THESE

en vue de l'obtention du grade de Docteur, délivré par

I'ECOLE NORMALE SUPERIEURE DE LYON

Ecole Doctorale N°512
Ecole Doctorale en Informatique et Mathématiques de Lyon

Discipline : Informatique

Soutenue publiquement le 13/07/2023, par :

Hugo HADJUR

Designing and modeling sustainable, autonomous, smart, and energy efficient Internet of Things systems, applied to precision beekeeping

Conception et modélisation de systèmes connectés durables, autonomes, intelligents et à basse consommation énergétique appliqués à l'apiculture de précision

---

Devant le jury composé de :

Nathalie MITTON	Directrice de recherche, Université de Lille	Rapporteur
David PALMA	Professeur, NTNU	Rapporteur
Isabelle GUERIN LASSOUS	Professeure des universités, Université Lyon 1	Examinatrice
Patricia PASCAL-STOLF	Professeure des universités, Université Paul Sabatier	Examinatrice
Denis TRYSTRAM	Professeur des universités, Université Grenoble Alpes	Examinateur
Doreid AMMAR	Professeur des universités, aivancity Paris-Cachan	Examinateur
Laurent LEFEVRE	Chargé de recherche HDR, Ecole normale supérieure de Lyon	Directeur de thèse

Acknowledgments

To my thesis directors,
Laurent Lefèvre and Doreid Ammar.

To my thesis referees,
Nathalie Mitton and David Palma

To my thesis examiners,
Isabelle Guerin Lassous, Patricia Pascal-Stolf and Denis Trystram.

To Ecole Normale Supérieure de Lyon,
to Avalon team of the LIP laboratory,
and to aivancity School for Technology, Business & Society.

To my family,
in particular my parents, my sister and my partner.

To my friends.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Meaning
A2C	**A**dvantage **A**ctor **C**ritic
ACF	**A**uto**c**orrelation **F**unction
AEHSS	**A**utonomous **E**nergy-**H**arvesting **S**tationary IoT **S**ystems
AI	**A**rtificial **I**ntelligence
API	**A**pplication **P**rogramming **I**nterface
ARIMA	**A**uto**R**egressive **I**ntegrated **M**oving **A**verage
AUC	**A**rea **U**nder the **C**urve
CC	**C**ascade **C**lassification
CCD	**C**olony **C**ollapse **D**isorder
CNN	**C**onvolutional **N**eural **N**etwork
CPU	**C**entral **P**rocessing **U**nit
CWT	**C**ontinuous **W**avelet **T**ransform
DFA	**D**iscriminant **F**unction **A**nalysis
DPIV	**D**igital **P**article **I**mage **V**elocimetry
DWT	**D**iscrete **W**avelet **T**ransform
EAPBS	**E**nergy-**A**ware **P**recision **B**eekeeping **S**ystem
FFT	**F**ast **F**ourier **T**ransform
FHT	**F**ast **H**artley **T**ransform
FPGA	**F**ield-**P**rogrammable **G**ate **A**rray
FPS	**F**rame **p**er **S**econd
GMMs	**G**aussian **M**ixture **M**odels
GNN	**G**lobal **N**earest **N**eighbor
GPN	**G**enerative-**P**rediction **N**etwork
GPU	**G**raphics **P**rocessing **U**nit
HHT	**H**ilbert **H**uang **T**ransform
HIDS	**H**ybrid **I**ntensity **D**epth **S**egmentation
HMM	**H**idden **M**arkov **M**odels
HSV	**H**ue, **S**aturation, **V**alue
IoT	**I**nternet **o**f **T**hings
k-NN	**k**-**N**earest **N**eighbors
KNN	**k**-**N**eighbors **C**lassifier
LDA	**L**inear **D**iscriminant **A**nalysis
LPC	**L**inear **P**redictive **C**oding

