43]. Traditionally, pollen is collected with pollen traps or directly extracted from honey, which is counted and then analyzed under microscopes. However, this process is manual and expensive since only skilled experts can perform an exact analysis. Recently, automated approaches for counting pollen have been proposed [50]. The presented approaches, however, are typically intrusive in that they constrain the natural behaviour and movements of the bees. Furthermore, to the best of our knowledge, no automated systems have been presented and evaluated in the literature that capture the pollen diversity.

In this work, we describe a fully non-invasive system that records these pollen metrics (count and diversity) in an automated fashion, using video and flora data. The video footage serves for counting the pollen and capturing the pollen colors at the hive entrance, using a combination of Artificial Intelligence (AI) and heuristics. Since the botanical origin of a pollen usually cannot be determined based on its color alone, we leverage flora data to narrow down the possible plants from which the pollen may derive, allowing us to estimate the pollen diversity. For

example, knowing when plants flower and in which places they do so, allows us to determine whether the bees from a given hive pollinate a specific plant. In a nutshell, we aim to answer the following research questions:

- RQ1: How can the pollen count and diversity be recorded in a best possible automated, non-invasive way?
  - RQ1a: What is the performance of the fully end-to-end pipeline for counting pollen?
  - RQ1b: Which systems provide flora data that help capture the pollen diversity?
  - RQ1c: How can flora data be leveraged to measure pollen diversity?

For RQ1a, we construct a fully modularized and scalable end-to-end pipeline for counting pollen. To this end, we use existing code for most of the components of the pipeline that previous contributors have delivered to the BLS project [24, 25, 1]. On top of that, we implement two separate pollen counting algorithms. We then evaluate the performance of this final pipeline at two different levels: firstly, on a computational level by measuring the processing speed of the pipeline and secondly, on a qualitative level by testing how accurate the reported pollen counts are.

RQ1b and RQ1c, on the other hand, concern the pollen diversity, for which flora data sources form the basis. For RQ1b, we first perform a comparison of systems providing different flora data sources, using a set of evaluation criteria. The outcome of this comparison is a decision on which data sources to integrate into the BLS platform. For RQ1c, we describe an algorithm for matching plants based on time and location by leveraging these data sources. To evaluate the accuracy of this plant-matching algorithm, we compare its results to the outcome of a microscopy analysis.

## 2 Background

In this section, we introduce key terms in biology relevant for this master thesis and explain the role of bees and pollen. Afterwards, we present the BLS project, describing its goals.

### 2.1 Botanic Basics

Our earth is home to many living organisms. These organisms can be classified into taxonomic ranks according to shared traits. Figure 1 shows the taxonomy ranks with two example species based on the nomenclature codes *International Code of Nomenclature for algae, fungi, and plants* [84] and *International Code of Zoological Nomenclature* [73]. The highest rank is the *domain* which distinguishes between *Archaea*, *Bacteria* and *Eukarya* [92]. The lowest rank forms the *species*

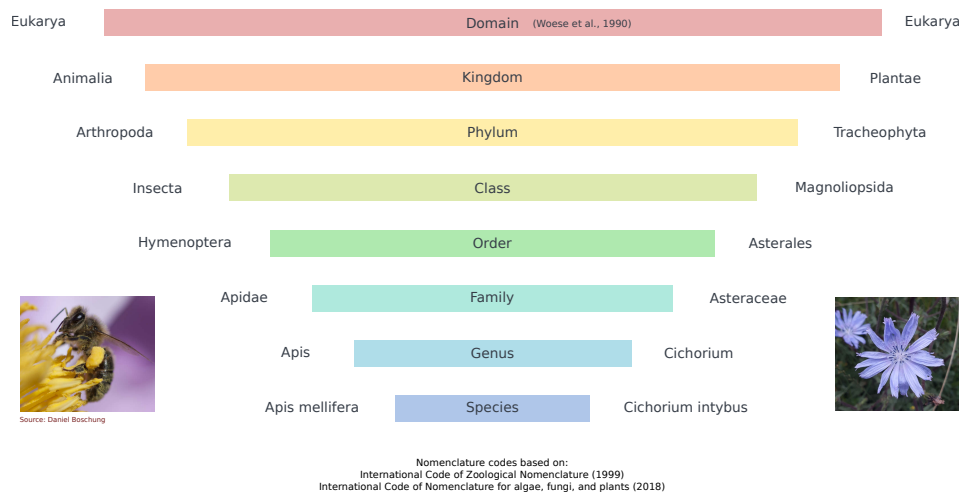


Figure 1: Taxonomic ranks with two example species *Apis mellifera* and *Cichorium intybus*.

and it is on that rank where our object of interest lies: the honeybee (*Apis mellifera*),<sup>1</sup> which feeds on plants such as *Cichorium intybus*.

The term *biodiversity* denotes the biological variation or variability of life on earth. This variation can be measured on different levels: within species, across species or across habitats. Biodiversity is important as it affects the services that an ecosystem provides to humanity such as crop yield and disease prevalence. Every species plays its role in an ecosystem and interacts with other species in complex ways. The pollination of plants represents a major part in this interplay.

Pollen is a powdery substance that seed plants such as trees, flowers and grasses produce to fertilize other plants of their kind. Wind and animals, especially insects and birds, help spread the pollen and therefore ensure the thriving of many plant species. Honeybees represent the most important pollinators. They fly from plant to plant to collect the pollen and thereby pollinate the plants along the way. Plants in turn provide fruits for birds and other animals. The pollination work of bees has a high economic value. One third of the plant products produced by agriculture relies on the pollination of the bees. According to [30], the value of pollination services is estimated at €153 billion.

The bees feed on pollen since they contain proteins which are required both for the production of royal jelly which they feed to the queen, larvae, drones and other bee workers [20], and their venom [49]. The nutritional composition of pollen differs greatly across plant species [75], both in terms of amino acids contained within the protein but also regarding lipids and vitamins. Thus, the nutritional value of the pollen is not purely defined by its protein content [21, 18, 59]. ‘Unbalanced’ pollen diets, e.g. when bees forage on certain monocultures [53],

<sup>1</sup>Henceforth, whenever we refer to ‘bees’, we mean the honeybee.



have been linked to a reduction of weight gain, longevity, royal jelly and venom [47, 93, 76, 12, 49, 59]. Noteworthy is that a mixed pollen diet is not necessarily superior to a monofloral diet. For example, a study found that a monofloral pollen diet comprised of *Asparagus sp.* or *Castanea sp.* stimulates the development of hypopharyngeal glands, which produce the royal jelly, just as well as a mixed pollen diet [59]. Nevertheless, a highly diverse diet is generally considered as more beneficial for bee health than monofloral diets [3, 23, 22, 43].

Many researchers have investigated how far bees fly to collect pollen, coming to different results. According to them, the mean foraging area of bees may range between 1km and 6km [88, 90, 8]. In this thesis, we assume a mean foraging distance of 3km for all our algorithms and analyses.

In summary, the bees benefit from a biodiverse environment and vice-versa contribute to more biodiversity.

## 2.2 BeeLivingSensor Project

BLS is an on-going citizen-science project led by the Institute of Molecular Systems Biology from the ETH and the Informatics and Sustainability Research Group at the UZH. The core idea of the project is to use the 30'000 to 50'000 honeybees from a colony as living sensors to track the biodiversity in the area as well as to measure the health of the colony.

News about the death of bee colonies have brought awareness about the importance of biodiversity to the general public. Over the past 30 years, the amount of flying insect biomass has declined by 75 percent in protected areas [33]. A multitude of factors have been identified that are held responsible for bee population losses and Colony Collapse Disorder (CCD) [79]:

- **Shortage of high-quality food (pollen and nectar):** This may be caused by urbanization [29], inappropriate agricultural practices (e.g. monocultures [45] and pesticides [31, 34, 26]) and weather/climate [80].
- **Pathogenes:** This includes parasitic infestations such as *Varroa* [74], bacterial infections such as foulbrood [28] and fungal infections such as nosemosis [63].
- **Mismanagement of apiaries:** inadequate feeding or incorrect pathogen control [41].

However, the exact impact of each individual factor and how these factors correlate and interplay is still unknown because it is difficult to conduct reproducible studies due to the number of factors that have to be controlled for and the sparseness of the data [85].

The BLS project aims to shed light on this intricate question by building a web-based platform that aggregates data from different sources in an automated, non-invasive fashion. The main data source form the cameras placed at the entrances

of beehives which capture the in- and outgoing bees. Using AI, we envision to detect bees, pollen and diseases in these videos. This enables us to automatically monitor bee traffic, foraging behavior and bee health. A second data source is the environmental data. This includes flora data which allow us to narrow down the potential plants that a pollen may derive from and compute the biodiversity and food supply around the hive. Furthermore, weather (temperature, humidity, etc.), hive sensor (weight, sound, etc.) and agricultural data (land use, pesticides, etc.) complete the picture. Thus, the platform has the following functions:

- The collection of data that serves as a basis for investigating the complex interrelationships of parameters (bee count, pollen input, weather, agricultural activities, etc.). This is why we place particular emphasis on data accessibility by making everything publicly available from the code to the raw data.
- The monitoring of beehives which allows beekeepers to track the biodiversity, food supply and the health of their bee colonies, and to take appropriate action when they decline.
- The connection of different stakeholders. For the installation of cameras and hive sensors, we need the cooperation of beekeepers. The creation of AI models requires enough labeled data which we obtain through the help of schools and citizen scientists. The same stakeholders also help gather plant observation data that botanists curate. Farmers can provide us data on their agricultural activities. Finally, researchers analyze the data.

Figure 2 shows an overview of the data that the platform plans to aggregate, the stakeholders that are involved in this and the required equipment.

We strive to make the data acquisition as automated as possible. However, some of the data is crowd-sourced and is therefore manual in nature, for instance, the labeling of images for the creation of new AI models or, currently, the observation of plants. The rest of the data can be gathered in an automated fashion, such as fetching weather station and hive sensor data from Application Programming Interfaces (APIs) as well as counting the bees and pollen flown into the hive (once models have been trained).

Furthermore, all our methods are non-invasive. Being a keystone species, bees have been extensively studied for a long time. Most of this research is performed in laboratories on individual bees or in the context of beekeeping. However, all too often, it is forgotten that bees live in colonies and that they adapt to the interventions of beekeepers. Therefore, the studied bees are often not behaving naturally. In an initial phase, BLS will also collect data from managed bee hives. However, the future vision is to monitor bees in a natural setting such as in tree cavities, using affordable technology like solar-powered cameras and 5G for the data transfer.

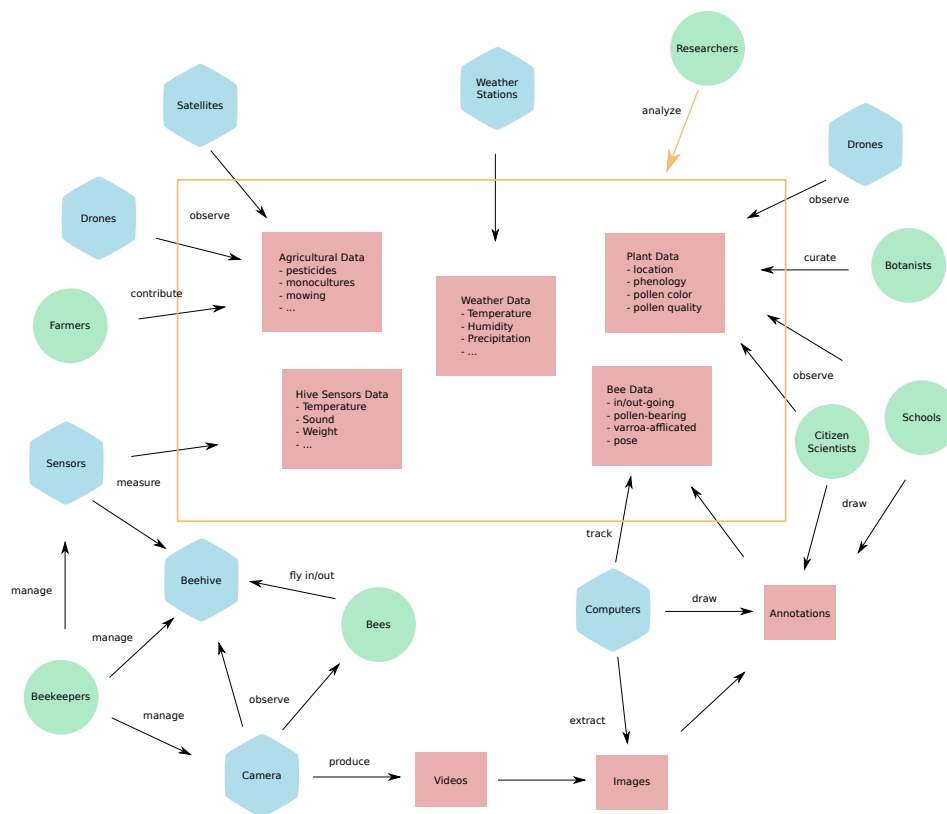


Figure 2: Data being aggregated by the platform. The green circles are the stakeholders, the blue polygons are the equipment (including software and hardware) and the red rectangles are the data.

### 3 Previous Technical Contributions to BeeLivingSensor

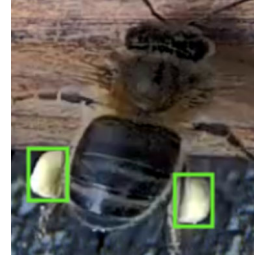
In total, four separate student groups have worked on the project over the past years, three of them on the creation of AI models and trackers [24, 25, 1], and one building the platform itself [42].

#### 3.1 AI Models & Trackers

BLS leverages AI to automate the counting of bees and pollen. To begin with, this requires bee and pollen detection models which, graphically speaking, draw rectangles (‘bounding boxes’) over bees and pollen in images as shown in Figure 3.



(a) Bee annotations [25, 1]



(b) Pollen annotations [24]

Figure 3: Annotations over bees and pollen

A video is a sequence of images (‘frames’) and the same bee typically appears in several images. To trace the movements of an individual bee, tracking algorithms assign IDs to bees to associate the same bees across multiple images using metrics such as Euclidean distance or Intersection over Union (IoU). The complete flow of steps is shown in Figure 4.

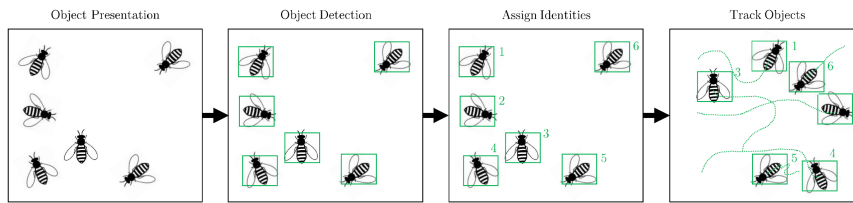


Figure 4: Steps for tracking bees [25]

Finally, to count the in- and outgoing bees, we observe whether a bee disappears within or outside the hive entrance by checking whether the bee center point lies inside or outside the entrance’s area (point-in-polygon test).

A first stab at detecting and counting bees and pollen in video images was undertaken by a student group from ETH in the summer of 2019 [24]. They built an end-to-end pipeline that accepts a video and outputs the bee and pollen counts.

They used the `TENSORFLOW` Object Detection API [35] in combination with the Faster-RCNN model [70] to train a bee detection model. For detecting pollen, they trained a custom convolutional network using the `PYTORCH` framework [62]. They evaluated their models on four different beehives. The bee models were evaluated with the IoU score metric, achieving performances between 63% and 80%. For the pollen models, they used the f1-score as an evaluation metric, achieving 84% and 88% in performance. For recording the bee traffic, they implemented a bee tracker using the IoU metric, defining the hive entrance as a region of pixels to decide whether a bee flies in or out.

In February 2020, a student group from the UZH continued this work [25]. Their main contributions were to experiment with other bee model algorithms and evaluate the models under different settings such lower image resolutions, light conditions and camera angle views. Furthermore, they labeled an additional 3000 images. They used the YOLO framework [69] for training the bee models, which is known to be faster than other frameworks. In total, they evaluated their bee models on ten hives using precision and recall as an evaluation metric. They reached a precision and recall score of over 90% in nine of the hives for the top models, which were either trained on the complete set of hive images or on images from a cluster of hives (e.g. according to entrance shape). Moreover, they implemented a bee tracker combining the IoU and Euclidean distance for matching bees in images. The pollen detection and counting mechanism of the previous group was not integrated.

In September 2020, another group from the ETH took up the work from the previous two groups. They trained a unified bee model using the same model architecture as the previous group, but using both enhanced train and dev set images for the training. This way, they outperformed all previous models, except for one (on average by 5% percent). For the pollen models, they labeled more bee images with pollen. They also trained the models with the YOLO framework, reaching a mean Average Precision (mAP) of up to 88%. Another contribution was to test out different model and image input sizes to assess the trade-off between processing speed and accuracy. For the pollen detection, they experimented with two different model sizes, the smaller model reaching a Frames Per Second (FPS) that is ten times higher than that of the larger model and a mAP that is only marginally worse (-2.64%). For the bee detection, they experimented with three different image input dimensions (224x224, 416x416, 608x608). The medium model only performed slightly worse (-1.56%) having a 30% faster inference time, while the small model was significantly worse (-10.64%). For tracking the bees, they used the Simple Online and Realtime Tracking (SORT) algorithm [10], achieving a better MOTA [9] score on the enhanced image material (+1.6%), and a worse score on the original image material (-0.9%). The bee and pollen counting mechanism itself was not implemented.

### 3.2 First Prototype of Platform

In June 2020, another group started building the actual web-based platform [42] which is hosted on AZURE. Since the platform processes big data (~64GB of video per hive per day), it runs on a KUBERNETES cluster to ensure scalability. The full architecture is shown in Figure 5.

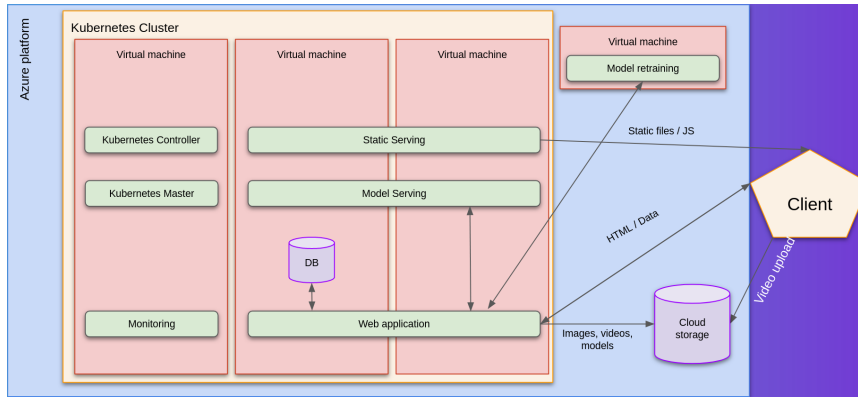


Figure 5: Architecture of first platform prototype [42]

The data is stored in two ways:

- **Cloud Storage:** This contains binary files such as videos, images and model weights.
- **Relational Database:** The remaining data is stored in a PostGIS database.