Abbreviation	Meaning
LR	**L**ogistic **R**egressing
LSTM	**L**ong **S**hort-**T**erm **M**emory
MAA	**M**ovement **A**ctive **A**rea
MAPE	**M**ean **A**bsolute **P**ercentage **E**rror
MFCC	**M**el-**F**requency **C**epstral **C**oefficients
MHT	**M**ultiple **H**ypothesis **T**racking
ML	**M**achine **L**earning
MLP	**M**ultilayer **P**erceptron
MOG	**M**ixture **o**f **G**aussians
NN	**N**eural **N**etwork
OF	**O**ptical **F**low
ORB	**O**riented Fast and **R**otated **B**RIEF
PB	**P**recision **B**eekeeping
PCA	**P**rincipal **C**omponent **A**nalysis
POLS	**P**rocedure of the **O**ptimal **L**inear **S**moothing
PPO	**P**roximal **P**olicy **O**ptimization
RAM	**R**andom-**A**ccess **M**emory
RCNN	**R**egion-based **C**onvolutional **N**eural **N**etwork
RF	**R**andom **F**orests
RFR	**R**andom **F**orests **R**egressor
RL	**R**einforcement **L**earning
RL/CI	**R**egion **L**abeling/**C**olor **I**dentification
RMS	**R**oot **M**ean **S**quare
RPPO	**R**ecurrence **P**roximal **P**olicy **O**ptimization
SGD	**S**tochastic **G**radient **D**escent
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
SOM	**S**elf-**O**rganizing **M**aps
STFT	**S**hort-**T**ime **F**ourier **T**ransform
SURF	**S**peeded **U**p **R**obust **F**eatures
SVM	**S**upport **V**ector **M**achine
TRPO	**T**rust **R**egion **P**olicy **O**ptimization
t-SNE	**t-**Distributed **S**tochastic **N**eighbor **E**mbedding
VLAD	**V**ector of **L**ocally **A**ggregated **D**escriptors
3DFT	**3 D**imensional **F**ourier **T**ransform

Abstract

Designing and modeling sustainable, autonomous, smart, and energy efficient Internet of Things systems, applied to precision beekeeping

Precision beekeeping aims to preserve honey bee colonies while supporting beekeepers in their work to develop colonies and produce resources, including honey. This topic naturally mixes apidology with several major computing disciplines: the Internet of Things, data science, and often, artificial intelligence. Precision beekeeping, which developed in the 2010s, sees the emergence of solutions thought under constraints, especially autonomy and energy efficiency because of the isolation of the deployed systems.

This thesis is part of the trend that aims to make digital technology greener by designing and modeling connected systems addressing precision beekeeping issues while focusing on energy efficiency.

First, a literature review is presented. This review is the only one in this field which covers the system architecture of a connected hive, but also all the services present in the literature, especially those using artificial intelligence. It raises questions and lists the challenges for future research in this field. Analyses of energy measurements are then shown to justify the following contribution: an autonomous and energy-efficient precision beekeeping system, that collects both beekeeping data and its own energy production and consumption. A generalization of this system is modeled to analyze different resource allocation scenarios. This simulation is used to design constellation-based Internet of Things systems and to determine the best scenario, between edge and cloud. Finally, a reinforcement learning method integrating a photovoltaic energy production prediction model is shown to be an effective answer to the question of task allocation under a limited energy budget for stationary autonomous energy-harvesting systems.

Résumé

Conception et modélisation de systèmes connectés durables, autonomes, intelligents et à basse consommation énergétique appliqués à l'apiculture de précision

L'apiculture de précision vise à préserver les colonies d'abeilles domestiques tout en épaulant les apiculteurs dans leur travail de développement des abeilles et de production de ressources, dont le miel. Par sa nature, ce sujet force à lier l'apidologie avec plusieurs grandes disciplines informatiques : l'Internet des objets, la science des données, et souvent, l'intelligence artificielle. L'apiculture de précision, qui s'est développée dans les années 2010, voit émerger des solutions pensées sous contraintes, notamment celles qui prennent forme à cause de l'isolement des systèmes déployés sur le terrain : l'autonomie et l'efficacité énergétique.

Cette thèse s'inscrit dans le courant qui vise à rendre le numérique plus vert en concevant et en modélisant des systèmes connectés qui répondent aux questions de l'apiculture de précision, tout en se concentrant sur des questions autour de l'énergie.

D'abord, une revue bibliographique est présentée. Cette revue est la seule du domaine de l'apiculture de précision qui couvre l'architecture système d'une ruche connectée, mais aussi tous les services présents dans la littérature, notamment ceux qui utilisent de l'intelligence artificielle. Elle permet de soulever des questions et lister les défis qui attendent la recherche à venir dans ce domaine. Des analyses de mesures énergétiques sont ensuite montrées afin de justifier la contribution suivante : un système d'apiculture de précision autonome et efficace en énergie, collectant à la fois des données apicoles, et sa propre production et consommation énergétique. Une généralisation de ce système est modélisée de façon à analyser plusieurs scénarios d'allocation de ressources matérielles. Cette simulation permet de concevoir des systèmes de l'Internet des objets par constellation et de déterminer le meilleur scénario, entre edge et cloud. Finalement, une méthode d'apprentissage par renforcement intégrant un modèle de prédiction de production d'énergie photovoltaïque se montre être une réponse efficace à la question de la répartition de tâches sous un budget énergétique restreint et pour tous les systèmes autonomes stationnaires produisant leur propre énergie.

Introduction

Contents for chapter 1

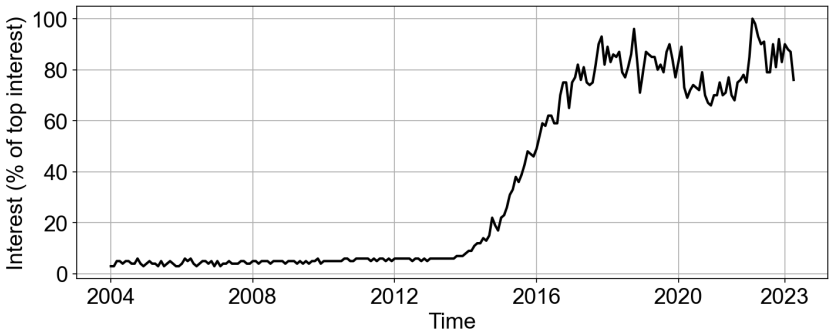
1.1 The intersection of data, IoT, AI, smart agriculture and energy

Data has always had a crucial role in the progress of science and more generally, the development of humankind. Whether it was engraved on rocks and metal plates or written on papyrus and paper, our ancestors passed along information by keeping their knowledge on long-lasting materials. The miniaturization of storage methods and hardware as a result of the recent rise of information technology and the development of storage devices have allowed for storing even more content, and today, every owner of a digital device can create and keep data.

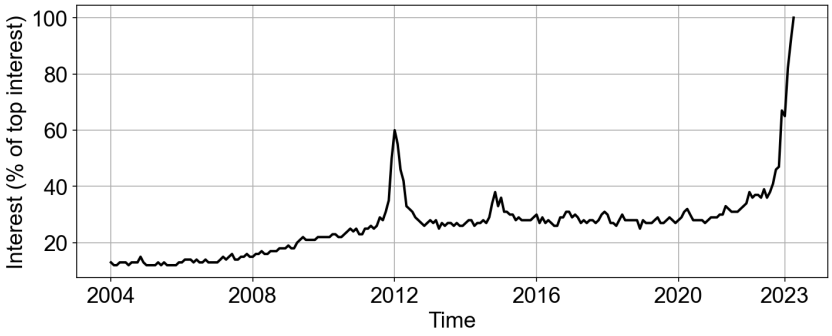
The process of data collection has also greatly evolved in the last few decades, especially with the contribution of the field of the Internet of Things (IoT). Thanks to sensors, some computation capabilities and some storage, automatic data collection can easily be implemented at affordable costs and every scale. Over the last decade, IoT's popularity skyrocketed (Figure 1.1a). This field found applications in diverse domains, including agriculture.

Using sensors, drones, robots or geolocation systems helps to better understand farming. Thus, it can increase the quality and the quantity of the production, all the while decreasing long-term costs. Optimizing irrigation (in [Corbari and Mancini, 2022]), predicting production (in [Midtiby and Pastucha, 2022]), detecting health issues (in [Prescott et al., 2023]) or monitoring the growth of plants and animals (in [Ulfa et al., 2022]) are, among others, the services that IoT systems bring to agriculture.

These services are made possible by decision models which process raw data and extract results from it. Most of the time, artificial intelligence (AI) is used with machine learning or even deep learning models which learn to classify between categories or predict numerical values. In a majority of cases, such models use supervised machine learning and learn from previously labeled data. In other cases, unsupervised learning models are given unlabeled data and try to find patterns by themselves. Another



(a) Interest over time of the term "IoT"



(b) Interest over time of the term "AI"

Figure 1.1: Interest over time of the terms "IoT" and "AI" on the Google search engine[1]. Interest values are indexed to the maximal value of each term, so a comparison between the two terms should not be made. For information, "AI" is a more searched term on Google in April 2023 than "IoT" by a factor of 50.

interesting approach is reinforcement learning, where an agent learns to make decisions by interacting with an environment and receiving rewards.

The IoT systems and the services embedded inside them are valuable. However, digital infrastructures come with many layers of costs which are often forgotten by the public. First, the collection of raw materials often requires precious metals which are only present in limited quantity on the Earth. After this first step, the production of digital objects also comes at a non-negligible cost: for an iPhone, hardware manufacturing represents more than 70% of its life cycle's carbon footprint (presented in [Gupta et al., 2021]). The usage comes after the production: the hardware itself requires power to function, and the large AI models, which are growing in popularity (Figure 1.1b), are often computationally expensive. An often underestimated step is the end-of-life process. Only a portion of IoT devices' materials can be recycled. A last example of non-negligible costs is transportation, which comes with digital objects throughout their entire life.