The platform consists of four services

- **WEBAPP:** Main service which is responsible for Create, Read, Update and Delete (CRUD) operations in the PostGIS database as well as the User Interface (UI). Most of the functionality is implemented there.
- **MODEL-SERVICE:** Performs the media (image and video) processing such as the extraction of images, the pass through the AI models and the tracking. At this point, only the bee detection from the second group [25] was integrated. Pollen detection as well as bee counting were missing as this work was still in progress by the other two AI groups [25, 1].
- **STATIC-SERVICE:** Serves static files like JAVASCRIPT and Cascading Style Sheets (CSS) files.
- **RETRAINING-SERVICE:** for training new AI models. Due to the high costs of Graphics Processing Units (GPUs), this service is manually started up by the admins and shut down when the training is finished.

The platform offers user, group, apiary, hive and sensor management with the usual CRUD operations. It allows the upload of media for a hive which are processed by the MODEL-SERVICE. It also features a labeling tool for annotating new images and retrieves weather data from an external API.

## 4 Performance of Pollen Counting

This section answers RQ1a:

*What is the performance of the fully end-to-end pipeline for counting pollen?*

As described in Section 3.1, several groups have contributed to the detection and tracking of bee and pollen, using a variety of frameworks and techniques. As the groups produced different artifacts for different components, the first task was to find out which artifacts for which component to use. Figure 6 shows the most important components of the pipeline and who made the corresponding contribution.

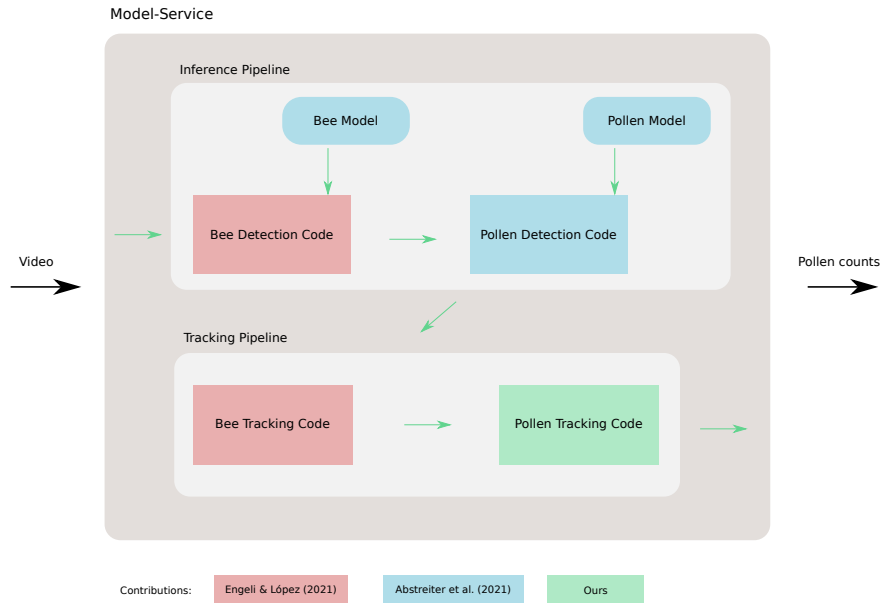


Figure 6: Components of the MODEL-SERVICE

The second task was then to modularize each component and connect them to each other, placing the data in appropriate data structures. For the last component in the chain – the pollen tracking – we devised two algorithms for testing pollen counting. This fully end-to-end pipeline then runs as part of a service, called the

MODEL-SERVICE, which accepts a video as an input and produces pollen counts (along with the detected pollen colors) as an output. Thus, the first contribution of this work was to establish this fully end-to-end pipeline in the first place.

The second contribution was to optimize the MODEL-SERVICE and ensure its scalability, with the overall goal of making it process videos faster. For this, we applied multithreading and multiprocessing mechanisms, and leveraged GPU processing. To achieve scalability, we dockerized the MODEL-SERVICE and created configurations for it to be run on a KUBERNETES cluster.

Finally, we evaluated the MODEL-SERVICE, firstly, by comparing its reported pollen counts with those from a manual counting and secondly, by measuring its processing speed across several machine types.

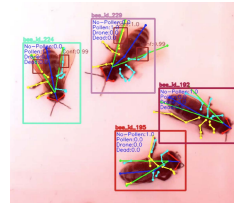
#### 4.1 Related Work

Automated bee monitoring systems have been in existence for a couple of years. They leverage a variety of techniques to monitor bees:

- APIC.AI [50] use a camera attached to the hive entrance to capture bee traffic, as seen in Figure 7a. Bee detection models and trackers are run on edge devices which compute the number of bees returning and leaving a hive [81]. The cropped bee images are then transferred to the cloud and analyzed for health-related insights using a multitask convolutional neural network. The model recognizes genus (bees, wasps, bumblebees and hornets), pollen, pose and bee type (Worker bees with pollen, worker bees without pollen, drones and dead bees), as shown in Figure 26b. Their devices have been deployed on ~50 beehives.



(a) Hive entrance [81]



(b) Detection and classification of bees [50]

Figure 7: System of APIC.AI

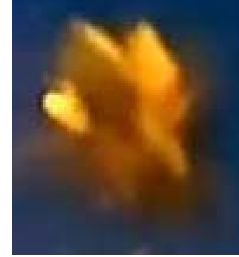
- [5] also use video footage to detect bees and pollen. They designed a box with an integrated camera (Figure 8a) and an edge device that is mounted on the front side of the beehive. The edge device detects bees and classifies them into whether they bear pollen or not (Figure 8b), using background subtraction and nearest mean classifiers based on color features. To reduce the adverse effects of shadows, bees are filmed inside the box using only



the light that emanates from the beehive opening. In the box, bees have to crawl a distance of 11 cm. Bee and pollen counting algorithms have not been implemented.



(a) Camera view [5]



(b) Bee with pollen detection [5]

Figure 8: System of [5]

- EASY BEE COUNTER [36] created gates in front of the entrance in which the bees are led through. In these gates, LEDs emit infrared light. If a bee is present, this light is reflected back to a sensor which triggers. When the bee is not present, the light is absorbed into the black surface. Figure 9 shows the gates in which the bees go in and out. Pollen counting is not implemented.



Figure 9: Bee counting with optical sensors [36]

- EYESONHIVES [83] set up cameras in front of the hive like our system. Video material is analyzed on an edge device which computes the average number of bees per second. This way, the number of bees flying in and out is not explicitly counted, but it allows for swarming and robbing detection. The count data along with highlight videos are uploaded to the cloud. Pollen detection is not implemented.

Except for EYESONHIVES, the systems differ from our proposed system in that they are intrusive and therefore disturb the natural behavior of bees. They perform modifications to the beehive [50, 5, 36] which itself can be considered an

interference as the bees have to get accustomed to the new entrance. More invasive are the creation of channels in which the bees can only go into one direction [50, 36], or the usage of artificial lighting [50].

A side-effect of our non-invasive approach is that our system can be deployed on all types of hives, including natural housing such as tree cavities, since no modifications to the beehive are necessary. This is not true for all aforementioned systems, e.g. the system of APIC.AI only works on ZANDER beehives. Furthermore, our system requires only commonly available equipment such as a solar-powered camera and a communication device that any beekeeper can install. Thus, no installation of special edge-devices or other hardware is required.

Except for APIC.AI, the other systems also do not observe foraging activity, and none of them capture the pollen diversity.

## 4.2 Implementation

The first prototype of the platform already featured a MODEL-SERVICE [42]. This initial version only extracted images from videos and ran a bee model over the images. Thus, no pollen detection nor bee and pollen tracking was included. Moreover, the MODEL-SERVICE had low performance as posting a handful of small videos would already render the service unresponsive. In the following, we explain how we modified and extended the service and how we made it more performant as well as scalable. Thereafter, we describe how we evaluated the performance of this end-to-end pipeline, both in terms of speed and accuracy.

### 4.2.1 Model-Service Abstractions

The previous version of the MODEL-SERVICE consisted of a range of methods and global variables polluting the global namespace. To add encapsulation and information hiding, we devised abstractions and packed methods and data into corresponding classes. We defined classes for processing media like images and videos (`MediaProcessor`), for models that create annotations (`ObjectDetector`) and for trackers for the number of bees that fly in and out (`BeeTracker`) as well as the number of pollen (`PollenTracker`). The detector classes can be extended in the future to allow for new types of model architectures.<sup>2</sup> Furthermore, we defined data classes, as shown in Figure 10, for storing the output of models and trackers, which we will reference throughout the next sections.

### 4.2.2 Media Data Flow

A second change was made to the way the MODEL-SERVICE interacts with the WE-BAPP. To understand why this change was necessary is to note that in the future, every hive will upload a day's worth of video material. A 15-minute video, for instance, is often more than one GB large, generating up to half a million images,

<sup>2</sup>At the moment, we only provide a detector with YOLO [69] models.

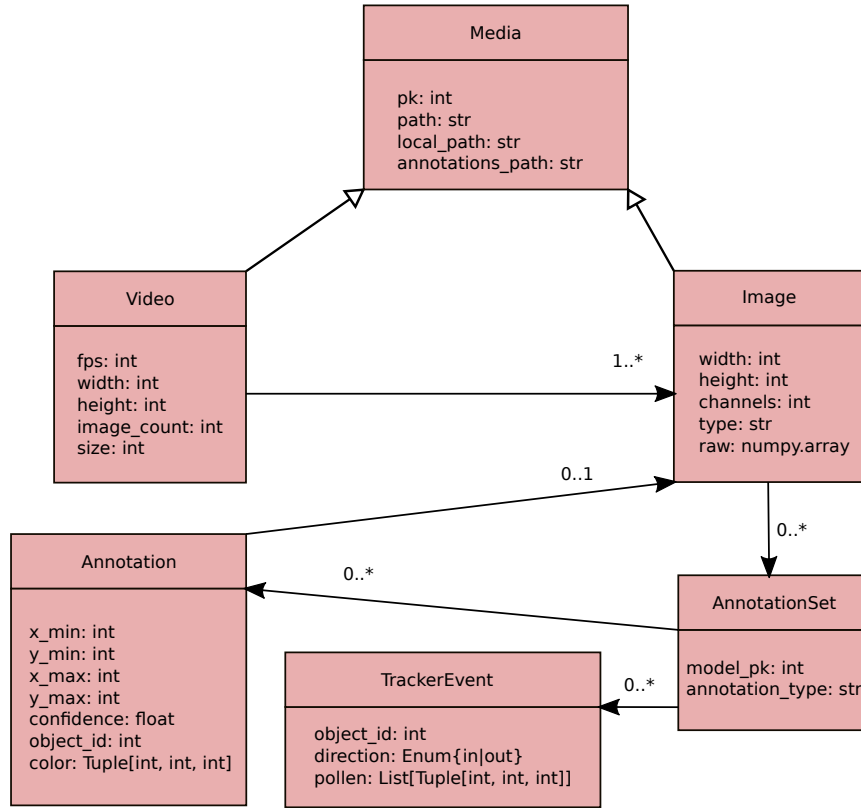


Figure 10: Class diagram for the data in the MODEL-SERVICE.

and therefore its processing can take hours depending on the GPU machine. Previously, the WEBAPP would send a request to the MODEL-SERVICE, waiting for its response which both contained the images along with the annotations. The WEBAPP would then take care of uploading the images to the cloud storage as well as committing the annotation data to the database. This approach had serious flaws:

- Two uploads are necessary, first from the MODEL-SERVICE to the WEBAPP and then from the WEBAPP to the cloud storage, which increases total upload time as well as bandwidth demands.
- Most servers have a time-out limit, i.e. the server will abort the processing of a request from a client after a given time. This is to prevent a client from waiting endlessly for the server to respond, blocking Input/Output (I/O) along the way. The servers of the MODEL-SERVICE and WEBAPP also time out after a few minutes per default, which in both cases, was too short. The MODEL-SERVICE undertakes heavy image processing on the GPU which can take several hours. The WEBAPP, on the other hand, performs many I/O operations by inserting images and annotations into the database, which also typically exceeds the time-out threshold.

- Most servers also impose body size and file upload limits in requests, including ours. By default, these limits are in the range of a couple of MB and serve as a countermeasure against denial-of-service attacks. Again, this does not meet our demands since the results of the MODEL-SERVICE can be several hundred MB in size.
- The WEBAPP is generally built with less performance in mind compared to the MODEL-SERVICE. It has limited disk space as well as limited memory, especially given that it is often scheduled on less powerful machines as opposed to the MODEL-SERVICE that runs on GPU machines with plenty of memory.

To overcome such limitations, the data flow was changed as follows, illustrated in Figure 11: As before, the WEBAPP uploads the media (video or image) to cloud storage (1). It then sends media and model metadata, namely their primary keys and their blob paths, to the MODEL-SERVICE (2). The MODEL-SERVICE downloads the media and the models from cloud storage as specified by the blob paths, extracts the images in the case of videos and runs the models over the images (3). Unlike the previous version, the images are directly uploaded to cloud storage (4), rather than sent to the WEBAPP and then uploaded by it. When the media processing is finished, the MODEL-SERVICE also uploads annotation data as a JavaScript Object Notation (JSON) file to cloud storage (5). The MODEL-SERVICE then notifies the WEBAPP about its completion, returning the blob path of this file (6). Finally, the WEBAPP downloads and reads this file and adds all data contained within it to the database. To circumvent any time-outs on servers, the processing of media is moved into separate threads and processes.

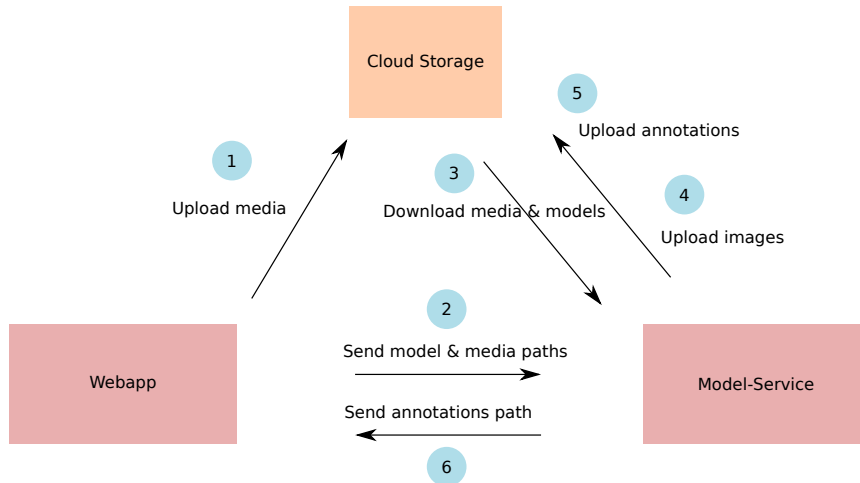


Figure 11: Media data flow

### 4.2.3 Media Process Scheduler

Previously, the `MODEL-SERVICE` would become unresponsive when too many videos are sent to it. This is because the processing of a single video requires at least 2 GB of memory. Since videos are typically sent in chunks of 5-15 minutes, 100-300 requests per hive and per day are common but will lead to a denial of service, especially when all videos are uploaded at the same time. To handle the memory consumption, several solutions are possible. On the infrastructure level, one can increase the memory capacity of a Virtual Machine (VM). However, assuming that 10 hives regularly upload videos, we arrive at 1000-3000 requests per day. This would require 2000-6000 GB of memory, if all videos were to be uploaded at the same time.

On the application level, one can control how many requests are processed concurrently. This method has the advantage that it is infrastructure-independent, i.e. the processing of media will work on any kind of computer (as long as this computer has a minimum of 2 GB memory). However, this comes at the price of longer processing times. As the `CELERY` framework is used within the `MODEL-SERVICE` (cf. Section 4.2.9), the number of Central Processing Units (CPUs) determines how many videos may be processed in parallel. Strictly speaking, this approach may still lead to memory exhaustion when the ratio of memory size and number of CPUs falls below a specific threshold. However, given that almost all GPU machines have a memory size that is a multiple of the number of CPUs, this scenario is unlikely to surface. Using a queue as a data structure, the media are processed according to a fairness principle: media processes waiting for the longest time are chosen first.

### 4.2.4 Model Inference Pipeline

The model inference pipeline is illustrated in Figure 12 and works as follows: first, we split the video into hive images (1), which we run through the bee model, giving us annotations of bees (2). Then, we crop the bees out of the hive image, using the bee annotations to obtain bee images (3). The last step is to run the pollen model over all the bee images to obtain the pollen annotations (4).

For the model inference (bee and pollen detection), we used the code from the second AI group [25] as this code was already integrated into the first prototype of the platform [42]. Furthermore, their code was written in `PYTHON` which is also the language of the platform, while the inference code of the third group [1] was available as shell scripts. For the image cropping, we used the code from the third group [1]. The three steps – bee detection, cropping and pollen detection – were not properly connected to each other and therefore, we implemented an end-to-end pipeline which takes an hive image as an input and outputs annotations for bee and pollen in one go. For this purpose, we adapted the code of the AI groups, removing hard-coded values such as paths, extracting the steps into functions of the media processors and placing output annotation data in data classes.

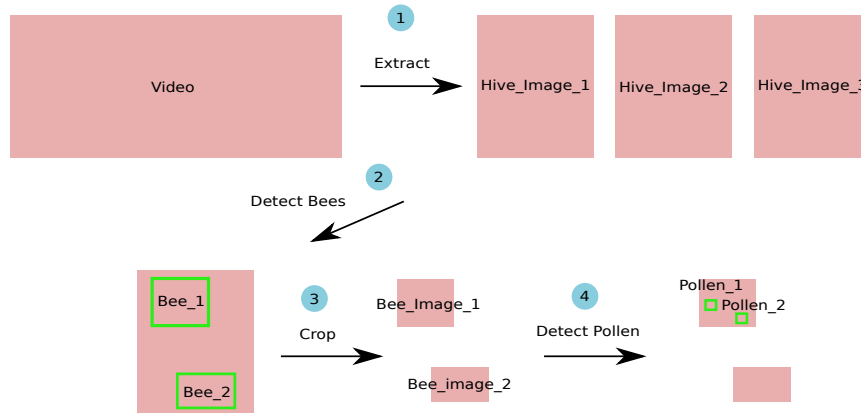


Figure 12: Model Inference Pipeline

#### 4.2.5 Pollen Color Extraction

The variety of pollen colors is already a good approximation of the biodiversity around a hive. Determining the true color of pollen on images is, however, challenging due to different lighting conditions caused by shadows, weather, daytime and the camera settings. To alleviate this problem, we can place a grey card in the camera view and correct all colors of the image using the known color of the card as a reference, as shown in Figure 13.



Figure 13: Grey card at the hive entrance for color correction [1]. Source: Daniel Boschung

The third AI group experimented with this kind of color balancing [1]. They used hierarchical clustering to group the pollen colors and found that color correction does not increase the homogeneity score significantly (94% vs. 96.6%), i.e. in both cases, almost all extracted colors can be assigned to exactly one color group.

A limitation of their work is that most of the pollen colors were simulated with pins rather than real pollen. Generally, it is to be assumed that the extraction of colors will never be precise with this technique. As shown in Figure 13, the card itself has different colors ranging from white to grey, which adds an additional layer of uncertainty. Moreover, adopting this approach implies a greater implementation effort because the location of the card in the image has to be known. At the moment, the coordinates are hard-coded and must be manually changed in the code. To fully automate this, we either need models that can detect such cards or we have to ask beekeepers to annotate them in sample images.