These steps are used in life cycle assessment, a methodology used to analyze the environmental impacts of a product. To improve the environmental impact of a product,

[1] https://trends.google.com
All footnote links in the manuscript were accessed on 2023-05-05

several directions can be taken. One could optimize the logistic, use less greenhouse gas emission energies, or decrease the total amount of used energy. This thesis falls under the last approach — reducing the use of energy in the usage of digital infrastructures.

1.2 Objectives and methods

The main objective of this thesis is to contribute to a more energy-efficient usage of IoT devices, with an application in the field of precision beekeeping. We will explore several topics that need to be taken into consideration.

The state of the art in precision beekeeping serves as a base to move toward a more energy-efficient smart beehive. Precision beekeeping is a multidisciplinary field that aims at helping beekeepers to make good use of colonies of bees and to protect them. This field regroups all the different systems capable of collecting the data of beehives, and also the services which process the data to produce value from it. It is key to understand how systems are designed and which services bring the most value to beekeepers. The research in precision beekeeping can help to lay the foundation for the subsequent results presented in this thesis, especially the existing literature focusing on energy efficiency. Those articles show the current solutions for the monitoring of the energy consumption of smart beekeeping systems, and also the needs of this subfield.

Then, the research conducted includes both theory and practice, by focusing on the development of an innovative solution to instantiate models. Developing our own autonomous energy-aware smart beekeeping system capable of collecting both bee-related and energy consumption data is the most adequate solution for conducting experiments in the field. Such applied research initiatives are a required task to achieve the main goal: our system goes through different prototypes which are deployed and tested to demonstrate their effectiveness. The development of a system, the collection of real data, their analysis, and the observation of potentially unexpected events constitute prerequisites for later optimization.

This optimization should then be multidimensional. We asked ourselves how much energy and money it costs to deploy one smart system, versus one hundred smart systems. We will also cover the optimization of where to allocate the computation, and the distribution of computationally heavy tasks: does the most energy-efficient architecture of hardware change if deep learning methods are added as a service? These criteria will be investigated through models simulated thanks to our own system's data and artificial intelligence methods. By doing so, we aim at giving answers to the questions of the scalability of our proposed system, and how to achieve the best beekeeping value per joule used.

1.3 Contributions

The contributions of this thesis, although following one guiding principle, can be divided into four main parts:

1. A literature review of the current state of the art in smart beekeeping. This review

covers the systems and services published in the literature on precision beekeeping over the last decade and more precisely the last five years. It not only lists the solutions from articles but also compares and analyzes them from performance and energy-efficiency standpoints. In this first contribution, we gain an overall understanding of the current field of precision beekeeping, we list the needs for this field, and we propose directions for future research. In the literature, the energy aspect is not a common criterion when designing and implementing a smart beehive, therefore the second step of the research is to measure the consumption of different hardware items and design a smart beehive that collects its own energy consumption.

2. We propose an autonomous energy-aware smart beekeeping system capable of collecting bee-related data and its own energy consumption. We describe its design and list the hardware and the financial costs. This second contribution contains the energy consumption analysis of an existing system. We decompose all tasks performed during a data acquisition iteration and measure their energy consumption. We highlight the limits of this existing precision beekeeping system and use this analysis to justify our choices for the newly proposed system. As a result, we have our energy-aware precision beekeeping system available for practical experimentation and more specifically, apiary and energy data collection. We collected data through the 2022 high season, and we make it available as an open-source dataset. The preliminary analysis of the multimodal bee-related data highlights some key states of the life of a colony. It shows the potential for our dataset to be integrated into artificial intelligence-based smart beekeeping services.

3. The third contribution is an energy consumption data analysis and a large-scale simulation of systems similar to the one previously introduced and their services. We first use the aforementioned deployed precision beekeeping system to measure real energy-consumption data. Then, a deep learning queen bee's presence classification model is added on top of the data acquisition and transfer routine. The energy consumption of the usage of this model is measured, and we identify the best "performance for energy" tipping point. Although optimized in terms of costs and performance, this extra layer raises the question of where to perform the deep learning task, which is computationally heavy and can overload the edge device. We evaluate the benefits of the addition of a cloud server on top of the edge system by simulating the whole system at a large scale for two computation scenarios (only using edge computing or using both the edge device and the cloud server). The computer simulation also takes into account the losses observed during the actual deployment of our system to produce realistic results. The simulation model generalizes and validates our whole solution, and serves as a tool for the placement of the services between the edge device and the cloud server.