For these reasons, we kept the color extraction algorithm simple: First, the pollen image is cropped based on the pollen annotation coordinates. An additional 40% is then removed from this pollen image so that no background is visible (given that pollen is not rectangular). Finally, the average color in this cropped image is computed and stored as an RGB value.

#### 4.2.6 Bee Tracking

As already mentioned in Section 3.1, three AI groups created three different bee trackers. Based on the reports and the code, we decided to use the bee tracker of the second group [25] as they performed more extensive experiments with the trackers than the other two groups, including the counting mechanisms which the third group did not implement.

Like the model inference code (cf. Section 4.2.4), we also modularized the tracking code, storing the output in data classes. We stored the information in the following way: Firstly, the tracker allocates IDs to the bees and accordingly each bee `Annotation` object stores the ID of the bee (`object_id`). Secondly, bees move across images, and the tracker can register certain key movements which are stored as `TrackerEvent` objects. At the moment, only two types of events are recognized: bees flying into and out of the hive which we codified as a `direction` variable in the `TrackerEvent` class.

To determine flight direction, the entrance coordinates have to be known in advance. In the original code, these coordinates were hard-coded and had to be manually changed for each hive. To make the coordinates configurable, we added an additional parameter for entrance coordinates to the endpoint accepting media process requests. The previous code supported coordinates of rectangles and circles. We extended this to support coordinates of polygons. Frequently, rectangles cannot be drawn precisely around the entrance, especially when footage is taken from a side angle, and therefore polygon shapes are better suited in this case. Furthermore, we added an parameter to offset the coordinates by some value as this influences the tracking, too (cf. Section 4.3).

In the `WEBAPP`, we created a new annotation type `entrance` so that users can also label entrances in images. Whenever a beekeeper creates a new hive, a labeling task for the entrance is automatically opened. When new videos are uploaded, the coordinates of these labels are automatically passed. In case none are available



yet, the `MODEL-SERVICE` generates coordinates by using one fourth of the image as the entrance.

There are some remaining issues with the bee tracker. Firstly, the tracker does not scale well. The speed for processing images steadily decreases with the number of images being processed. Secondly, the tracker crashes when there are no annotations. As we do not understand the bee tracker in all its details, we produced a workaround for both cases by occasionally resetting the tracker during the processing.

#### 4.2.7 Pollen Tracking

So far, we know how many bees enter and leave the hive. However, we also want know the number of pollen that is brought into the hive. The easiest strategy to count pollen is to use the last image in which an entering bee was detected and check whether pollen was detected on that image as well. However, this approach is less robust as the bee or pollen detection might exactly fail in this very image, for example when the bee is positioned in a way that the pollen is not visible.

For this reason, we implemented two different algorithms for evaluation. The first algorithm works as follows:

1. For each bee that flies in, extract its ID.
2. Get all images in which this bee was seen (using its ID).
3. Extract the number of pollen detected in the images.
4. Compute the average number and color of the pollen detected across all images in which the bee occurs.

This approach is very similar to the one described by the first AI group [24]. They checked whether half of the images for a specific bee contain (at least one) pollen. In our algorithm, we average the pollen counts over all images, which in most cases leads to the same result. The main difference is that they classify images as bearing pollen or not, while our algorithm precisely counts the pollen (0, 1 or 2) and also extracts pollen colors for them. As already noted by the group, this strategy has its pitfalls. Generally, the pollen counts reported by the `MODEL-SERVICE` were low at this point. This is because the bee tracker has a tendency to confuse bees when bees walk over each other as is often the case when there are many bees present at the entrance. This can cause the tracker to assign the ID to the wrong bee, possibly to one that does not carry pollen and therefore skewing the number of images where pollen was detected. To alleviate this problem, the first group proposed to use pose detection to recognize in which direction bees face.

Training a new model for detecting bee pose was out of the scope of this thesis. Instead, we came up with an alternative algorithm for counting bees. The main assumptions of it are:



- Bees with pollen eventually fly inside the hive<sup>3</sup>
- Bee ID switches typically happen when bees are bunched together, which is usually the case right before the entrance opening, i.e. when the bee is about to enter the hive.

The algorithm works as follows:

1. For each unique bee (i.e. unique bee ID), count the number of images in which the bee occurs and in which pollen was also detected.
2. If pollen was detected in a sufficient number of images (as defined by some threshold), the unique bee is marked as bearing pollen.
3. Average pollen counts and colors for this marked bee.

This algorithm reacts less sensitive to bee ID switches because when a bee with pollen enters the hive and a bee ID switch happens right before entering, the pollen is still counted as the pollen was recorded in sufficient numbers with the previous ID. This algorithm obviously fails when the same bee was registered with pollen in sufficient images for more than one ID. In this case, too many pollen pellets are counted.

#### 4.2.8 GPU Processing on Kubernetes

The previous model inference code only supported image processing with CPUs. Since GPUs can accelerate this process by a large margin, we deployed our `MODEL-SERVICE` on GPU machines.

Switching to GPU machines required only minor changes on the inference code itself, namely instructing the model to use CUDA [54] and selecting a random GPU device so as to distribute the work when there are multiple GPUs on a machine. However, they required major adjustments on the machine and container configuration. Firstly, the GPU node has to be configured by installing CUDA drivers and NVIDIA DOCKER [58]. These dependencies are the prerequisite for running the NVIDIA DOCKER images [55] which allow us to leverage GPUs within containers. Using the base image of NVIDIA, we had to completely revamp our original `Dockerfile` which used the FASTAPI [67] image as a base image which in turn used other base images (like GUNICORN's [16] and PYTHON's). To include the same functionality as before, we manually copy-pasted the layers from these FASTAPI images into the new `Dockerfile`. Furthermore, we removed the installation of OPENCV [60] via PIP as the binaries are shipped without GPU support. Instead, OPENCV is built from source with CUDA, cuDNN [56] and PYTHON bindings.

<sup>3</sup>In practice, bees may come out of an entrance with pollen again, but this only happens occasionally.

To make the GPU accessible from within pods in KUBERNETES, further additions were necessary. Firstly, it must be noted that support for GPU access from containers is quite recent. The `--gpu` flag in DOCKER was introduced in 2019 in version 19.03.0. DOCKER COMPOSE followed suit in early 2021 with version v1.28.0+ enabling access to GPUs. However, managing GPUs within KUBERNETES is still an experimental feature [46]. Currently, the way to access them is to use device plugins which are vendor-specific (AMD or NVIDIA). We used the NVIDIA plugin [57] which is a Daemonset, configuring the runtime accordingly. There are some limitations when using these plugins. While pods can request one or more GPUs, they cannot share them with other pods, or request a fraction of it as in the case of CPUs. This also means that when only one GPU is available, new versions of the MODEL-SERVICE cannot be rolled out gradually but the old deployment has to be deleted first before the new version can be deployed.

#### 4.2.9 Parallelization with Threads and Celery

Typically, the FPS reported for models only refers to the inference duration of an image [32], i.e. running the image through the model. However, between inferences, there are intermediate steps such as downloading videos from cloud storage, reading images from disk into memory, uploading images to cloud storage, saving annotations in data structures, converting data and cropping images, which in their entirety increase processing time substantially.

For example, including image reads decreases FPS by almost 60% (=19 frames) for the bee models on a *NVIDIA Tesla P100* machine. For the pollen models, this effect was much lower (~10%) because the image size is small. To avoid unnecessary image reads, we keep the image in memory for the complete duration of its processing rather than writing the image to disk as in the previous version. Another optimization was to load the model only once rather than for each image separately in the video. Most bee models are around 200MB in size and loading them takes some time.

As the processing speed was still unsatisfactory at this point, despite those optimizations, we used the PYINSTRUMENT [72] profiler for tracking down which methods took the longest for processing. This revealed the following insights evident from Figure 14.

- Uploading images (`save_image`) consumes almost 3/4 of the total processing time.
- Parsing the output of the network (`parse_network_output`) adds at least an equal amount of processing time as running the image through the model itself (`pass_through_network`)
- Annotating pollen (`add_pollen_annotation_set`) consumes almost as much time as annotating bees (`add_bee_annotation_set`).

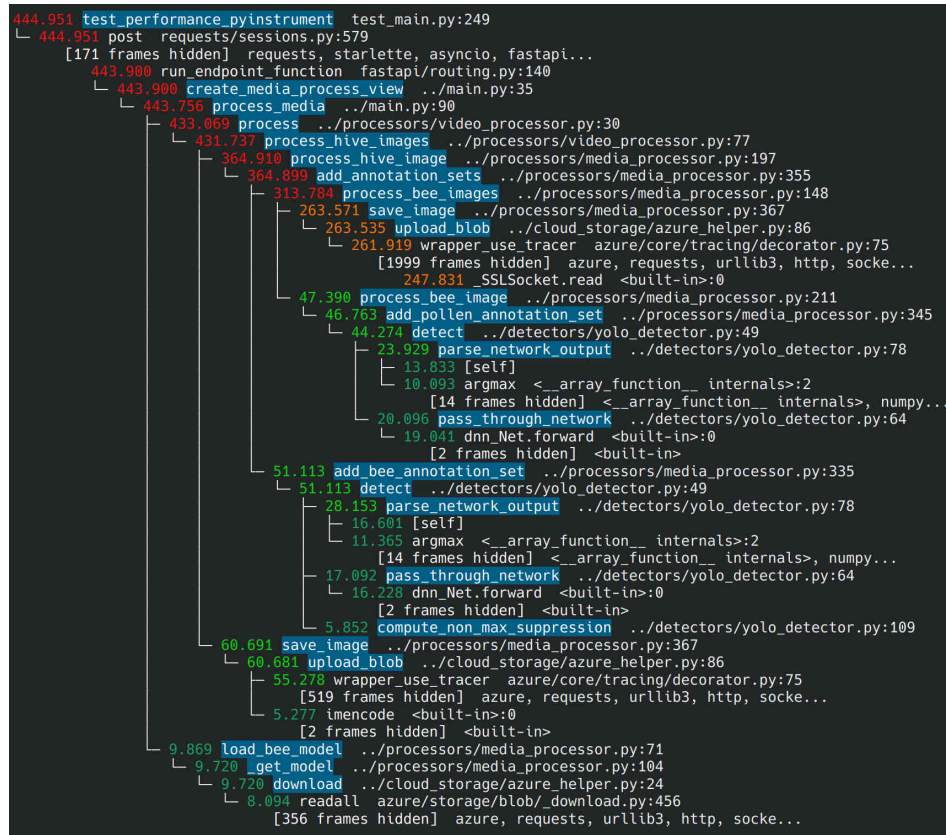


Figure 14: First profiling of MODEL-SERVICE. Functions are highlighted in blue and numbers left to them denote the cumulative runtimes.

In the first version of the MODEL-SERVICE, all steps were executed sequentially. However, we can make use of multithreading and multiprocessing to execute the steps in parallel. This parallelism is possible when we have several CPUs at our disposal as well as when there is a mix of I/O and CPU-bound operations. Conveniently, the MODEL-SERVICE has parts that are I/O-bound such as the image reads/writes and the downloads/uploads, but also parts that are CPU-bound such as parsing the output of the model.

PYTHON offers several modules for controlling concurrency and parallelism such as MULTIPROCESSING, THREADING and ASYNCIO. Because of the Global Interpreter Lock, ASYNCIO and THREADING can only run on a single CPU and therefore are only suitable when the program spends considerable time performing I/O operations.

As the upload is the major bottleneck, we moved the uploads into separate threads which may run at the same time as the CPU-bound computations. This yielded a significant improvement as shown in Figure 15.

As can be seen, the MODEL-SERVICE does not spend time waiting for uploads

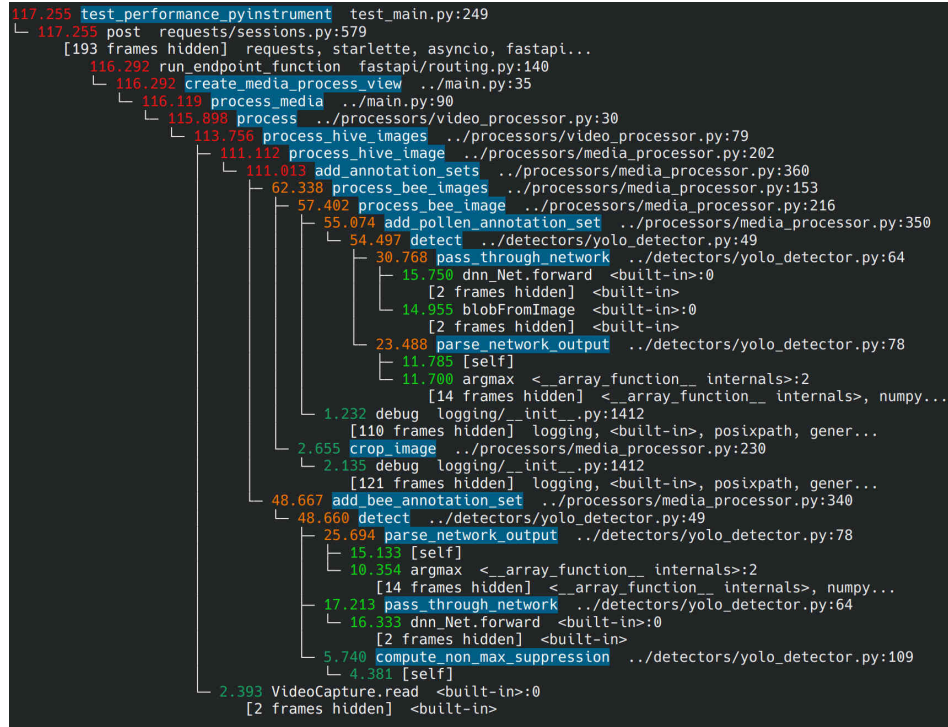


Figure 15: Second profiling of MODEL-SERVICE

anymore, reducing total processing by ~75%. However, the upload threads have visible side-effects: calling the `blobFromImage()` function for hive images now takes considerably more time than before. This is because this is also an I/O-bound operation that accordingly contends with the upload operations for I/O.

Another bottleneck, albeit a smaller one than the upload, represents the parsing of network outputs which now accounts for ~40% of the processing time. Since this is a CPU-bound operation, it cannot be optimized through threading. Therefore, we tried to parallelize the parsing across multiple processes. However, the runtime became much worse through this. Presumably, the creation of processes creates an overhead such that it cancels out the benefits of distributing the work.

Another possible way of optimization is to parallelize the processing of each hive image. However, multithreading and multiprocessing generally have the disadvantage of making the code less readable and less maintainable as the code is more prone to race conditions which are hard to debug. For example, parallelizing the hive image processing required a lock on accessing the model as well as control over when images may be deleted (it may not be deleted during the upload but also not during its processing). We experimented with multiple threads and processes. Predictably, multithreading not only deteriorated runtime, it also obliterated profiling information, only showing that most time is spent on acquiring locks. On the other hand, multi-processing using the `ProcessPoolExecutor` from

the `CONCURRENT` module<sup>4</sup> did not work as it raised an error stating that `OPENCV` objects cannot be pickled. This error has been reported on Q&A channels but does not appear to have a fix [6, 78]. For this reason, we could not perform further optimizations on the pipeline itself.

However, on the service level, we can achieve additional parallelism. We implemented the `MODEL-SERVICE` with the `FASTAPI` framework [67] which in turn uses `GUNICORN` [16] as a server. By default, `GUNICORN` starts up the same number of workers (aka processes) as there are number of CPUs. This means that multiple requests may be handled across different CPUs, i.e. multiple media processes may run in parallel. However, there is no guarantee that `GUNICORN` distributes requests across multiple workers because the load balancer might schedule several requests on the same worker process, which practically happened more often than not. Presumably, the load balancer serves for evenly distributing the amount of requests rather than distributing them based on CPU/GPU workloads. For this reason, we decided to use the `CELERY` framework [15] for explicitly scheduling media processes across CPUs. Figure displays how the media processing is orchestrated in `CELERY`.

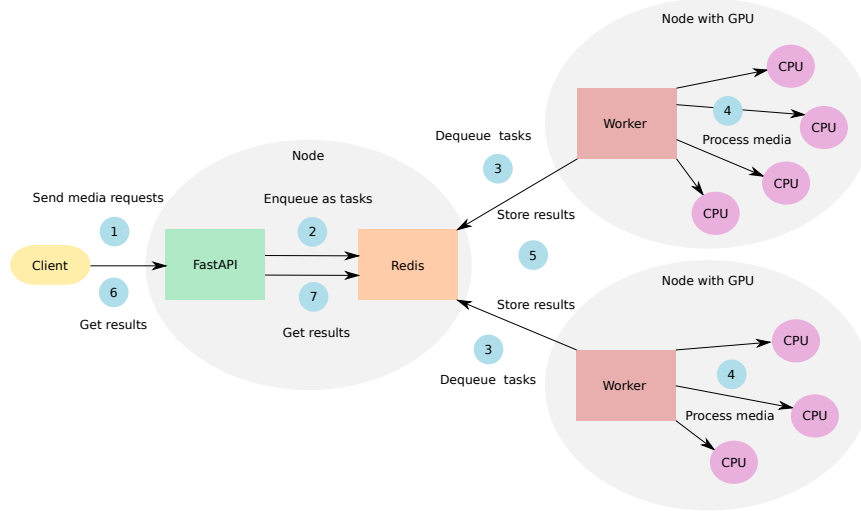


Figure 16: Distributed parallel processing with `CELERY`

Firstly, we have the client that sends multiple media requests to the `MODEL-SERVICE` (1). The `FASTAPI` service accepts these requests and dispatches them as tasks to the `REDIS` service, which puts the tasks in a queue (2). This way, older tasks are processed first, following the First-In-First-Out (FIFO) principle. `CELERY` workers constantly monitor this queue for new tasks. When a worker is deployed, it spawns a certain number of child processes (defaulting to the number of CPUs). Each such child process can process media. When a child process is idle – either

<sup>4</sup>The `CONCURRENT` module internally uses the `MULTIPROCESSING` module within the `ProcessPoolExecutor`.

because it has not processed media yet or it has finished processing media – the worker dequeues a task from REDIS (3) and starts that task in the child process (4). The task that they execute is the whole pipeline that we have outlined in the previous sections. Finally, the results of the process are again stored on REDIS (5). The client can retrieve this result from the FASTAPI service (6) which in turn requests it from REDIS (7).

Using CELERY gives us high scalability as we can not only distribute tasks across CPUs on the same machine, but also on different machines. Hence, if we add more hives, we only need to spin up more GPU machines and start a CELERY worker instance on each of them. For example, in Figure 16, we have two machines with 4 and 3 CPUs, respectively. Each machine has a GPU which the CPU processes can access (not explicitly shown here) and a worker being deployed on it. This means, on the first machine 4 media can be processed in parallel, while on machine two, 3 of them can be. Hence, in total 7 media can be processed in parallel.

### 4.3 Accuracy Evaluation

In this section, we evaluate whether the pollen tracker of the MODEL-SERVICE delivers accurate pollen counts. We identified the following factors that may influence the counting mechanism:

- **FPS:** As noted by the second AI group [25], the bee trackers for counting bees work more precisely on videos that have a higher FPS. However, a higher FPS also translates into longer runtimes. Thus, a good trade-off between speed and accuracy has to be established.
- **Camera angle:** We noticed in pre-test video material that one common problem is that the trackers get confused when bees repeatedly fly in- and outside the camera view without (immediately) entering the hive, especially when these bees fall within the pixel area of the entrance opening. Another problem is that the pollen is not always visible in the images due to the bee's position. Therefore, we tried out different camera angles to see whether this makes a difference.
- **Entrance coordinates:** The entrance area is defined as a set of coordinates of some shape (rectangle, polygon, circle). In pre-tests, we observed that the bee tracker is also influenced by how generously we draw the entrance area, i.e. how much padding we give around the opening.