4. For the last contribution, we focus on the possibility for IoT systems to select on their own, from a list of available tasks (or services), the combination of tasks that they run. The tasks all have some added value (from the end user's per-

spective) and some energy costs, and we want our task allocation strategies to maximize the total value of consecutive iterations of the IoT system's execution, while autonomously keeping sufficient energy budget. We explore the solution of modeling the environment of autonomous energy-harvesting stationary IoT systems and passing this environment into a reinforcement learning model. To do so, we introduce the dynamics of the environment with which the IoT system interacts: what actions the system can choose, what the system observes from the environment, the different values of rewards given by the environment to the system, and how the environment reacts to a given action. We also provide this environment with a machine learning model that predicts the photovoltaic energy intake of solar panels from outdoor parameters. Finally, we instantiate our environment with values from precision beekeeping systems, and compare the strategies and performances of four reinforcement learning methods and two manually-designed heuristics. We look for the method that generates the absolute best performance, the one that creates the most consistent strategies between different instances, and we analyze the use of the energy budget with respect to the weather conditions in different setups of simulation.

1.4 Structure

After this introduction, the literature review is divided into two chapters because of its bipolar nature. Chapter 2 introduces beekeeping and precision beekeeping, and covers the currently available solutions to design a precision beekeeping system. It also focuses on the efforts of researchers toward measuring the costs of their systems. Then, Chapter 3, explores the available services of a smart beehive. It describes the different ways of collecting bee-related data and processing this data with methods often using AI. Chapter 4 introduces our own smart beekeeping system. This system has the particularity of being able to collect both bee data and its own power consumption. Chapter 5 validates the previous chapter's contribution thanks to a large-scale simulation and a comparison between two main scenarios involving computation at the edge and in the cloud. Chapter 6 proposes the task placement reinforcement learning model which can be deployed in any smart beehive-like IoT system to optimize the value of its routines. Chapter 7 offers a reflection on the work presented. Finally, Chapter 8 concludes this thesis and lists future works.

Systems in Precision Beekeeping

Contents for chapter 2

This literature review describes the recent advances in precision beekeeping (PB) as systems and as services. Knowledge about bees, beekeeping and some context elements are given and defined in Section 2.1. Different types of sensors, networks, and power sources in PB are covered in Section 2.2. The collection and use of data are described, and the performances of PB services are assessed. The sustainability of the proposed solutions is estimated in Section 2.3, taking into account their scalability, efficiency, and economic cost, because beekeepers need deployable research results. For consistency across the thesis, the literature review is split into two separate chapters.

2.1 Beekeeping and its smart solutions

2.1.1 Bees and beekeeping

Honey bees are one of the main vectors of pollination: they transfer pollen between flowers to fertilize them. For some plant species, cross-pollination (pollination thanks to external organisms) is necessary for their reproduction. This type of pollination applies to both spontaneous and cultivated species. Agricultural production that depends on animal pollination quadrupled in the past 50 years [FAO, 2016].

The earliest human ancestors started to hunt bee nests for their honey. However, bees have been domesticated since 2400 BC [Crane, 1999]. Bees are extremely intelligent animals: they can learn tasks involving colors and shapes differentiation [Giurfa et al., 2001]. Bees also cooperate within a colony. For instance, foragers perform a waggle dance to their nestmates to indicate the direction, distance, and quality of potential food resources [von Frisch et al., 1967]. Beyond pollination, bees produce honey, royal jelly, propolis, pollen, and beeswax. Each of these substances has nutritional or health benefits. In temperate regions, bees hibernate during winter, start production in spring and continue throughout the summer, stop production and revert into the hibernation state in autumn. In hot and tropical regions, pollen and nectar sources can be available on a year-round basis, so bees do not hibernate.

Most of the cited articles in our survey focus on the *apis mellifera* species (western honey bee). This species is the most popular among beekeepers worldwide due to several selected advantages such as the ability to survive during dearth periods or "when honey flow is poor, the resistance to disease, the maximum amount of honey storage, the tendency to sting, and the ease of pacification by smoke" [Weber, 2013].

Honey bees are social insects. They live in colonies, with different kinds of members [Crane, 1999]:

- The queen: there is usually one per healthy colony, and it is the only female capable of reproduction. She is, therefore, the mother of all the bees of the colony. The queen is also the colony's barometer: if she is healthy, the colony is most often doing well. During their beehive control routines, beekeepers are looking for the queen to assess a beehive's state.

- Female workers: they are the most numerous individuals of the colony and perform tasks like cleaning the honeycombs, producing royal jelly to feed larvae and the queen, building the honeycombs, storing pollen and nectar, protecting the beehive, and foraging (listed chronologically in the life of a worker).

- Males (drones): apart from mating with the queen, male honey bees do not perform productive work for the colony. They alternate between eating, resting, and mating during the high season (spring to fall).

As opposed to wild nests, which are also studied by researchers like in [Gabitov et al., 2022], beehives are referred to as the human-made adaptation of bees' habitat. Modern beehives consist of several honeycomb frames, where bees store brood, pollen, and honey. There are different kinds of models: some hang frames vertically, others horizontally. Despite the variety of models, they all have in common an orifice for the entrance, a central part containing the frames, and surrounding walls to protect bees from external threats and weather. Nowadays, the most popular type of beehive in beekeeping is the Langstroth hive [Langstroth, 2004], consisting of a box where frames are stored vertically. Smaller boxes with extra frames called honey supers can be added on top of the hive to boost honey production when a colony is expanding.

Beekeeping is an activity that can go from a hobby to a full-time professional occupation. Even if professional beekeepers only represent small percentages among all

beekeepers, their production is substantial: in France, in 2021, the top 4.8% of beekeepers (professionals owning more than 150 hives) produced 64% of the national volume of honey (presented in [FranceAgriMer, 2022]). According to [ADA-France, 2022], this subset also owns 60% of the total number of beehives in France: 1 046 000 out of 1 738 000.

Beekeeping is also labeled as a multifaceted field. It has environmental (pollination and production by bees), socioeconomic (other sectors benefit from beekeeping), and sociocultural impacts (source of research and activities), as described in [Etxegarai-Legarreta and Sanchez-Famoso, 2022].

2.1.2 Threats to bees

Colony Collapse Disorder (CCD) is an unusual phenomenon that appeared in the early 2000s and which was first described in [vanEngelsdorp et al., 2009] as a sudden loss of honey bees. This phenomenon remains relevant today and several factors are associated with bee mortality: pesticides, Varroa destructor mite, genetic strains, habitat degradation, the Asian hornet, and viruses. However, it is still difficult to prove causality and to predict future consequences. Beekeepers should refine their diagnosis of honey bee colonies' health and invest more time in prevention to reduce the mortality rate of bee colonies that is currently in rapid growth.

There are little to no affordable health prevention tools on the market. There are however a few promising treatments against the varroa mite which, even if not perfectly effective, do not affect bees in the short and middle term. During winter, beekeepers can spray oxalic acid directly on frames. An alternative is using formic acid during fall.

The main goal of parasite monitoring research is to develop tools that automatically diagnose parasite threats and prevent those threats as early as possible.

2.1.3 Precision beekeeping

Precision beekeeping (PB) is a branch of precision agriculture, first mentioned in [Zacepins et al., 2012] and described as "an apiary management strategy based on the monitoring of individual bee colonies to minimize resource consumption and maximize the productivity of bees". PB tries to tackle challenges thanks to bee data gathered over time and throughout two main biological levels: apiary-level and colony-level.

- The apiary-level consists of several beehives consisting of one colony, each. The colonies share the same perimeter within a geographical location. It is also worth noting that the environmental conditions of the geographical location impact bees' behavior.

- The colony-level focuses on a single beehive. More precisely, on the colony's organization, bees' life, and behaviors.

This thesis uses the term *connected beehive* to designate a beehive that can collect bee-related data and possibly transfer data as part of a PB installation. A *smart beehive*

is a connected beehive with some intelligent services, for example, a beehive capable of diagnosing health issues.

Precision Beekeeping's main objective is to allow beekeepers to lean on automated tools to decrease their workload while strengthening the production coming from hives. Apart from targeting each seasonal crop alone, another goal of PB is to protect bees over time. Technology is needed to help beekeepers save time: using wireless installations, data about hives can be accessed directly on a smartphone. Therefore, the number of on-site visits, often using cars, can be reduced.

The authors of [Lettmann and Chauzat, 2018] addressed a form to French beekeepers in 2018 which aimed at understanding the dynamics of this profession. Professional beekeepers represent 13% of the respondents, but this ratio increases to 44% when omitting those who do not use connected devices in their beehives. In the same direction, 52% of beekeepers who use digital technology have been performing beekeeping for more than ten years. Therefore, it is clear professional beekeepers fuel the demand for smart solutions.

As of today, professional beekeepers who choose to use technology rely on two main categories of metrics: on the one hand, the weight of the beehive, which is a good indicator of honey flow evolution and flowering. It facilitates optimizing the selection of flora and the timing of transhumance[1]. On the other hand, internal and external temperature and humidity values can help track a colony's health and development [Cousin et al., 2019].