To find out the best combination of variables, we set up three different cameras at a hive with different FPS: 50, 100 and 200. All cameras had the same resolution but came from different vendors. We positioned them from three different angles: front, side and vertical down as shown in Figure 17.

We rotated the positions of the camera. For each rotation, we took two takes of 5 minutes. To synchronize the cameras, we used a piece of paper that we held





Figure 17: Camera setup. We used the video material from the three black cameras for evaluation. The white camera surveils the hive permanently and its footage was not included in the evaluation. Source: Daniel Boschung

in the camera view at the start of a take. In total, we gathered 6 takes and 18 videos. Given that we could not start all cameras at the same time, we tried to cut the lengths of the videos such that they show the same sequence. Here, it turned out that from the 6 takes, only 2 were usable because of synchronization issues in the others. Owing to the time-consuming nature of the evaluation, we only considered one take from the two for evaluation in the end.

We manually counted the three videos of this take. Due to the long opening of the hive that creates a large area of where bees can enter, we decided to crop the videos in two halves, counting the halves separately. To assess the counting agreement, two people counted the videos separately. Initially, we had larger deviations in the pollen counts for some of the videos. Firstly, the FPS of the video influenced how many pollen pellets we caught or missed. Therefore, we slowed down all videos to a FPS of around 200. Secondly, we did not define whether only the pollen with high certainty or also the disputed cases of pollen should be counted. Uncertain cases are especially those where bees bring in a small-sized pellet or one that is partially obscured due to lighting conditions and the bee's position. Eventually, we decided to only consider the pollen with high certainty. Thirdly, it was not always clear whether a bee that enters in the middle of the video should be counted to the first or second crop of the video. Finally, some bees came out of the hive again with pollen in which case the pollen count could be decremented or not. We decided to not decrement the count. In the end, it took us several renditions to establish a consistent way of counting. Table 1 shows the final pollen counts of the two counters for the three videos of the take.

Camera	Position	FPS	# 1st person	# 2nd person	# Avg
Gopro Hero 7	Vertical down	200	184	193	188.5
Sony	Side	100	176	180	178.0
Canon	Front	50	213	212	212.5

Table 1: Set-up and counts of video take used for accuracy evaluation of MODEL-SERVICE.

In general, the disagreement between the counters is not significant and we decided to take the average as the gold standard against which we compare the MODEL-SERVICE’s computed pollen counts. More noticeable were the differences in the pollen counts between different camera angles. This was a surprising outcome for us as this already suggests that pollen is best visible from the front angle. Nevertheless, we evaluated the pollen trackers on all angles against the corresponding manual counts of the respective angle. In future, evaluation on further takes should be performed to exclude the possibility that the superiority of the front angle is a chance result.

#### 4.3.1 Results

The first pollen counting algorithm performed very poorly. Almost no pollen pellets were recognized for all three video angles. Changing the entrance coordinates and the FPS did not improve this. On the other hand, the second algorithm reached more reasonable pollen counts as shown in Table 2. For the front angle at a FPS of 25, it computed the most accurate counts with only a deviation of 9% from the manual counts. Increasing the FPS, however, makes the pollen tracker overestimate the pollen counts by 97%. For the other two angles, the accuracy of the pollen counts was generally low, but increased with a higher FPS.

Angle	FPS	Pollen count	Deviation
Front	25	193	-9%
Front	50	419	+97%
Side	25	16	-91%
Side	50	43	-76%
Side	100	99	-44%
Down	25	10	-95%
Down	50	19	-90%
Down	100	35	-81%
Down	200	51	-73%

Table 2: Accuracy of pollen counts from the second pollen tracking algorithm.



### 4.3.2 Discussion

While the first pollen tracking algorithm generally detected in-going bees that had pollen in certain images, the criteria that this must be the case in half of the images in which the bee occurs was almost never met, especially given that many bees occurred in a large number of images. The second pollen tracking algorithm, on the other hand, has less restrictive criteria. Firstly, the bees bearing pollen do not have to be registered to be flying in by the bee tracker. Secondly, the minimum number of images where a bee must be detected with pollen is much lower. For the evaluation, it was set to 4. As a result, these weaker assumptions lead to higher pollen counts for the second pollen tracking algorithm. To understand the differences in the pollen counts better, we inspected a short portion of each video, visualized with annotation and tracking data. This revealed issues both in bee detection and bee tracking. Since the pollen tracker depends on the bee tracker and the pollen detection and the bee tracker and pollen detection, in turn, depend on the bee detection, errors in bee detection, pollen detection and/or bee tracking propagate into the pollen tracking.

Bee detection appeared to be accurate for front angle images, while for side and vertical down angles, the bee detection module missed many bees. This is probably due to the fact that the bee model was mostly trained on front angle images. Given that the bee tracker strongly relies on the bee detection, its accuracy could only be assessed for the front angle where it performed very poorly based on a preliminary manual counting. The main reason appears to be that bees often switch IDs, especially when they overlap each other in the image. Unfortunately, such bee overlaps are rather frequent in our evaluation set, thereby distorting both the bee and pollen counts. Nevertheless, when the camera is positioned at a front angle view and set to 25 FPS, the pollen count was close to the manual counts. In all other settings, however, the pollen counts substantially diverged from the manual counts. This might be due to the fact that we only tested and tuned the second pollen tracking algorithm on a short video from a front angle at 25 FPS.

That being said, the best performance from the front angle/25 FPS setting could also be a coincidence. In general, a much more extensive and balanced evaluation set should be created in the future, encompassing a mix of hives, filmed at various times of the day and under different camera angles. Due to time constraints, our current evaluation set only consisted of one take comprising videos of three different angles for one single hive. It is noteworthy that our small evaluation set tests the pollen tracking under more difficult conditions. Firstly, we filmed at midday where we had high bee traffic, and more bees create more overlaps between them, confusing the bee trackers more often. Secondly, the entrance of this particular hive had a light wooden color which greatly resembles the highly prevalent pollen colors of orange and yellow, and was also partially cast in shadow. Therefore, the pollen was challenging to see, even in the manual counting.

Meanwhile, the manual counting itself should be performed in a more rig-

orous way. We had larger deviations in the counting initially which we tried to minimise by following similar counting procedures. For future reference, we recommend to additionally record every counted pollen and resolve any disputed cases. This could be done, for instance, by recording the times at which a pollen is counted, extract those pollen instances where only one person counted them and double-check them with the person that did not count them. Another possibility is to count the videos in pairs, analogous to pair programming.

In summary, the second pollen counting algorithm was more robust against mistakes in the bee detection and tracking than the first one. Nevertheless, due to the poor performance of the bee detection and tracking, we could not determine the accuracy of the pollen tracking adequately. Thus, apart from creating a precise and balanced evaluation set, the following future work should be undertaken: Firstly, the bee models (and perhaps also the pollen models) have to generalize better to perform well under different camera angles. Secondly, the bee tracker requires considerable improvement, especially under high bee load. Once these two things have been fixed, the two pollen tracking algorithms should be re-evaluated again.

#### 4.4 Speed Evaluation

The runtime of a bee monitoring is an important metric as it heavily influences the operational costs. Furthermore, it is also in the interest of beekeepers to provide them up-to-date information on their bees' activities. For this reason, we subjected the *MODEL-SERVICE* to a stress test, measuring its processing speed on a larger set of videos in parallel.

We compiled video material from one hive on one day in June. For each hour of the day, we randomly selected one 5-minute video at a FPS of 30. Thus, we used 24 videos with a total length of 2 hours, which we ran through the *MODEL-SERVICE* at the same time. We repeated this experiment on different GPU machines on *AZURE*, each having 4 GPUs and 24 CPUs. The runtime that we measure encompasses – apart from the processing – the download of the video and models from cloud storage as well as the upload of the images and the annotation result file to cloud storage. It does not include the upload of the video to cloud storage.

##### 4.4.1 Results

Table 3 shows the average speed for the different GPU machines for processing the 24 videos. For each machine, also the fastest and slowest processing time of a video is given. Furthermore, the costs per month in Swiss francs are stated for a regular instance as well as for a spot instance.<sup>5</sup> The fastest one was the *NC24s\_v2* machine which processes 5-minute videos in 54 minutes on average, while for machine *NC24* and *NV24* the average duration was around 3 hours and 1.5 hours, respectively.

<sup>5</sup><https://azure.microsoft.com/de-de/services/virtual-machines/spot/>

Machine	Costs/month (Spot)	GPU	avg	min	max
NC24	2586.21 (337.69)	NVIDIA Tesla K80	03:08	01:29	05:38
NV24	3275.87 (550.56)	NVIDIA Tesla M60	01:26	00:59	02:33
NC24s_v2	5948.29 (776.11)	NVIDIA Tesla P100	00:54	00:38	01:39

Table 3: Average, minimum and maximum speeds for processing 5-minute videos on three different GPU machines from AZURE.

#### 4.4.2 Discussion

There is great variability both between machines and between the videos. Not surprisingly, the strongest machine (*NC24s\_v2*) was more than three times faster (3.5x) than the weakest machine (*NC24*), being also two times as expensive (2.3x). Compared to the *NV24* machine, the *NC24s\_v2* machine was also 1.6x faster, but 1.8x more expensive. Thus, the best cost and speed ratio comes with the *NV24* machine. Since all videos are processed in parallel, 2 hours of video material can generally be processed in the stated average speeds of the given machines. Thus, the *NC24s\_v2* machine can handle the daily video material of two hives, the *NV24* machine one hive, while the *NC24* machine cannot cope with the amount of data of a single hive.

In general, videos that had many bees in the images were the slowest to process. This makes sense as more bees translates into more pollen model inferences as well as image crops and uploads. Figure 18 shows the profiling results of one video processed on the *NC24s\_v2* machine with an average runtime.

It is noteworthy that the runtime was lower when running this video on its own compared to running it along with other videos (00:40 vs. 00:52). In a parallel setting, there are still resources that processes have to share. For example, in our case, six video processes were competing for access to the GPU. All profiling described hereinafter was performed in a sequential setting to avoid skews due to resource contention.

As in the pre-testing phase (cf. Section 4.2.9), parsing the model network outputs (`parse_network_output`) took an unreasonable amount of time, namely 54% of the total processing time. The vast remainder of the runtime (30% of the total runtime) is consumed by running the image through the model network (`pass_through_network`). We obtained an inference time (`dnn_Net.forward`) of ~35.7 FPS on the medium bee model. This is faster than the reported FPS of 26 from the second AI group, who also tested on a *NVIDIA Tesla P100* [1]. This difference might be due to the fact that our inference code uses `OPENCV` [60] while the second AI group used `DARKNET` [69] for inference. In fact, others have also reported such a benchmark gap for different frameworks [68]. Nevertheless, this gap could also be explained through different video material used for testing. Indeed, when we profiled the `MODEL-SERVICE` on the video that took the longest to process, the FPS dropped to 14.7. Profiling the video with the shortest runtime again yielded 35.5



Figure 18: Profiling of MODEL-SERVICE on a 5-minute video.

FPS. Thus, the only explanation for longer inference times is the number of bees that an image contains.

There was also substantial variance in the runtimes of obtaining bee (`add_bee_annotation_set`) and pollen annotations (`add_pollen_annotation_set`). In the pre-testing phase, these were almost equally matched. In the evaluation, creating bee annotations took longer than creating pollen annotations on average (1.6x). The situation flips when many bees are in the images. For the longest running video, generating pollen annotations lasts 1.8 times longer than for bee annotations.

In summary, the costs for monitoring a single hive around the clock amount to around 3000.- CHF per month when using a regular machine. These costs can be cut by up to 90% using a spot instance. Spot instances may be evicted occasionally, though, unlike regular instances that guarantee high availability. Since the MODEL-SERVICE is tolerant towards interruptions as CELERY features a retry configuration to re-run tasks when they are stopped, spot instances are a viable alternative. In this case, the costs are around CHF 400.- per hive. Note that these costs only apply for this particular hive and for the specific recording time (mid June). Consequently, a more extensive evaluation has to be performed in the future to precisely estimate the expenses, comprising video material from around

the year and for a mix of hives. That being said, CHF 400.- is a very high price that beekeepers are most likely not willing to pay for. Thus, to reduce the operational costs, we need to decrease the runtime of videos. We see two ways of doing so:

- **Optimize code:** First and foremost, a more efficient algorithm for parsing the model network outputs should be devised. Perhaps, it might also help to run this portion of code on the GPU instead of the CPU. Furthermore, training a unified model that detects both bees and pollen at the same time could also help shorten the processing time, especially in the case where many bees appear in an image. This would also reduce the complexity of the code substantially. Moreover, this unified model could also detect other things such as the entrance for the bee tracker or the grey card for color correction.
- **Reduce data:** It might not be necessary to analyze all the data that cameras produce. Instead, it could be that a fracture of the video material, carefully sampled across the day, suffices for reaching the same results as when all video material is processed.

## 5 Systems Providing Flora Data

This section answers RQ1b:

*Which systems provide flora data that help capture the pollen diversity?*

In this section, we turn our attention to the pollen diversity metric, which denotes the number of different plants that bees pollinate. The pollen diversity shows us the biodiversity around a beehive, but also how balanced the diet of the bees is. Together with the pollen counts (cf. Section 4), we can compute the food supply for a beehive.

Our cameras capture the colors of the pollen that bees fly into the hive. Pollen can come in a variety of colors, providing a first clue as to which plants may have produced them. However, determining a pollen’s botanical origin based on its color alone is usually not accurate enough since many plants produce pollen with similar colors. Furthermore, different lighting conditions make the color matching very challenging.

A more robust strategy for computing pollen diversity is to use the knowledge of which plants release pollen at a specific point of time and place. The basis of this plant-matching is sufficient data on flora. In the following, we present a set of flora data sources that we considered for integration into our platform.

### 5.1 Methodology

Since various platforms provide flora data sources, we only selected the most suitable ones for our needs. We defined the following evaluation criteria:

- **User-Friendliness:** How easy is it to contribute data, especially by laymen, e.g. is there help with the identification? How much is manual or automated? Is there a mobile app to make contributions more convenient in the field? Furthermore, are there incentives for making continuous contributions, e.g. gamification and community-support?
- **Data Accessibility:** Whether the data can be easily retrieved and parsed and whether it is publicly available. This also includes a stable API that does not have too restrictive rate limits and ideally does not cost anything. Moreover, the data should be updated in real-time and kept up-to-date.
- **Coverage:** Whether there is enough data available and/or whether users will deliver enough data in the future. The following dimensions are of interest:
  - Plant coverage: The number of plant species for which we have data.
  - Temporal coverage: Data should be available for every time of the day and throughout the year.
  - Spatial coverage: Data should be available in all regions of Switzerland, ideally nearby the hives on our platforms.
- **Accuracy:** Whether the data is correct and precise e.g. regarding species identification, location, time and phenology data.
- **Integration effort:** How much implementation effort it takes to interface with the system in question or how easy it is to extract and parse the data.
- **Maturity:** Whether the system as a whole runs stable and is maintained. The BLS project is a long-term project and ideally depends on systems that also have a long-term vision so as to decrease the maintenance effort. This is only relevant when we regularly pull live data from the system. Some of the data (like pollen colors and pollen quality) may also be static and therefore maturity does not play a role there.

We did not evaluate every system as there are a great number of them. We excluded the following:

- Systems offered by businesses rather than research-oriented projects since they typically do not share data for free.
- Systems with a small user base as determined, for instance, via the number of downloads<sup>6</sup> as this affects coverage as well as maturity.
- Systems that mostly produce data outside of Switzerland since our beehives will stand in Switzerland.

---

<sup>6</sup>The threshold was set at <5000 downloads on GOOGLE PLAY.

- Data sources that cannot be shared on all operating systems (both iOS or Android for mobile apps or/and a web-based platform).

In the following sections, we describe the systems that we considered and how strongly they meet the evaluation criteria. The sections are structured around what kind of data the systems provide:

- **Plant observation:** Where plants are observed to flower.
- **Pollen color:** The colors of the pollen that plants produce.
- **Pollen abundance:** How much pollen plants release for bees.

## 5.2 Plant Observation

Plants have been observed for centuries. Historically, observational data was collected manually, typically by botanists who also had the knowledge of correctly identifying the plants. With the rise of the internet, this paradigm changed: citizen scientist platforms sprang up and allowed anyone to share observations, the plant identification being supported both by the community as well as AI.

A plant observation delivers us two pieces of information that are beneficial for our platform:

- **Plant occurrence:** Plants do not grow everywhere. Certain plants may only grow at certain altitudes or in certain regions (e.g. alpine flora). Bees have a foraging area of approximately 3km around a hive. Therefore, obtaining spatially precise data is important so as to know on which plants the bees forage.
- **Plant phenology:** A plant undergoes various phases throughout the year such as budding, leaf development, flowering, fruiting, leaf coloration and fall. Since bees can only collect pollen from plants when they flower, observational data should be annotated with phenology traits. To obtain the onset and end of a flowering phase for plants, temporal coverage is crucial.

Ideally, we obtain such observational data ‘live’ because whether plants currently flower in some place depends on more factors than just the time and place. For example diseases, weather and agricultural activities also have an impact on plant thriving. As the spatial and temporal coverage of such data varies across plant species and is often sparse, we also considered historical data which can give us a range of when plants may potentially flower and in which regions this is likely the case.

### 5.2.1 BeeKeepr

BEEKEEPR [7] is an app from the PlanBee-Project, founded in 2018 and supported by the University of Passau. It provides tools and information for managing beehives

such as a beekeeper diary, information on varroa and bee weather, a flowering calendar showing when plants flower and the quality of pollen they produce as well as a pollen color directory. Moreover, they provide an image detection module that serves for building up an extensive database of pollen colors with the help of their users.

- **User-friendliness:** BEEKEEPER is available as an app. The user uploads an image, ideally with bees sitting on them collecting pollen, which (s)he can crop. An AI module analyzes the image for plant and bee species, and pollen color. The user may then agree and disagree with those predictions and suggest other species and pollen colors. The identification process is not supported by a community as one's data is not visible to other users. Coordinates and time are automatically extracted. Generally, we found taking images of bees with pollen on flowers challenging with a mobile phone. The creation procedure in the app is intuitive. However, no feedback is given after submission and the history of uploaded images is not visible which might be demotivating to users.
- **Data accessibility** Data cannot be downloaded from the app and is also not retrievable via an API.
- **Coverage** The exact number of observations is unknown. As the project is young and does not have a large user base yet, it can be assumed that the number is low.
- **Accuracy:** Plant, bee and pollen detection is based on AI and is confirmed or rejected by the user. There are no second opinions by other people. Location and time are automatically extracted, but cannot be edited by the user (and are also not visible in the app).
- **Integration effort:** The team offered us to transfer all the images and annotations that we upload to our servers. This would require a medium implementation effort as one side has to open up an API endpoint.
- **Maturity:** The app was deployed in February 2020 and has around 5000 downloads.

### 5.2.2 Flora Incognita

FLORA INCOGNITA [82] is a project carried out by the Technical University of Ilmenau and the Max Planck Institute for Biogeochemistry, and since 2014 funded by several federal ministries and agencies in Germany. The project provides an app for determining plants using AI.