Location of research papers Chapters 2 and 3 cite a total of 108 PB-oriented publications, including 12 surveys. Figure 2.1 shows the distribution of the 96 result-oriented cited publications. Our priority is to describe recent state-of-the-art techniques in a global context. Although centered around Europe, where historical apidology research had been performed, PB research initiatives come from all over the world and have been gaining attention in developing countries, especially in the last few years. In particular, the authors of [Fiedler et al., 2020] are part of a 3-continent collaboration. They present a PB system deployment in Ethiopia and Indonesia, involving researchers from the latter two countries and Germany and Latvia.

It is however hard to deduce any geographical tendency as many research papers are often the only PB-related contribution of the first author.

Existing literature surveys In the past few years, a subset of PB surveys has been published in the literature. Each survey tackles PB with a different angle of analysis. The authors of [Zacepins and Karasha, 2013] describe the progress of the research in temperature monitoring of bee colonies, from simple thermometers and data collected manually to infrared imaging and the start of embedded solutions. The survey paper presented in [Meikle and Holst, 2015] reflects on the advancement of PB data analysis by listing comprehensively the types of data that a beehive can offer. The research described in [Zacepins et al., 2015] is close to this review because it covers

[1]Transhumance refers to the seasonal transport of beehives to different locations to take advantage of specific sources of nectar and pollen.

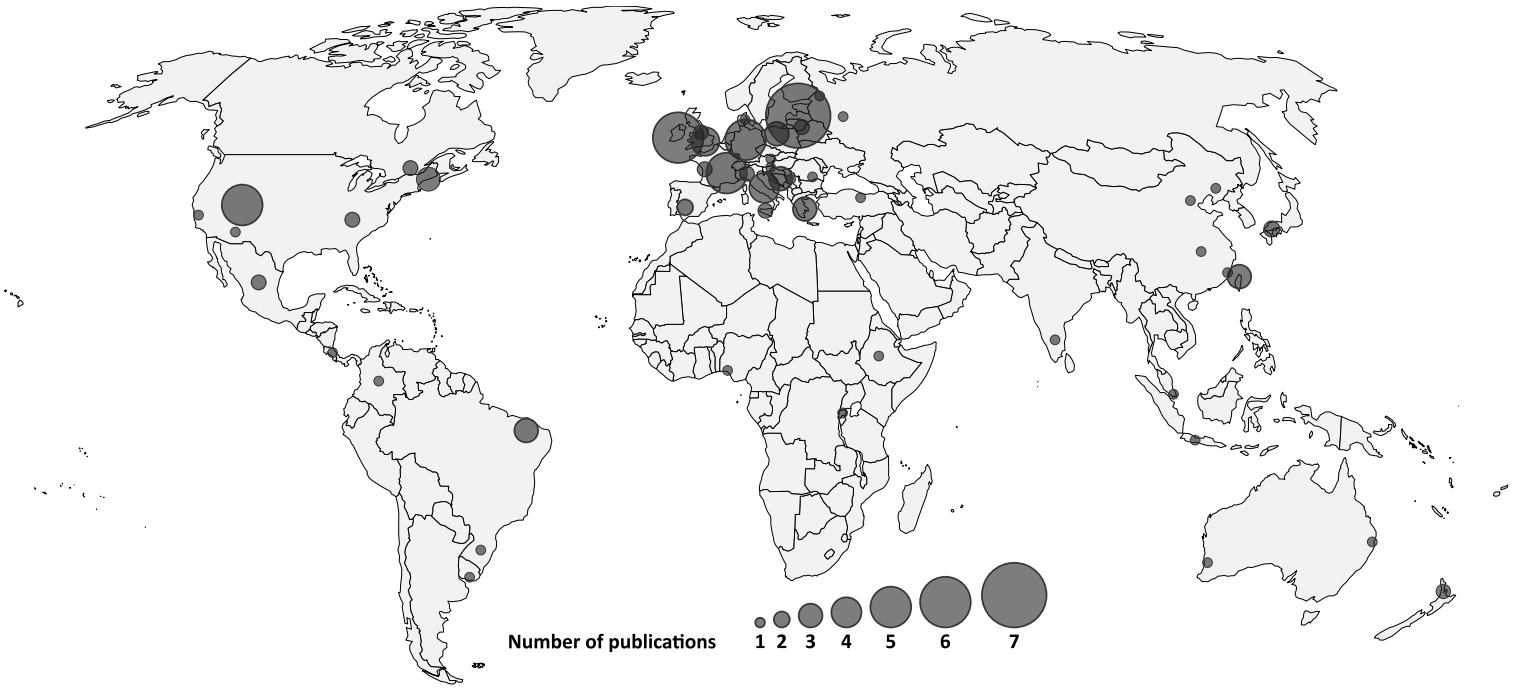


Figure 2.1: World distribution of PB research papers cited in this survey based on the location of the first author (surveys are omitted from this map)