- **User-friendliness:** FLORA INCOGNITA is available as an app. A user takes images of different parts of the plant and based on these, an AI module suggests the best matches from which the user can choose from. Coordinates



and time are automatically extracted. The identification process is not supported by a community as one's data is not visible to other users.

- **Data accessibility** Generally, data of other users are not visible. Although it is possible to download one's own data, there is currently no (public) API to retrieve the data of other users.
- **Coverage:** The app is specialized on plants from Central Europe containing around 4800 species and has a focus on Germany. As there is no API, we cannot determine how many observations are available in Switzerland. According to the website their database contains 15'000 observations from Germany.
- **Accuracy:** Plant determination is based on AI and what the user chooses. Location and time are automatically extracted, but cannot be edited by the user. There are no second opinions by other people.
- **Integration effort:** As there is no API, the only way to obtain the data of other users would be to ask them to export images and identifications, and submit them on our platform. This would not only imply a high implementation effort but also many manual steps.
- **Maturity:** The app has been in operation for around three years and has around 1 million downloads.

### 5.2.3 iNaturalist

iNATURALIST [39] is a social network platform for sharing sightings of organisms. The project began back in 2008 and became a joint initiative between the California Academy of Sciences and the National Geographic Society in 2017.

- **User-friendliness:** iNATURALIST is available as an app and as a web version. The user takes or uploads images of a species and may then propose what kind of species (s)he observed. This identification is supported by AI as well as the community that may agree or disagree with such an identification afterwards. Furthermore, the user can add an annotation for the phenology (*flowering budding, flowering, fruiting*) which the community can confirm as well.
- **Data accessibility:** iNATURALIST provides a public and free Representational State Transfer (REST) API with rate limits of 10'000 requests per day.
- **Coverage:** iNATURALIST contains ~24 million plant observations world-wide from ~115'000 species and ~62'000 observations of *Angiospermae/Pinopsida*<sup>7</sup>

---

<sup>7</sup>This is an approximation for plants that deliver pollen.

plants in Switzerland from ~2'200 species contributed by ~4000 users. Observational data is shared across the year, but peaks in the summer time as illustrates Figure 19. Figure 19 also shows that the number of observations trends upwards with ~24'000 *Angiospermae/Pinopsida* plants observations shared by the community last year.

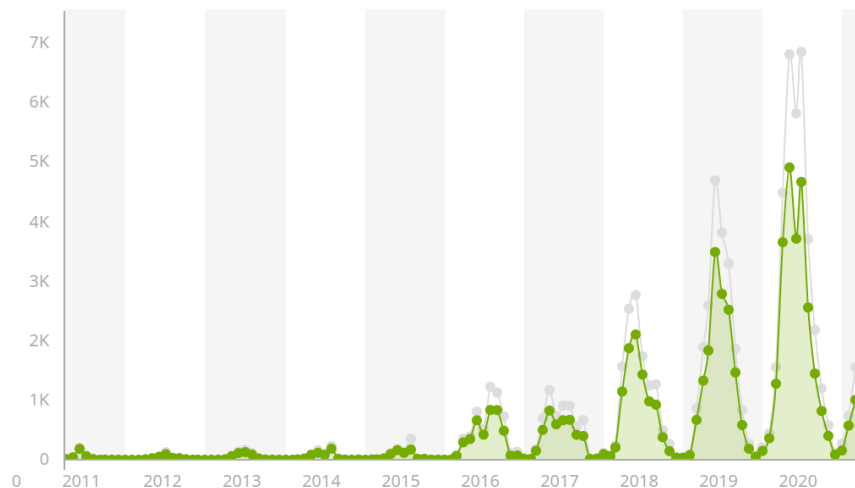


Figure 19: History of plant observations in Switzerland on iNATURALIST [39]

The spatial coverage of *Angiospermae/Pinopsida* plants in Switzerland is visualized in Figure 20. As can be seen, there are gaps in certain regions of Switzerland, for example in the region of Emmental-Oberaargau and Luzerner Hinterland. When looking at a concrete species, in Figure 21 showing *Bellis Perennis* as an example, the sparseness is evident.

As previously mentioned, blooming status is also available, although the vast majority of observations have no phenology annotation. Nevertheless, the temporal coverage is sufficient for more common plants to deduce onset and end of flowering. For example Figure 22 shows the phenology course of *Bellis Perennis* and *Corylus Avellana*.

- **Accuracy:** Time and location are automatically extracted and can be manually edited. iNATURALIST assigns quality labels to observations. An observation can have three quality grades: *hobby*, *needs ID* and *research grade*. Research grade is allocated when at least two people have identified the plant species and when image, coordinates and date of the observation are available. As we only consider observations with research grade, the data can be considered as very accurate. Only the phenology annotation does not enter into the research grade. However, as seen in Figure 22, the phenology appears to be accurate for common plants. There are only two phases for flowering (*flowering budding*, *flowering*). The end of flowering is not

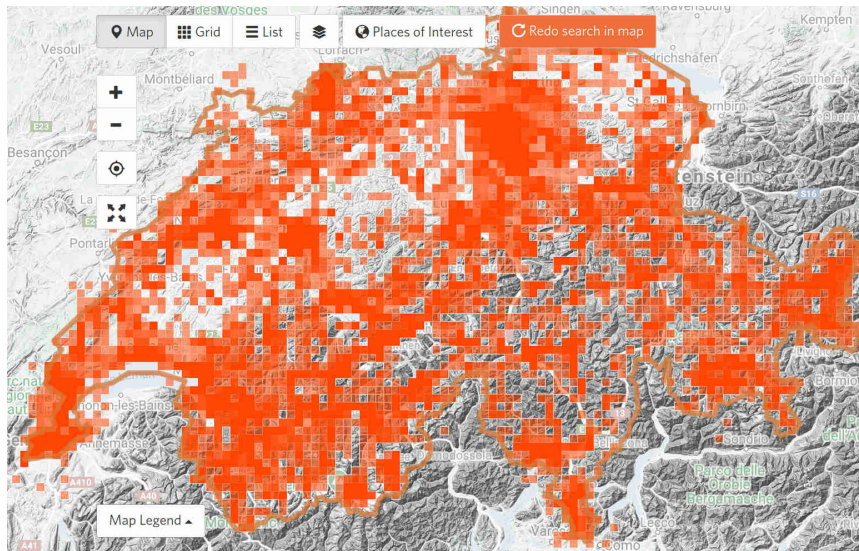


Figure 20: Spatial coverage of *Angiospermae/Pinopsida* plants in Switzerland on iNATURALIST [39]. Red squares denote areas with observations and stronger red shades indicate more observations in the area.

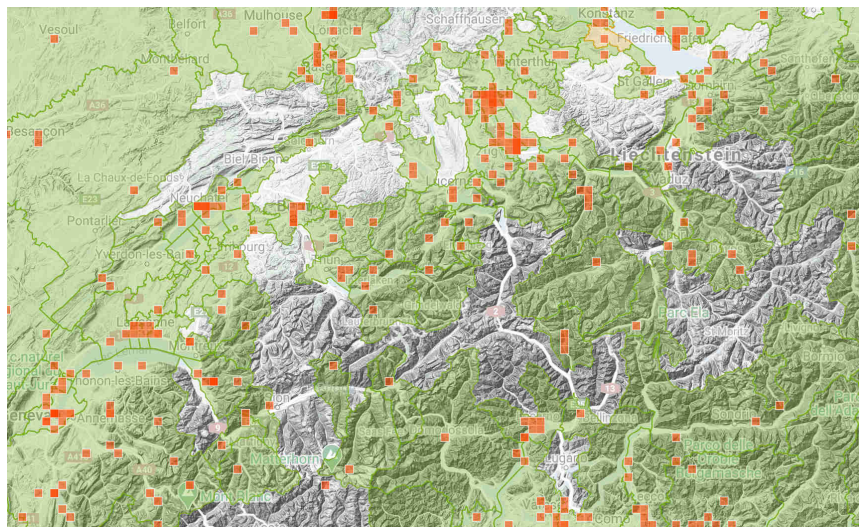


Figure 21: Spatial coverage of *Bellis perennis* in Switzerland on iNATURALIST [39]

explicitly annotated which means that the end of the blooming cannot be computed precisely.

- **Integration Effort:** As there is a REST API with many filter functions, the implementation effort is low.

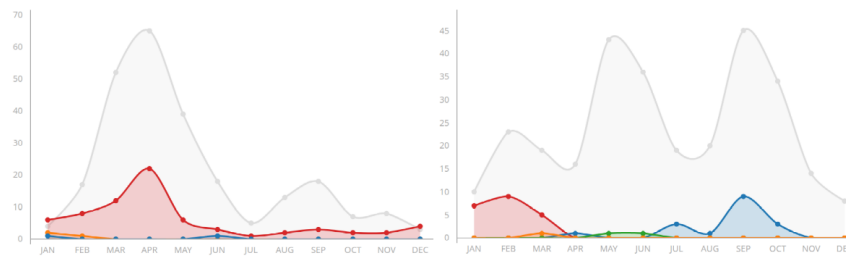


Figure 22: Phenology data for *Bellis Perennis* (left) and *Corylus Avellana* (right) on iNATURALIST [39]. The colors of the curves denote the following: red → *flowering*, grey → *no annotation*, blue → *fruiting*, orange → *flowering budding*, green → *no evidence*.

- **Maturity:** The platform has 1.8 million registered users around the world and has been in operation for more than 10 years. The app was deployed in 2011 and has 1 million downloads on GOOGLE PLAY.

#### 5.2.4 Info Flora

INFO FLORA [40] is a private, non-profit organization founded by the city of Geneva, Pro Natura, the Swiss Botanical Society and the Swiss Academy of Natural Sciences, and supported by the Federal Office for the Environment. The foundation provides information on wild plants in Switzerland and promotes their conservation. It maintains a database of Swiss flora observations which are documented in maps.

- **User-friendliness:** INFO FLORA provides an app. The user first selects or enters a species name (scientific or common name), then specifies location and time and optionally adds images and other metadata such as abundance, habitat, phenology and more. Thus, a good level of flora knowledge is assumed when adding observations. Therefore, the app is not suitable for botany beginners, especially given that the identification process is also not supported by a community.
- **Data accessibility:** There is a free and public REST API. The data is available at different accuracy levels, such as 5kmx5km,<sup>8</sup> 1kmx1km or as individual observation records. The usage of more precise data is more restrictive (written approval by data centers, no deposition in repository, etc.) for data privacy and plant protection reasons.
- **Coverage:** INFO FLORA's database covers all wild-growing plants from Switzerland (i.e. no cultivated plants). The temporal and spatial coverage is high as

<sup>8</sup>Data is handed out 'summarized', i.e. for a 5km area, we obtain a list of plant species that have been observed there.



there is a large amount of observational data available from different places which can date back to the 19th century. For example, Figure 23 shows the spatial coverage of *Bellis Perennis*. Moreover, there is detailed information on plants such as flowering period, habitats and ecological indicators. In the past years, ~700'000-800'000 observations were registered, a number that tends to rise every year.

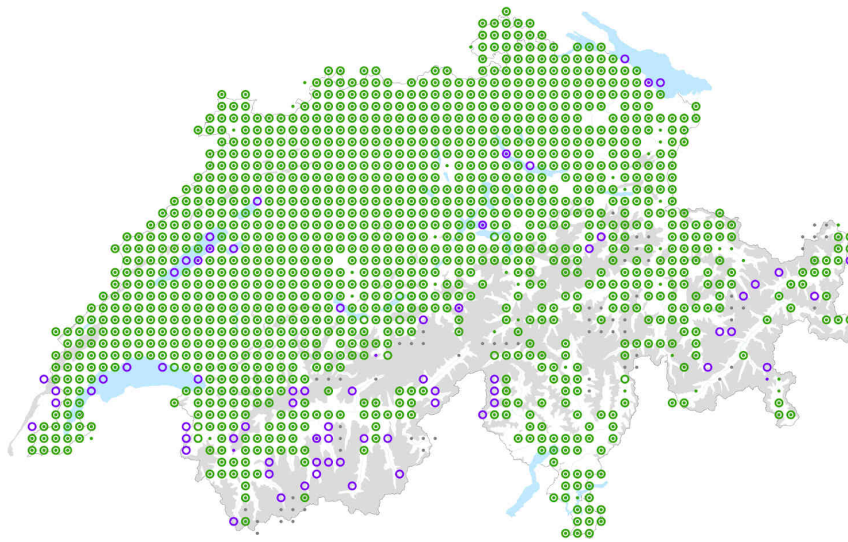


Figure 23: Spatial coverage of *Bellis Perennis* on INFO FLORA [40]. Green circles denote validated observations, while the purple ones are not validated yet.

- **Accuracy:** The app works without an AI module. The target group are people who know plants well. Moreover, outliers or problematic data points are double-checked by the INFO FLORA team. Time and location are manually added. Thus, there is a margin of error in precision. Nine phenology phases are available to select from, five of them concerning the flowering status (*not flowering*, *with flowering buds*, *start of flowering*, *full flowering*, *end of flowering*).
- **Integration effort:** The integration effort is low as there is a REST API.
- **Maturity:** The foundation has existed since 2004. The app was launched in 2016 and has 5'000 downloads.

#### 5.2.5 PhaenoNet

PHAENONET [86] is a network of schools and scientists that documents the phenology of a selected few plants throughout the year. The network is supported

by GLOBE Switzerland, the Federal Office for the Environment, MeteoSchweiz, ETH, Science et Cité, the Plant Science Center and the Botanic Garden of the University of Bern. Part of the data stems from METEOSCHWEIZ [52], while the rest of the data is collected by PHAENONET itself via a web-application. Not only the phenology phases are documented, but also the attributes that affect the phenology of plants such as weather, exposition and environment. This allows researchers to investigate the interplay of phenology and other factors. Recently, PHAENONET has started collaborating with SWISSCOM on the deployment of Internet of Things (IoT) devices, measuring weather parameters in trees and in the roots of plants.

- **User-friendliness:** There is only a web-based version. Users specify location, plant species, phenology stage and some other attributes affecting phenology such as environment and exposition. For the identification process, there is no AI module nor a community that can support this. Thus, the app is more difficult to use for novices in botany. However, given that only knowledge about 15 plants is required, the learning effort is not too high.
- **Data accessibility:** There is a free and public REST API to retrieve the data.
- **Coverage:** PHAENONET covers 15 carefully-chosen plant species from which the phenology of most other plants can be deduced. ~8'000 data points are delivered every year, a number that is rising. The temporal coverage is high as the plants are observed regularly. The spatial coverage is medium as the weather stations from METEOSCHWEIZ are more or less evenly distributed. Observation density is naturally high around schools collecting the data. On the other hand, there are gaps in certain areas such as in the mountains. Figure 24 shows the phenology observations of *Corylus avellana* in the year of 2021 in Switzerland.
- **Accuracy:** Observations from METEOSCHWEIZ are added by semi-professional people. The internal data is contributed by schools where teachers instruct the students on how to determine plants and their phenology phases. In terms of accuracy, there are only minor differences (max. 5% deviation) between the two data sources. The phenology data is precise as phenology is expressed through the BBCH-scale. Time and location are manually added. Thus, there is a margin of error in precision.
- **Integration effort:** The integration effort is low as there is a REST API.
- **Maturity:** The web platform has been in operation for several years.

#### 5.2.6 Pl@ntNet

PL@NTNET [64] is a citizen-science platform for plant identification in images. The project was launched in 2009 by a group of scientists from various French institutes.

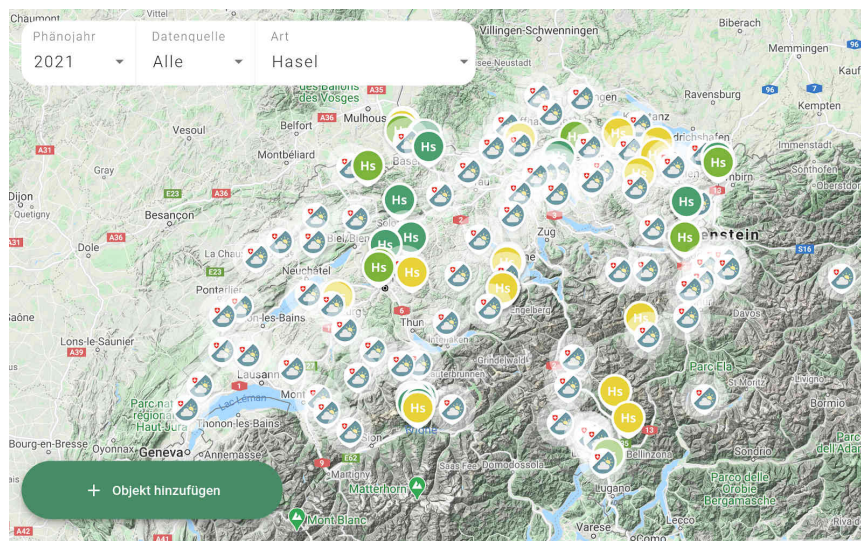


Figure 24: Phenology observations of *Corylus avellana* in the year of 2021 on PHAENO.NET [86]. Observations denoted as *Hs* are directly from PHAENO.NET, the others are from METEO SCHWEIZ. Yellow observations denote flowering status.

- **User-friendliness:** The project provides both a mobile and a web-based version. To share a plant observation, users provide an image of the plant and select what part of the plant they photographed (e.g. leaf, flower, fruit, etc.). An AI module analyzes the image and suggests the best matches from which the user can confirm one. Furthermore, the community can confirm every identification.
- **Data accessibility:** There is a REST API that allows users to submit images and have them analyzed for identification. 200 requests per day are free of charge for research purposes. There are no endpoints to get the observations produced by the community. However, PL@NTNET annually shares image data and has released the observations from the years 2017-2018 in the French territory and Corsica. There is an option to export one's own observations to comma-separated values (CSV).
- **Coverage:** PL@NTNET's database contains around 4 million observations from West Europe. We could not determine the number of observations of plants in Switzerland.
- **Accuracy:** Users identify plants supported by AI and the community. Time and location are automatically extracted from the image's metadata which, however, cannot be edited by the user manually.
- **Integration effort:** As the observations made by the community are not accessible via an API (and the observations from France are not relevant),

the only way to integrate with PL@NTNET would be to add an observation sharing page on our platform. Users would upload images of plants there that are forwarded to PL@NTNET to obtain a plant identification. However, this would be a high integration effort.

- **Maturity:** The app was published in 2014 and has 10 millions downloads.

### 5.2.7 Discussion

Based on the evaluation criteria, we opted for the data of iNATURALIST, INFO FLORA and PHAENONET to be integrated into our platform. All three systems have a strong open-access philosophy making (almost) all data publicly available with useful filtering functions. The other investigated systems score lowly on data accessibility and therefore the integration of their data would entail a greater implementation effort.

In terms of user-friendliness, iNATURALIST convinced us the most. There is a strong sense of community because (most) observations are publicly visible and users help each other to identify species. In this regard, PL@NTNET also scores highly. Indeed, if PL@NTNET were to release endpoints for retrieving observation data, it might win over iNATURALIST.<sup>9</sup> The other systems, on the other hand, are more ‘closed ecosystems’. We think that an open, helpful community keeps people motivated to share more observations. Furthermore, INFO FLORA and PHAENONET require some level of plant knowledge because there is no AI module aiding the identification process. Therefore, we decided to use iNATURALIST as our main platform for our citizen scientists to share new observations.

Regarding coverage, we could not adequately compare the systems as the observation, species and user counts were not always available, in particular for Switzerland. The download count itself is a vague estimation for how many observations users will potentially contribute to in the future. In this regard, PL@NTNET has an edge over the other systems with 10 million downloads. However, to obtain fair comparison metrics between the systems, we would require the counts for plant observations made in Switzerland in the recent years. Among the systems for which we have numbers, INFO FLORA was the strongest with almost 800'000 observations contributed every year - a number that is most likely not surpassed by the other systems for which we do not have the counts. Therefore, we consider INFO FLORA as the best source for retrieving spatial coverage of plant species. However, as the observations are summarized in 5km regions (i.e. they are not available as individual data points), a spatial precision of less than 5km is not possible and has to be complemented by data from a different system (e.g. iNATURALIST). With respect to phenology data, PHAENONET captures the exact onset and end of flowering phases in the most systematic way. They record environmental parameters (e.g. the temperature) together with phenology metrics, which allows

---

<sup>9</sup>In fact, this feature is planned in future versions according to the FAQ.



us to more accurately deduce whether other plants in other locations with similar environmental values also flower.

Regarding plant identification accuracy, INFO FLORA and PHAENONET rely on the user's knowledge, BEEKEEPER and FLORA INCOGNITA additionally on AI, while PL@NTNET and iNATURALIST additionally involve the community to confirm the identifications. We consider identifications supported by AI and approved by the community as the most accurate. As we can filter observations based on that on iNATURALIST (setting quality to *research grade*), this data will be very accurate. Moreover, iNATURALIST allows users to edit location and time of observations unlike the other systems, which increases the accuracy of the data further as mobile phones do not always register the location correctly.

In conclusion, data accessibility was a key differentiator between the systems. BEEKEEPER generally scores lower on all evaluation dimensions. However, it is the only system that would not only deliver us the basic plant data (identification, coordinates, time), but also the pollen color selected by the user. This would help us build up a database of pollen colors. Nevertheless, we did not pursue a collaboration as the implementation effort is higher and interfacing with a system that is rather young poses a risk.

### 5.3 Pollen Color

Every plant species produces pollen in specific colors. The MODEL-SERVICE (cf. Section 4) also extracts the colors of the pollen, which we can match against the pollen colors of a plant species. For this reason, we strove for a pollen database that covers as many plant species as possible and is encoded in a format that allows for similarity measurements (e.g. as a RGB triplet or in hexadecimal format). Finding an extensive and well-parsable database of pollen colors proved challenging. We found several sources containing information on pollen color but the color information was not easy to extract. For example, [2] published a booklet that describes plants in a non-tabular format, the pollen color shown as an image. Therefore, we searched for tabular-encoded pollen color sources on the internet. We found four such tables:

- Akesi Farms [87]: ~50 plants common in North America, some of which are stated with their scientific names and others only with their English trivial names. The table is available as an HyperText Markup Language (HTML) file.
- Honigbörse [37]: ~120 plants common in Germany which are only stated with their German trivial names. We found identical tables also on other beekeeper websites (e.g. [38], [27]). The table is available as an HTML file.
- Schulbiologiezentrum Hannover [77]: ~130 plants common in Germany which are stated with their scientific names. The table is available as an Excel file.

- Wikipedia [91]: ~80 plants which are stated with their scientific names. The table is available as an HTML file.

The color information is easier to parse in HTML files as the colors are encoded in RGB or hexadecimal format in the style attributes of elements. In the case of the Excel table, custom functions have to be written to extract the background color via a custom function. However, in general, parsing HTML is more complex than Excel tables (which can be exported to CSV).

We decided to use the *Schulbiologiezentrum Hannover* table as it has the largest plant coverage. Moreover, all plants are listed with their scientific names which eases the plant matching.

#### 5.4 Food Abundance

Plant species differ in how much food they provide in the form of pollen and nectar for bees [66]. This is interesting for us this also gives us an estimation of how likely a bee brings pollen from that particular plant species. There are data sources providing information about this food abundance. The largest, most reliable and accessible source we found is the *Pritsch* table that is based on [66]. It contains the nutritional values of 220 plants common in Germany. Each plant receives a score between 0 and 4 for nectar and pollen separately and a total score computed from the nectar and pollen values. The higher the score is, the more abundant the food is. We also found other sources containing nutritional values, for example two pollen color tables described in Section 5.3 also list nutritional values [91, 37]. However, both of them have a lower plant coverage and the *Wikipedia* table does not distinguish between pollen and nectar.

## 6 Plant-Matching Algorithm

This section answers RQ1c:

*How can flora data be leveraged to measure pollen diversity?*

In the previous section, we selected the flora data sources suitable for integration into the BLS platform. These data sources provide information on current and potential plant occurrence, phenology, pollen color and food abundance, which allow us to estimate the pollen diversity. The module for computing this metric is called the `BEEFOODCHECKER`. Its heart is the plant-matching algorithm which returns a set of plants that (may) grow and flower at a given point of time and location as evidenced by the flora data. Each plant is ranked according to a score calculated from various matching criteria. Counting the plants from the upper ranks, we have a measure for pollen diversity.

## 6.1 Related Work

Assessing the pollen diversity can be done in manual or automated ways. To obtain reliable and accurate results, pollen is typically collected and analyzed based on colors and shapes. Since this is a manual process, it is more expensive and time-consuming. On the other hand, automated systems overcome this shortcoming but might be less accurate.

### 6.1.1 Palynology

Palynology is the scientific term for pollen analysis. To analyze pollen, the pollen has to be collected first, which can be done in the following ways:

- **Pollen trap:** A pollen trap is an apparatus placed at the entrance of a hive [59, 13]. The opening of a trap is so small that bees have to crawl through to enter the hive. During this passage, bees lose some of the pollen, typically 30-70% of the pellets depending on the size of the opening and the pellets [14].
- **Melissopalynology:** One can also collect pollen from the honey. The pollen usually stems from plants visited by bees, but may also come from wind-pollinated plants or algae and fungal spores [89]. Melissopalynology is not only performed to determine the biodiversity around a beehive, but also to detect incorrect labeling of honey.

Once pollen is collected, it can be analyzed using the following techniques:

- **Naked eye:** The simplest way of determining pollen diversity is to count the number of different pollen colors and shapes as seen by the naked eye. For instance, [17] evaluated the correlation between color diversity and real floral diversity, coming up with a formula for calculating the floral diversity from the color diversity. As noted by the authors, this only yields a rough measure because the number of different colors does not equal the number of different plants. Indeed, many plants produce pollen in the same colors and some plant species produce different pollen colors. Furthermore, they point out that individual pollen colors can map to a different number of plant species (e.g. many plants produce yellow pollen, but only a few produce blue pollen).
- **Light microscopy:** Using a light microscopy is the most common method of pollen analysis [59, 13]. It is also more precise than an analysis by the naked eye because the pollen can be determined up to the species level. However, since existing databases of reference pollen forms are incomplete, researchers are sometimes still unable to assign a pollen grain to a specific plant species [13], in which case the pollen grain can only be categorized into a higher taxonomic rank.

- **Metabarcoding** A more recent and more accurate method is to use DNA-based identification [65, 71].

### 6.1.2 Data-Driven Approach

Pollen traps are invasive and do not retain every pollen pellet. Moreover, the collection of pollen samples and their analysis is expensive and time-consuming. Therefore, pollen data is often temporally and spatially limited, meaning pollen data is only available in specific locations, e.g. in agricultural or more urban areas, and at specific times of the year [13]. Using a citizen science approach, the temporal and spatial coverage can be increased [13]. However, using a purely computational, data-driven approach, pollen data can be gathered in a non-invasive way with low human intervention and with as little temporal and spatial ‘gaps’ as possible.

As outlined at the beginning in this thesis, we leverage flora data to determine pollen diversity. Such an approach is not entirely new as there are already websites that also exploit flora data, albeit in more rudimentary forms. For example, [4] provide a tool for matching plants based on pollen colors and flowering time, supplying information on pollen abundance for each matched plant. Another example is [11], which matches plants based on flowering time and planting place (balcony, garden, field, forest, trail, meadow). It also shows the pollen abundance and the total food availability for a given month. However, both tools do not consider regional deviations nor fluctuations in the flowering periods due to environmental differences. The algorithm that we present in the ensuing section also takes into account such other variables.

## 6.2 Implementation

In Section 5, we decided to integrate the observational data from iNATURALIST, INFO FLORA and PHAENONET, the pollen color data of SCHULBIOLOGIEZENTRUM HANNOVER and the pollen abundance data of PRITSCH. These data sources are the foundation for the plant-matching algorithm and therefore we describe their integration into the BLS platform first.<sup>10</sup> Thereupon, we present an algorithm that leverages all these data sources to establish the pollen diversity in a certain area at a given point of time. The performance of this algorithm is then evaluated against the results of a microscopy analysis of two pollen samples.

### 6.2.1 Database Schema

Integrating the data sources into our platform means to import their data points into our POSTGIS database. To do so, we first devised a database schema which is shown in Figure 25. The plant model forms the center of the schema and stands

<sup>10</sup>Due to time reasons, PHAENONET is the only data source that could not be integrated as part of the thesis. Instead, flowering information is taken from iNATURALIST and PRISCH for the time being (cf. Section 6.2.6).

for a taxon within the plant (*Plantae*) kingdom. It specifies basics such as its scientific name (`name`) and taxonomic rank (`rank`) as well as more bee specific attributes such as the general abundance value (`nutrition_grade`) and the color (`pollen_color`) of the pollen. A plant taxon may belong to a plant taxon of higher rank (e.g. the species *Bellis perennis* belongs to the genus *Bellis*) which is realized via a foreign key to itself. As taxa can have several scientific names as well as trivial names in different languages, a model `COMMONNAME` was created. The data of this particular table proved useful for mapping plant names since the different data sources designated the plants with various scientific and trivial names. For the observational data, we created two separate models: `Observation` and `PlantOccurrence`. The former literally encodes an observation of a plant with precise location (`location`), time (`observed_on`) and phenology status (`flowering`). The latter, on the other hand, is employed when observations are summarized within a region (as in the case of `INFO FLORA`), i.e. it only specifies that a plant occurs in that region (`region`) with a corresponding observation count (`obs_nb`). Finally, there is a `Nutrition` table that gives more fine-grained information as to how much (`value`) pollen and nectar (`food`) a plant delivers at a given point of time (`month`, `start_day`, `end_day`).

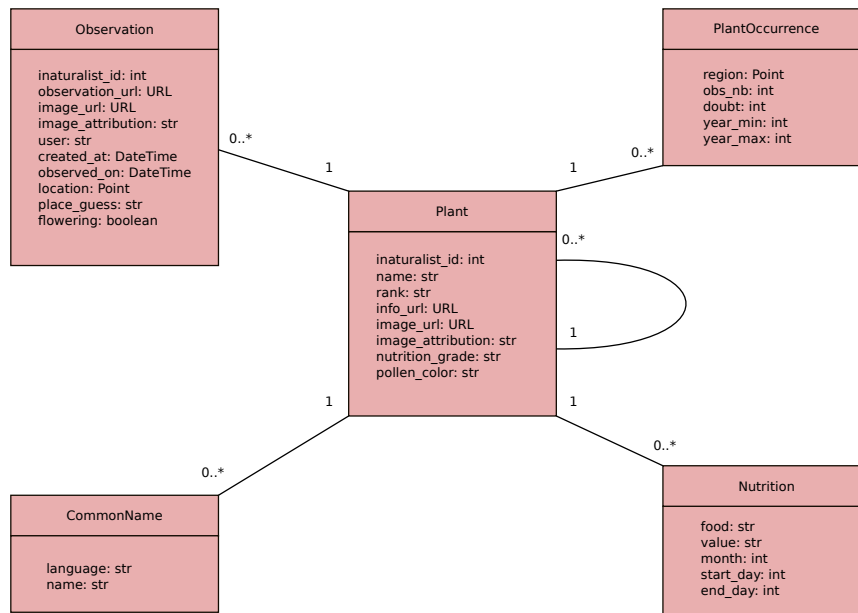


Figure 25: Database schema for the BEEFOODCHECKER

### 6.2.2 iNaturalist Integration

For iNATURALIST, we decided to import the complete *Angiospermae/Pinopsida* taxa data and the Swiss observational data with research quality. Taxa and observa-

tional data are served by two different endpoints in iNATURALIST's REST API. We pulled down the data by iteratively sending requests to the endpoints and storing the output in JSON files.<sup>11</sup>

Both dumps (taxa and observations) are then uploaded to cloud storage due to their massive size (~9 GB) so that we can repeatedly re-import the data into the database without having to run the time-consuming API requests. From both dumps, we then populate the tables `Plant`, `CommonName` and `Observation`. From the taxa dump, which contains more than 250'000 taxa, we only insert taxa on or above species rank and ignore species with less than 50 observations for performance reasons. Finally, we created a `CRON` job that retrieves new observations from iNATURALIST daily. Figure 26 shows how the data is visually displayed on our platform.

### 6.2.3 InfoFlora Integration

For INFO FLORA, we pulled down data from the taxa and atlas endpoints as the observation endpoint is not public and has restrictive uses. The atlas endpoint provides us occurrence data for a single taxon at a precision of 5km, whose output is also stored as a JSON file. The whole dump is around 300 MB in size and is therefore also uploaded to cloud storage. From the dump, we populate the table `PlantOccurrence`. Occurrence data is visually shown on a map from OPEN-STREETMAP [61] as seen in Figure 27.

### 6.2.4 Pollen Color Integration

The pollen color table from SCHULBIOLOGIEZENTRUM HANNOVER is an Excel file. The table is structured by blooming time phases. To make the pollen color easier to parse, we reduced the table to two columns, one for the plant name and the other one for the color information, and exported it to CSV. Plants specified as hybrid were removed from the table. From this CSV, the `pollen_color` attribute of the plants is populated.

### 6.2.5 Pollen Abundance Integration

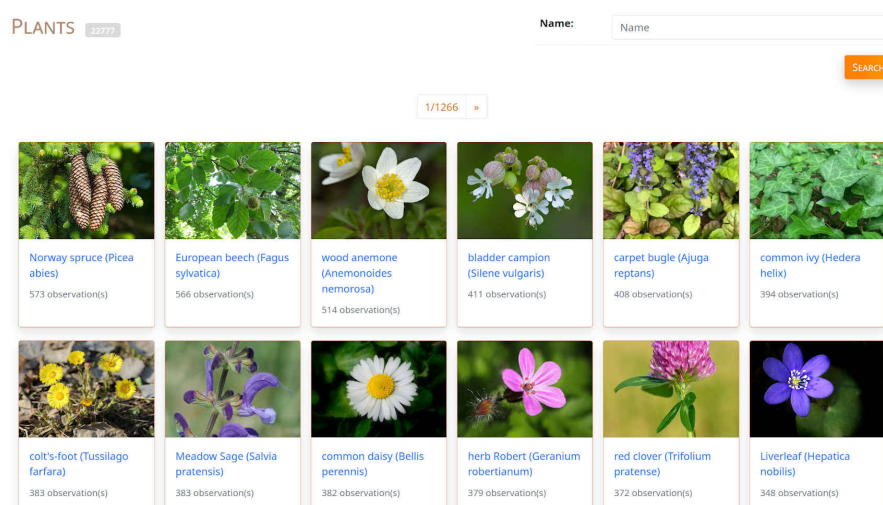
The PRITSCH table is available as an Excel file which we exported to CSV. Each row gives nutritional values of a plant for both nectar and pollen and for both the first and second half of a month. The data in these rows is then used to populate the table `Nutrition`. Figure 28 shows how the data is visualized on our platform.

### 6.2.6 Plant-Matching Algorithm

We define pollen diversity as the number of distinct species plants from which bees collect pollen. Thus, to compute this floral diversity, we must match the

---

<sup>11</sup>The API has a page limit size, restricting the output of an endpoint response to a maximum of 200 data points. Therefore, we have to pull the data in chunks of 200.



(a) Web page listing plants

OBSERVATIONS 74567

1/2983

FETCH OBSERVATIONS

Plant	Observed on	Place	User	Flowering	Link
Cornflower ( <a href="#">Centaurea cyanus</a> )	June 27, 2021, 8:21 a.m.	Chemin du Couchant, Cugy (VD), Vaud, CH	olgl	True	<a href="#">84667770</a>
common mugwort ( <a href="#">Artemisia vulgaris</a> )	June 27, 2021, 6:53 a.m.	6716 Leontica, Suisse	ericguglielmazzi	unknown	<a href="#">84662302</a>
Cobweb Houseleek ( <a href="#">Sempervivum arachnoideum</a> )	June 27, 2021, 2:35 a.m.	Chalet Weisshornblick, 3925 Grächen, Switzerland	lizzie_r	True	<a href="#">84592421</a>
Large-flowered Evening-primrose ( <a href="#">Oenothera glazioviana</a> )	June 26, 2021, 7:23 p.m.	Derendingen	mobau	unknown	<a href="#">84594114</a>
Perennial Cornflower ( <a href="#">Centaurea montana</a> )	June 26, 2021, 5:27 p.m.	Route de la Dent, Le Pont, Vaud, CH	speedounours	unknown	<a href="#">84578058</a>
Globe-flowered Orchid ( <a href="#">Traunsteinera globosa</a> )	June 26, 2021, 5:20 p.m.	Le Vaud, Le Vaud, Vaud, CH	manuelramalho	unknown	<a href="#">84577211</a>
chicory ( <a href="#">Cichorium intybus</a> )	June 26, 2021, 4:02 p.m.	Schubertstrasse, Zürich, Zürich, CH	jreim	unknown	<a href="#">84613125</a>
Red Helleborine ( <a href="#">Cephalanthera rubra</a> )	June 26, 2021, 4:01 p.m.	3087 Niedermühlern, Switzerland	agosti	unknown	<a href="#">84600371</a>
cypress spurge ( <a href="#">Euphorbia cyparissias</a> )	June 26, 2021, 3:47 p.m.	1902 Evionnaz, Suisse	sabinecrettenand	unknown	<a href="#">84546833</a>

(b) Web page listing observations

Figure 26: Data of iNATURALIST visually displayed on our platform

plants that are most likely to grow and flower in a particular area and at a specific moment of time, using the available flora data that we have integrated previously. To this end, we defined a set of criteria for matching plants and assigned a score to each one of them. As shown in Table 4, not every criterion receives equal weight.

Plants that potentially occur within a radius of 5km of the specified location (*Occ.*) and that potentially flower at the specified time (*Flow.*) are given the highest weight of 10. Occurrence data derives from INFO FLORA, while flowering data comes from both iNATURALIST and PRITSCH. If either of those (or both) mark the plant as flowering, this criterion is matched. The reason we combine both sources is that they complement each other. In general, iNATURALIST covers more plants



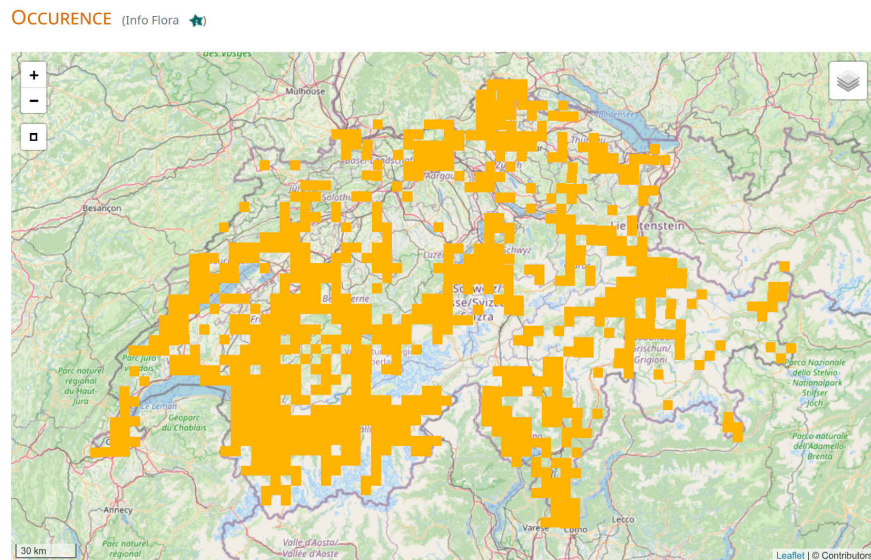


Figure 27: Plant occurrence for *Hepatica nobilis* displayed on a map from OPEN-STREETMAP [61] on our platform. Each yellow square stands for a 5km x 5km region in which the plant occurs.



Figure 28: Food abundance for *Trifolium repens* as shown on our platform.

with flowering status, but lacks them for some common trees producing pollen which in turn PRITSCH includes, for instance *Fagus sylvatica* (European beech).

Additional scores are attributed when a plant was observed in the vicinity of 3km of the specified place (*Obsv.*), especially when discovered within the previous and following 7 days of the specified month and day in any year (*Obsv./time*), in the same year (*Obsv./year*), within the previous and following 7 days of the specified month and day in that year (*Obsv./year/time*) and when explicitly annotated as flowering (*Obsv./year/time/flow.*). All this information stems from iNATURALIST.

Finally, we also award scores when the plant has a pollen nutrition value or color specified. The rationale behind this is that these are the more common plants



Criterion	Score
Occurring (Occ.)	10
Flowering (Flow.)	10
Observed (Obsv.)	1
Observed around this time (Obsv./time)	1
Observed in this year (Obsv./year)	1
Observed in this year around this time (Obsv./year/time)	1
Observed in this year around this time flowering (Obsv./year/time/flow.)	1
Pollen nutrition grade (Grade)	2
Pollen color (Color)	2

Table 4: Plant matching criteria with their scores. Abbreviations for the criteria are put in parentheses.

known to produce pollen for bees. Note that our plant database also contains rare plants that are accordingly less likely to be visited by bees, but also plants that honey bees do not pollinate at all. To give such plants not too much weight when they are observed, a match for *Grade* and/or *Color* is worth a score of 2.

The scores of the matched criteria for a plant are summed up to a total score, which indicates the potential of bees bringing pollen from this plant into the hive in that particular area at the given time. These total scores then serve as ranking for the plants where plants with equal total scores are subsumed under the same rank.

Figure 29 shows how the ranking is represented on the platform. As can be seen in Figure 29a, the user has the possibility to tweak several parameters, the most important ones being the location and the time, but also the radius and the acceptable time range for matching observations.

### 6.3 Evaluation

To verify how accurate the plant-matching algorithm is, we collected pollen from the same hive at *Döttingen* on two different warm and dry days. As bee traffic varies and plants deliver pollen at different times of the day [51], we collected the pollen throughout the day (between 9am-4pm). We also experimented with different collection lengths (2h, 1h, 15min, 5min, 1min) as bee traffic may strongly fluctuate, even within a quarter of an hour.

For retaining the pollen, we used a hive that was equipped with a pollen trap. At the start of a collection session, we inserted the empty pollen container into the hive. When the time was up, we pulled out the pollen container and tipped the contents into cans, labeling them with the start and end time. We dried (and froze) the pollen samples for several days. To keep the costs for analysis at bay, we drew one pollen sample from each collection day. We then sent both pollen samples to a palynology specialist for analysis. In total, 364 pollen pellets were

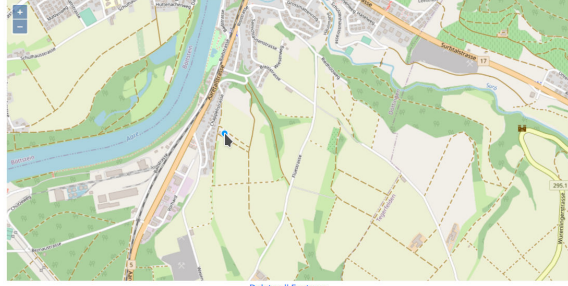
CHECK BEE FOOD AVAILABILITY CHECK FOOD

Radius (km):

Time:

Days (+/-):

Search by Map Delete all Features



(a) Search parameters for plant matching. Location is indicated on a map from OPEN-STREETMAP [61].

PLANT MATCHING HOW RANKING WORKS

Total matched plants: 912

#	Plant	Occ.	Flow.	Obsv.	Obsv./time	Obsv./year	Obsv./year/time	Obsv./year/time/flow.	Grade	Color
Rank: 1		Score: 29							Count: 8	
1	<a href="#">Cornus sanguinea</a>	✓	✓	✓	✓	✓	✓	✓	3	🟢
2	<a href="#">Ligustrum vulgare</a>	✓	✓	✓	✓	✓	✓	✓	3	🟡
3	<a href="#">Onobrychis vicifolia</a>	✓	✓	✓	✓	✓	✓	✓	5	🔴
4	<a href="#">Robinia pseudoacacia</a>	✓	✓	✓	✓	✓	✓	✓	5	🟠
5	<a href="#">Salix alba</a>	✓	✓	✓	✓	✓	✓	✓	4	🟡
6	<a href="#">Syringa vulgaris</a>	✓	✓	✓	✓	✓	✓	✓	3	🟡
7	<a href="#">Trifolium pratense</a>	✓	✓	✓	✓	✓	✓	✓	4	🔴
8	<a href="#">Trifolium repens</a>	✓	✓	✓	✓	✓	✓	✓	5	🔴
Rank: 2		Score: 28							Count: 1	
9	<a href="#">Acer pseudoplatanus</a>	✓	✓	✓	✓	✓	✓	✗	5	🟢
Rank: 3		Score: 27							Count: 9	
10	<a href="#">Centaurea jacea</a>	✓	✓	✓	✓	✓	✓	✓	4	
11	<a href="#">Dianthus carthusianorum</a>	✓	✓	✓	✓	✓	✓	✓	2	
12	<a href="#">Fragaria vesca</a>	✓	✓	✓	✓	✓	✓	✓	1	
13	<a href="#">Geranium pyrenaicum</a>	✓	✓	✓	✓	✓	✓	✓	3	
14	<a href="#">Knautia arvensis</a>	✓	✓	✓	✓	✓	✓	✓	1	

(b) Results of plant matching.

Figure 29: Plant matching visualized on platform.

analyzed. Most pollen pellets could not be mapped to a plant species, but only to higher taxonomic ranks such as genus or family. Henceforth, we call such a mapping – be it to a plant species, genus or some other rank – a plant type. There were also some cases of doubts where the palynology expert declared two plant types. In those cases, we tried to assign them to the next higher rank. Plant types above family rank were discarded, however, as this would match too many plants. Table 5 summarizes the two samples.

Date	Time	Pellets	Plant types	Species	Genera	Families
4th May 2021	13:10-13:25	195	15	3	9	3
1st June 2021	13:30-13:35	169	20	5	11	4
Total		364	35	8	20	7

Table 5: The two analyzed pollen samples

On each collection day, we also took images of plants in the surrounding area of the hive and added the observations to iNATURALIST. As the foraging area of bees is around 3km and only two of us observed the area, we could not cover the whole area. However, we observed plants from different biotopes. In the case of the hive in Döttingen, we took images on (agricultural) fields, in the forest and at the river in a nature protection area, assuming that different plants potentially grow in different biotopes. In total, we made 56 and 84 observations on the first and second day, covering 33 and 45 species, respectively.

To test the performance of the plant-matching algorithm, we used recall, precision and f1-score as evaluation metrics. As our plant-matching algorithm returns a list of plant species (unlike the pollen analysis that often classifies pollen into higher taxonomic ranks), we proceeded as follows: For each plant type found in the pollen analysis, we checked whether our algorithm found a species that is a descendant of or equal to the plant type (recall). Vice versa, for each matched plant species, we verified whether the pollen analysis yielded a plant type that is an ancestor of or equal to the plant species.

### 6.3.1 Results

Figures 30 and 31 show the results for the first and second pollen sample, respectively. For each sample, we computed the evaluation metrics for different minimum total scores to assess the best trade-off between precision and recall.

For pollen sample 1, the best f1-score was 56.9% when setting the minimum total score to 24 for the plant-matching, while for pollen sample 2, it was 25, achieving a f1-score of 65.3%. In both cases, recall drops the more restrictive the plant-matching is (i.e. the greater the minimum total score), whereas precision tends to rise. The plant-matching algorithm generally performed better on the second pollen sample than the first one (+8.4%) due to higher precision values.

### 6.3.2 Discussion

Our plant-matching achieves an average f1-score of 61.1% across both samples. Table 6 shows the plants missed by the algorithm for the best minimum total score of both pollen samples.

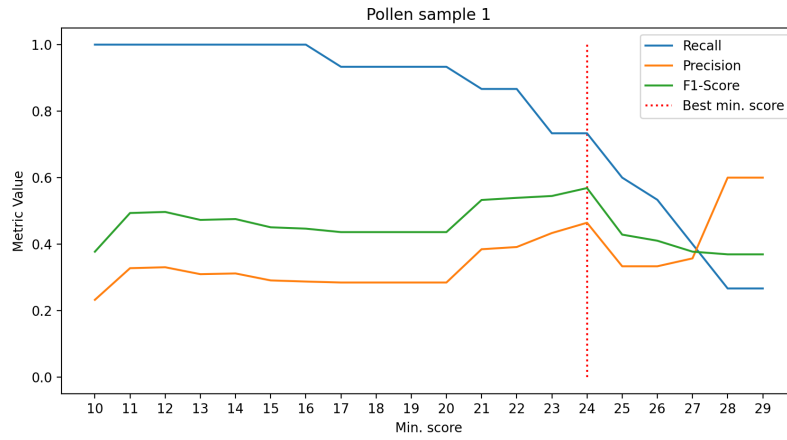


Figure 30: Results for pollen sample 1.

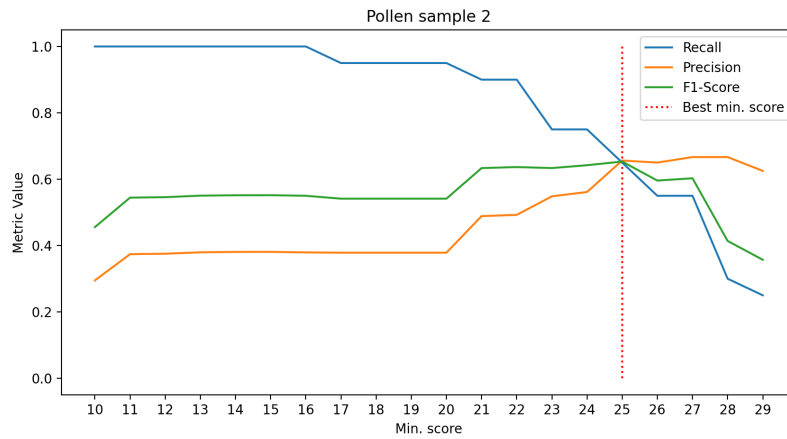


Figure 31: Results for pollen sample 2.

When we photographed the area with iNATURALIST, we apparently missed some of the plants found in the pollen samples (*Ligustrum*, *Rubus*, *Lonicera*, *Viola*, *Helianthemum* and *Campanula*). The simplest explanation of why we missed them is that we could not cover the whole area. For instance, many plants also grew in nearby private gardens that we could not enter. Furthermore, we most likely overlooked some of the plants. This was, for example, the case for *Ligustrum* which we did not observe on the first collection day, but on the second. Our palynology specialist also found pellets of plants that are rarely observed in general, e.g. there are only 120 observations of *Helianthemum* in Switzerland with research grade.

For the rest of the false negative cases, we observed the plant in question, but

Sample(s)	Rank	Name
1	Genus	Ligustrum
1	Species	Sanguisorba minor
1	Species	Ranunculus acris
1/2	Genus	Lonicera
2	Genus	Rosa
2	Genus	Rubus
2	Genus	Viola
2	Genus	Helianthemum
2	Genus	Campanula
2	Species	Brassica napus

Table 6: False negatives of the plant-matching algorithm.

it did not enter into our database because the observation did not obtain research grade. Plants that are annotated as cultivated (e.g. *Brassica napus*) or were identified by less than two people (e.g. *Sanguisorba minor*) are not awarded research grade in iNATURALIST. We also observed many plants of genus *Ranunculus*. Since they have a great number of species with similar features, they are difficult to identify and are left on genus level, in which case the observation also does not receive research grade. This is clearly one point to improve upon. Instead of filtering out all observations without research grade, we could still include these observations in our database, possibly attaching a weight to those in doubt.

Regarding precision, it must be noted that, firstly, there is a degree of uncertainty added through the pollen collection and analysis itself. Not all pollen pellets are retained by pollen traps, and of those that are retained, not all were analysed under the microscope. Therefore, our plant algorithm might have matched some plant types, whose pollen were actually brought in by the bees, but were not found in the pollen analysis. As an example, our palynology expert noted down two other plant types whose pellets were not sorted out for analysis, but were found in traces on other pellets of different plant types. We plan to back up the microscope analysis through an additional DNA analysis of the same pollen samples in the future.

Secondly, our two pollen samples are not representative for the whole day. We only collected pollen for a period of 5 and 15 minutes. There is, however, great variability in the pollen intake throughout the day. Firstly, different plants release pollen at different times of day. There are plants serving pollen in the morning (e.g. *Papaver rhoeas*), in the afternoon (e.g. *Vicia faba*), during the night (e.g. pumpkins) or also throughout the day (e.g. fruit trees) [51]. Furthermore, the release duration also differs, with some plants emptying their pollen supply gradually and others in a very short time (sometimes within 10-30 minutes) [51]. Lastly, plants react to weather stimuli. For example, pines produce less pollen at

low temperatures [51].

Currently, the plant-matching algorithm does not consider such factors as we do not have the data for it yet. As it stands, the plant-matching works best at a day-level precision without making differences within the times of a day. Therefore, it would have been more ideal to compare the plant-matching algorithm against pollen samples gathered from a whole day. There are two reasons why we abandoned this idea in the first place. Firstly, analyzing a daily pollen sample costs considerably more. Secondly, we also wanted to combine the plant data with the video data. Each pollen sample was collected under video surveillance. The initial goal was to match the colors seen in the camera with the colors in the plant database to quantify how much pollen of which plant is brought into the hive, and evaluate this against our analyzed pollen sample. Due to the limited scope of this work, this was not pursued in the end. Nevertheless, the color matching is a promising future work, albeit a demanding one. Before we sent the pollen sample to the palynology expert, one of us attempted to sort them by colors. As it turned out, our expert disagreed with the sorting by a substantial margin. This suggests that even for humans it is difficult to discern pollen colors consistently and that chromatic assessment as put forward by [17] may fail due to perceptual differences or different lighting conditions. It remains to be seen whether this also applies for computers.

That being said, a highly interesting avenue for future research is to systematically record from which plants at which times of the day bees collect pollen. This could be realized, for instance, through a citizen approach where people report for an observation made on iNATURALIST whether bees collected pollen. This way, we can also establish which plants are actually pollinated by bees. Currently, our system only knows for a few hundred plants from PRITSCH and SCHULBIOLOGIEZENTRUM HANNOVER) that they produce pollen for bees. However, our plant database also contains plants that are pollinated by wind and are therefore erroneously considered by the plant-matching algorithm as well.

In conclusion, there is still much potential for improving the plant-matching algorithm. We did not perform extensive experiments with the plant-matching algorithm in general. For example, we did not run the evaluation against different weights for the various criteria. Furthermore, the algorithm also relies on observations made with iNATURALIST at a given place and time, and was also evaluated under such conditions. As such, there is a manual element involved to obtain accurate results with our algorithm. Furthermore, observations may have an inherent skew. In discussion with the INFO FLORA team, they mentioned that people have a tendency to observe more rare plants than the more common ones. This is problematic for us as bees are more likely to bring home pollen from plants that occur in abundance.

The future vision is that our plant-matching algorithm also works reasonably well when no one makes observations for a specific place and time. Instead, this should be deduced based on observations made in other areas with similar environmental conditions. One first step towards this is to integrate more data on

plant flowering status. Blooming phases are heavily influenced through various variables such as location, time and weather. This is also strongly reflected in the fact that different works of reference disagree in the exact flowering phase of plants. For instance for *Onobrychis viciifolia*, PRITSCH specifies a flowering phase from May to July and FLORA HELVETICA from May to August [48], while on iNATURALIST the earliest observation in Switzerland is in April and the latest in October. Although, iNATURALIST may provide us the exact flowering status of a plant for a place, the coverage is generally too low. Integrating the data of PHAENONET which also correlates phenology with weather and other environmental parameters allows us to deduce the flowering status of a particular place even if that plant has not been explicitly observed there.

## 7 Conclusion & Outlook

In this thesis, two pollen metrics were integrated into the BEE-LIVING-SENSOR platform that act as an indicator for biodiversity and bee health. As it stands, the pollen count metric does not deliver precise results yet. This is mostly due to the poor performance of the bee detection and bee tracking on which the pollen tracker relies. On the other hand, we could show that the pipeline for computing the pollen counts runs stable, distributing the processing work evenly across CPUs and GPUs. However, the speed for computing these is still too slow, incurring high operational costs. One way to address this in the future is to reduce the amount of data to be processed as well as optimizing the code and models.

In comparison, the pollen diversity metric already produces acceptable results. Deviations from the microscopy analysis were mostly due to the lack of data, especially regarding the time of when plants release pollen. Fortunately, data on plants is fast increasing, strongly aided by the citizen science community, and therefore we expect our plant-matching algorithm to get stronger on its own.

Apart from more data on plants, other kinds of data sources shall be incorporated into the platform in the future, in particular data from hive sensors and agriculture. Furthermore, we plan to enhance the visualization of the data on our platform, providing first insights into the influence of certain parameters on the life of bee colonies.

## Bibliography

- [1] Korbinian Abstreiter, Andrius Buinovskij, and Simon Böhm. Pollen Detection for the BeeLivingSensor Platform. Data Science Lab, Swiss Federal Institute of Technology in Zurich, 2021.
- [2] Agroscope. *Wichtige Pollen- und Nektarquellen für die Honigbienen in der Schweiz*. Agroscope, 2020.
- [3] Cédric Alaux, François Ducloz, Didier Crauser, and Yves Le Conte. Diet effects on honeybee immunocompetence. *Biology letters*, 6(4):562–565, 2010.
- [4] Apis und Landwirtschaftskammer Nordrhein-Westfalen. Die Honigmacher: Pollen-Bestimmung. <https://www.die-honigmacher.de/kurs2/pollenresult.html>. [Online; accessed 2021-04-01].
- [5] Zdenka Babic, Ratko Pilipovic, Vladimir Risojevic, and Goran Mirjanic. Pollen bearing honey bee detection in hive entrance video recorded by remote embedded system for pollination monitoring. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:51, 2016.
- [6] Oliver Batchelor. GitHub issue: TypeError: can't pickle cv2.VideoCapture objects. <https://github.com/MVIG-SJTU/AlphaPose/issues/128>. [Online; accessed 2021-05-13].
- [7] BeeKeepr team. PlanBee-Project - Tools für Imker und Gärtner. <https://planbee-project.com/>. [Online; accessed 2021-04-08].
- [8] Madeleine Beekman and FLW Ratnieks. Long-range foraging by the honeybee, *apis mellifera* l. *Functional Ecology*, 14(4):490–496, 2000.
- [9] Keni Bernardin, Alexander Elbs, and Rainer Stiefelhagen. Multiple object tracking performance metrics and evaluation in a smart room environment. In *Proceedings of the Sixth International IEEE Workshop on Visual Surveillance*, volume 90, page 91. IEEE, 2006.
- [10] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.
- [11] bienformatik. Trachtpflanzen. <https://www.trachtfliessband.de/>. [Online; accessed 2021-06-30].
- [12] Robert Brodschneider and Karl Crailsheim. Nutrition and health in honey bees. *Apidologie*, 41(3):278–294, 2010.
- [13] Robert Brodschneider, Kristina Gratzner, Elfriede Kalcher-Sommersguter, Helmut Heigl, Waltraud Auer, Rudolf Moosbeckhofer, and Karl Crailsheim.



- A citizen science supported study on seasonal diversity and monoflorality of pollen collected by honey bees in Austria. *Scientific reports*, 9(1):1–12, 2019.
- [14] Rusty Burlew. HoneyBeeSuite: Pollen traps require constant attention. <https://www.honeybeesuite.com/pollen-traps-require-constant-attention/>, 06 2010. [Online; accessed 2021-04-01].
- [15] Celery. A Distributed Task Queue. <https://docs.celeryproject.org/en/stable/index.html>. [Online; accessed 2021-06-10].
- [16] Benoit Chesneau. Unicorn. <https://gunicorn.org/>. [Online; accessed 2021-06-10].
- [17] Ida Conti, Piotr Medrzycki, Francesca V Grillenzoni, Francesca Corvucci, Simone Tosi, Valeria Malagnini, Martina Spinella, and Mauro G Mariotti. Floral diversity of pollen collected by honey bees (L.) – validation of the chromatic assessment method. *Journal of Apicultural Science*, 60(2):209–220, 2016.
- [18] Samantha M Cook, Caroline S Awmack, Darren A Murray, and Ingrid H Williams. Are honey bees’ foraging preferences affected by pollen amino acid composition? *Ecological Entomology*, 28(5):622–627, 2003.
- [19] K Crailsheim, LHW Schneider, N Hrassnigg, N Bühlmann, U Brosch, R Gmeinbauer, and B Schöffmann. Pollen consumption and utilization in worker honeybees (*Apis mellifera carnica*): Dependence on individual age and function. *Journal of Insect Physiology*, 38(6):409–419, 1992.
- [20] Karl Crailsheim. Interadult feeding of jelly in honeybee (*Apis mellifera* L.) colonies. *Journal of Comparative Physiology B*, 161(1):55–60, 1991.
- [21] Antonius P de Groot. Amino acid requirements for growth of the honeybee (*Apis mellifica* L.). *Experientia*, 8(5):192–194, 1952.
- [22] Garance Di Pasquale, Cédric Alaux, Yves Le Conte, Jean-François Odoux, Maryline Pioz, Bernard E Vaissière, Luc P Belzunces, and Axel Decourtye. Variations in the availability of pollen resources affect honey bee health. *PloS one*, 11(9):e0162818, 2016.
- [23] Garance Di Pasquale, Marion Salignon, Yves Le Conte, Luc P Belzunces, Axel Decourtye, André Kretzschmar, Séverine Suchail, Jean-Luc Brunet, and Cédric Alaux. Influence of pollen nutrition on honey bee health: do pollen quality and diversity matter? *PloS one*, 8(8):e72016, 2013.
- [24] Nicholas Dykeman and Dominique Heyn. Using Deep Learning for Non-Invasive Pollen Detection on Honey Bees. Data Science Lab, Swiss Federal Institute of Technology in Zurich, 2019.

- [25] Pascal Engeli and Gabriela López Magaña. Bee tracking in videos. Platform design specifications and deep learning algorithms. Master's project, University of Zurich, 2021.
- [26] Hannah Feltham, Kirsty Park, and Dave Goulson. Field realistic doses of pesticide imidacloprid reduce bumblebee pollen foraging efficiency. *Ecotoxicology*, 23(3):317–323, 2014.
- [27] Jochen Fischer. Imkerei Fischer: Pollenfarben und Trachtpflanzen. <https://fischerjochen.jimdofree.com/pollenfarben-und-trachtpflanzen/>. [Online; accessed 2021-04-01].
- [28] Eva Forsgren. European foulbrood in honey bees. *Journal of invertebrate pathology*, 103:S5–S9, 2010.
- [29] Laura Fortel, Mickaël Henry, Laurent Guilbaud, Anne Laure Guirao, Michael Kuhlmann, Hugues Mouret, Orianne Rollin, and Bernard E Vaissière. Decreasing abundance, increasing diversity and changing structure of the wild bee community (hymenoptera: Anthophila) along an urbanization gradient. *PloS one*, 9(8):e104679, 2014.
- [30] Nicola Gallai, Jean-Michel Salles, Josef Settele, and Bernard E Vaissière. Economic valuation of the vulnerability of world agriculture confronted with pollinator decline. *Ecological economics*, 68(3):810–821, 2009.
- [31] Richard J Gill, Oscar Ramos-Rodriguez, and Nigel E Raine. Combined pesticide exposure severely affects individual-and colony-level traits in bees. *Nature*, 491(7422):105–108, 2012.
- [32] Jan Gąsienica-Józkowy. GitHub issue: How to calculate various model speed in FPS for my GPU? <https://github.com/AlexeyAB/darknet/issues/4627>. [Online; accessed 2021-05-10].
- [33] Caspar A Hallmann, Martin Sorg, Eelke Jongejans, Henk Siepel, Nick Hofland, Heinz Schwan, Werner Stenmans, Andreas Müller, Hubert Sumser, Thomas Hörren, Dave Goulson, and Hans de Kroon. More than 75 percent decline over 27 years in total flying insect biomass in protected areas. *PloS one*, 12(10):e0185809, 2017.
- [34] Mickaël Henry, Maxime Beguin, Fabrice Requier, Orianne Rollin, Jean-François Odoux, Pierrick Aupinel, Jean Aptel, Sylvie Tchamitchian, and Axel Decourtye. A common pesticide decreases foraging success and survival in honey bees. *Science*, 336(6079):348–350, 2012.
- [35] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio

- Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3296–3297. IEEE Computer Society, 2017.
- [36] Thomas Hudson. Easy Bee Counter. <https://github.com/hydrronics2/2019-easy-bee-counter>. [Online; accessed 2021-04-09].
- [37] Imkerverein Balingen-Geislingen-Rosenfeld. Pollenfarben. <https://www.honigboerse.de/wissenswertes/pollenfarben>. [Online; accessed 2021-04-01].
- [38] Imkerverein Klosterneuburg. Pollenfarben. <https://www.imker-klosterneuburg.at/index.php/infos/18-pollenfarben>. [Online; accessed 2021-04-01].
- [39] iNaturalist. A community for Naturalists. <https://www.inaturalist.org/>. [Online; accessed 2021-04-08].
- [40] Info Flora. The National Data and Information Center on the Swiss Flora. <https://www.infoflora.ch/en/>. [Online; accessed 2021-04-08].
- [41] Antoine Jacques, Marion Laurent, Epilobee Consortium, Magali Ribière-Chabert, Mathilde Saussac, Stéphanie Bougeard, Giles E Budge, Pascal Hendrikx, and Marie-Pierre Chauzat. A pan-european epidemiological study reveals honey bee colony survival depends on beekeeper education and disease control. *PLoS one*, 12(3):e0172591, 2017.
- [42] Anna Jancso and Vladimir Masarik. Platform for Bee Tracking in Videos. Master’s Project, University of Zurich, 2020.
- [43] Benjamin F Kaluza, Helen M Wallace, Tim A Heard, Vanessa Minden, Alexandra Klein, and Sara D Leonhardt. Social bees are fitter in more biodiverse environments. *Scientific reports*, 8(1):1–10, 2018.
- [44] Irene Keller, Peter Fluri, and Anton Imdorf. Pollen nutrition and colony development in honey bees: part 1. *Bee world*, 86(1):3–10, 2005.
- [45] Christina M Kennedy, Eric Lonsdorf, Maile C Neel, Neal M Williams, Taylor H Ricketts, Rachael Winfree, Riccardo Bommarco, Claire Brittain, Alana L Burley, Daniel Cariveau, et al. A global quantitative synthesis of local and landscape effects on wild bee pollinators in agroecosystems. *Ecology letters*, 16(5):584–599, 2013.
- [46] Kubernetes. Schedule GPUs. <https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/>. [Online; accessed 2021-06-10].
- [47] Karl Kunert and Karl Crailsheim. Seasonal changes in carbohydrate, lipid and protein content in emerging worker honeybees and their mortality. *Journal of Apicultural Research*, 27(1):13–21, 1988.

- 
- [48] Konrad Lauber, Gerhart Wagner, and Andreas Gygax. *Flora Helvetica – Illustrierte Flora der Schweiz*. Haupt Verlag, 2018.
- [49] WM Lauter and VL Vrla. Factors influencing the formation of the venom of the honeybee. *Journal of Economic Entomology*, 32(6):806–807, 1939.
- [50] Julian Marstaller, Frederic Tausch, and Simon Stock. Deepbees - building and scaling convolutional neuronal nets for fast and large-scale visual monitoring of bee hives. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 271–278, 2019.
- [51] Anna Maurizio and Friedgard Schaper. *Das Trachtplanzenbuch: Nektar und Pollen - die wichtigsten Nahrungsquellen der Honigbiene*. Ehrenwirth, 1994.
- [52] MeteoSchweiz. Phänologisches Beobachtungsnetz. <https://www.meteoschweiz.admin.ch/home/mess-und-prognosesysteme/bodenstationen/phaenologisches-beobachtungsnetz.html>. [Online; accessed 2021-07-10].
- [53] Susan Wendy Nicolson and Hannelie Human. Chemical composition of the ‘low quality’ pollen of sunflower (*helianthus annuus*, asteraceae). *Apidologie*, 44(2):144–152, 2013.
- [54] NVIDIA. CUDA. <https://developer.nvidia.com/cuda-zone>. [Online; accessed 2021-06-10].
- [55] NVIDIA. CUDA and cuDNN Docker images. <https://hub.docker.com/r/nvidia/cuda>. [Online; accessed 2021-06-10].
- [56] NVIDIA. cuDNN. <https://developer.nvidia.com/cudnn>. [Online; accessed 2021-06-10].
- [57] NVIDIA. Device plugin for Kubernetes. <https://github.com/NVIDIA/k8s-device-plugin>. [Online; accessed 2021-06-10].
- [58] NVIDIA. NVIDIA Container Toolkit. <https://github.com/NVIDIA/nvidia-docker>. [Online; accessed 2021-06-10].
- [59] Eslam Omar, Aly A Abd-Ella, Mohammed M Khodairy, Rudolf Moosbeckhofer, Karl Crailsheim, and Robert Brodschneider. Influence of different pollen diets on the development of hypopharyngeal glands and size of acid gland sacs in caged honey bees (*apis mellifera*). *Apidologie*, 48(4):425–436, 2017.
- [60] OpenCV. Open Source Computer Vision Library. <https://github.com/opencv>. [Online; accessed 2021-06-10].
- [61] OpenStreetMap contributors. Planet dump retrieved from <https://planet.osm.org>. <https://www.openstreetmap.org>, 2017.

- [62] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
- [63] Robert J Paxton. Does infection by *nosema ceranae* cause “colony collapse disorder” in honey bees (*apis mellifera*)? *Journal of Apicultural Research*, 49(1):80–84, 2010.
- [64] Pl@ntNet. Pl@ntNet. <https://identify.plantnet.org/de>. [Online; accessed 2021-07-10].
- [65] Caitlin Potter, Natasha De Vere, Laura E Jones, Col R Ford, Matthew J Hegarty, Kathy H Hodder, Anita Diaz, and Elizabeth L Franklin. Pollen metabarcoding reveals broad and species-specific resource use by urban bees. *PeerJ*, 7:e5999, 2019.
- [66] Günter Pritsch. *Bienenweide: 200 Trachtplanzen erkennen und bewerten*. Kosmos, 2018.
- [67] Sebastián Ramírez. FastAPI. <https://github.com/tiangolo/fastapi>. [Online; accessed 2021-06-10].
- [68] Sovit Rath. Deep Learning with OpenCV DNN Module: A Definitive Guide. <https://learnopencv.com/deep-learning-with-opencvs-dnn-module-a-definitive-guide/#object-detection-using-opencv-dnn>. [Online; accessed 2021-05-02].
- [69] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
- [70] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.
- [71] Rodney T Richardson, Hailey R Curtis, Emma G Matcham, Chia-Hua Lin, Sreelakshmi Suresh, Douglas B Sponsler, Luke E Hearon, and Reed M Johnson. Quantitative multi-locus metabarcoding and waggle dance interpretation reveal honey bee spring foraging patterns in midwest agroecosystems. *Molecular ecology*, 28(3):686–697, 2019.
- [72] Joe Rickerby. pyinstrument. <https://github.com/joerick/pyinstrument>. [Online; accessed 2021-06-10].

- [73] WDJL Ride, HG Cogger, C Dupuis, O Kraus, A Minelli, FC Thompson, and PK Tubbs. *International code of zoological nomenclature*. The International Trust for Zoological Nomenclature, 1999.
- [74] Peter Rosenkranz, Pia Aumeier, and Bettina Ziegelmann. Biology and control of varroa destructor. *Journal of invertebrate pathology*, 103:S96–S119, 2010.
- [75] Justin O Schmidt. Feeding preferences of *apis mellifera* l.(hymenoptera: Apidae): individual versus mixed pollen species. *Journal of the Kansas Entomological Society*, pages 323–327, 1984.
- [76] Li S Schmidt, Justin O Schmidt, Hima Rao, Weiyi Wang, and Ligen Xu. Feeding preference and survival of young worker honey bees (hymenoptera: Apidae) fed rape, sesame, and sunflower pollen. *Journal of Economic Entomology*, 88(6):1591–1595, 1995.
- [77] Schulbiologiezentrum Hannover. Pollenkalender. <http://www.schulbiologiezentrum.info/Bienen.htm#pollenkalender>. [Online; accessed 2021-04-01].
- [78] Stack Overflow: user2386301. Python multiprocessing can't pickle opencv videocapture object. <https://stackoverflow.com/questions/53308334/python-multiprocess-cant-pickle-opencv-videocapture-object>. [Online; accessed 2021-05-13].
- [79] Zoran Stanimirović, Uroš Glavinić, Marko Ristanić, Nevenka Aleksić, Nemanja M Jovanović, Branislav Vejnović, and Jevrosima Stevanović. Looking for the causes of and solutions to the issue of honey bee colony losses. *Acta veterinaria-beograd*, 69(1):1–31, 2019.
- [80] Matthew Switanek, Karl Crailsheim, Heimo Truhetz, and Robert Brodschneider. Modelling seasonal effects of temperature and precipitation on honey bee winter mortality in a temperate climate. *Science of the Total Environment*, 579:1581–1587, 2017.
- [81] Frederic Tausch, Katharina Luise Schmidt, and Matthias Diehl. Current achievements and future developments of a novel ai based visual monitoring of beehives in ecotoxicology and for the monitoring of landscape structures. *BioRxiv*, 2020.
- [82] Technische Universität Ilmenau. Flora Incognita. <https://floraincognita.com/>. [Online; accessed 2021-04-08].
- [83] Kelton Temby, Nick Cunningham, Jon Simpson, and Scott Ross. Eyesonhives. <https://www.eyesonhives.com/>. [Online; accessed 2021-04-10].

- [84] Nick J Turland, John Harry Wiersema, Fred R Barrie, Werner Greuter, David L Hawksworth, Patrick Stephen Herendeen, Sandra Knapp, Wolf-Henning Kusber, De-Zhu Li, Karol Marhold, Tom W May, John McNeill, Anna M Monro, Jefferson Prado, Michelle J Price, and Gideon F Smith. *International Code of Nomenclature for algae, fungi, and plants (Shenzhen Code) adopted by the Nineteenth International Botanical Congress Shenzhen, China, July 2017*. Koeltz Botanical Books, 2018.
- [85] Adam J Vanbergen and the Insect Pollinators Initiative. Threats to an ecosystem service: pressures on pollinators. *Frontiers in Ecology and the Environment*, 11(5):251–259, 2013.
- [86] Verein GLOBE Schweiz. PhaenoNet. <https://www.phaenonet.ch/de/>. [Online; accessed 2021-04-08].
- [87] Meghan Vesey and Kwesi Haizel. Akesi Farms: Bee pollen chart. <https://www.akesifarms.com/library/bee-pollen-chart/>. [Online; accessed 2021-04-01].
- [88] P Kirk Visscher and Thomas D Seeley. Foraging strategy of honeybee colonies in a temperate deciduous forest. *Ecology*, 63(6):1790–1801, 1982.
- [89] Werner Von Der Ohe, Livia Persano Oddo, Maria Lucia Piana, Monique Morlot, and Peter Martin. Harmonized methods of melissopalynology. *Apidologie*, 35(Suppl. 1):S18–S25, 2004.
- [90] Keith D Waddington, Thomas J Herbert, P Kirk Visscher, and Monica Raveret Richter. Comparisons of forager distributions from matched honey bee colonies in suburban environments. *Behavioral Ecology and Sociobiology*, 35(6):423–429, 1994.
- [91] Wikipedia. List of pollen sources. [https://en.wikipedia.org/wiki/List\\_of\\_pollen\\_sources](https://en.wikipedia.org/wiki/List_of_pollen_sources). [Online; accessed 2021-04-01].
- [92] Carl R Woese, Otto Kandler, and Mark L Wheelis. Towards a natural system of organisms: proposal for the domains archaea, bacteria, and eucarya. *Proceedings of the National Academy of Sciences*, 87(12):4576–4579, 1990.
- [93] Ahmed Ali Zaytoon, Mitsuo Matsuka, and Masami Sasaki. Feeding efficiency of pollen substitutes in a honeybee colony: effect of feeding site on royal jelly and queen production. *Applied Entomology and Zoology*, 23(4):481–487, 1988.

## Glossary

AI	Artificial Intelligence
API	Application Programming Interface
BLS	BeeLivingSensor
CCD	Colony Collapse Disorder
CPU	Central Processing Unit
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheets
CSV	comma-separated values
ETH	Eidgenössische Technische Hochschule
FIFO	First-In-First-Out
FPS	Frames Per Second
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
I/O	Input/Output
IoT	Internet of Things
IoU	Intersection over Union
JSON	JavaScript Object Notation
mAP	mean Average Precision
REST	Representational State Transfer
SORT	Simple Online and Realtime Tracking
UI	User Interface
UZH	University of Zurich
VM	Virtual Machine