the data collection and the data analysis steps of smart beehives up until 2014. The authors of [Marchal et al., 2020] examine the range of data collected in an apiary and the expansion to smart technologies directly on flowers, bees, and nests. The author of [Bumanis, 2020] states the different trajectories that PB will follow in the future. One trajectory is to combine different sources of data to make them more sophisticated. In [Terenzi et al., 2020], the authors describe the state of the art in PB sound analysis from early works to 2020. For each literature piece, they explain the methods and study their efficiency and drawbacks. Their chronological order of the listing creates an interesting historical development. The existing solutions of bee counting are reviewed in [Odemer, 2021]. Recent machine learning approaches are listed and commented in [Dimitrijevic and Zogovic, 2022]. The authors of [Horvath et al., 2022] focus on all the prerequisites before the processing of data. The authors' work aims at guiding future bee-related data research projects away from the difficulties which often arise. It explains the differences between hardware-related choices and the potential sudden events, among others. The review of the literature presented in [Uthoff et al., 2023] shows the state of the art of queen bee and swarming detection using audio and vibrational data.

Compared to other surveys, this state-of-the-art section integrates the whole data pipeline from the design of systems and the collection of data to the underlying intelligent models, which often embed artificial intelligence. At every step, we weigh the pros and cons of the solutions introduced in the literature and particularly care about the works which focus on the energy of PB's systems and services. The main objective of this literature review is to gain a global understanding of how smart beehives are

structured, collect data and process data. By criticizing different approaches, we also aim to raise questions for the future of PB, so that emerging research initiatives can gain a global view of the pivotal decisions to be made.

The following sections are organized as follows: Section 2.2 focuses on the connected beehive system (the hardware) and Section 2.3 highlights reproducible and sustainable research in terms of costs.

2.2 System overview of a connected beehive

This section summarizes the different existing scientific approaches and outlines the architecture (i.e., the system). The services will be detailed in Section 3.1 and 3.2.

2.2.1 Architecture of a connected beehive

In this thesis, we use the term Internet of Things (IoT) as defined by [ITU, 2012] as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies".

In all PB IoT-related papers, a connected beehive consists of a standard beehive with sensors connected to a microprocessor, powered by a battery and usually connected to a network to send the collected data to a remote server. The design of a connected hive must ideally fulfill the needs of beekeepers, be technically feasible and be suitable for bees' life.

Every smart beehive has a common structure:

- The brain of it is its microcontroller. At regular intervals, it gathers and centralizes the data before transferring it to a distant server. There are different families of microcontroller boards, two of the most popular ones being Arduino and Raspberry Pi. Each type of board has strengths and weaknesses. For instance, Arduino-based boards can consume less power than Raspberry Pi models. However, the latter can run an entire Linux-based OS, bringing more possibilities like programming with different languages. Larger embedded GPU-based computing devices are used for systems requiring real-time results with complex processing (e.g., Jetson TX2 in [Ngo et al., 2021] and Jetson Nano in [Wachowicz et al., 2022]). For example, the contribution described in [Rigakis et al., 2023] is a custom FPGA specifically developed and optimized for PB services.

- The organs of a smart beehive are its sensors. They are either directly plugged into the microcontroller or communicate the gathered data through wireless protocols. Section 2.2.2 will list the different types of sensors.

- Once the data is collected with sensors, it can be transmitted to a remote server. The information is either formatted and displayed to beekeepers on an application that groups the data by hive [Ammar et al., 2019], or analyzed by researchers [Giammarini et al., 2015].

- Finally, an energy source is essential to power the system. The classical autonomous setup consists of a solar panel charging a battery. In cases when beehives are located near a power outlet, microcontrollers can directly be plugged into a reliable source of current.

In most PB system research papers, researchers place sensors at various locations around and inside the beehive. As for the microcontroller, they position it outside the beehive. For instance, inside the roof of the beehive [Anuar et al., 2019] or in a dedicated box [Pérez et al., 2016] that is sometimes 3-D printed [Tashakkori et al., 2021]. Isolating the microcontroller has the advantage of being close to the bees while being protected from their defensive reactions. [Kulyukin and Reka, 2016] describe a PB architecture containing hardware placed in an empty super (a part of the beehive). There is no evidence of any beehive temperature imbalance in this case. An extension of the entrance of the hive as a container for the electronic system, serving as a take-off and landing platform for bees is proposed in [Chen et al., 2012], [Chen et al., 2015] and [Bjerge et al., 2019].

The design of a PB system needs to be well-thought and anticipate the barriers that come with deployment:

- The size of the beehive restrains the type of hardware. Training large AI models which require computation power on-site is an example of a challenging task, given this hypothesis.

- Gathering well-labeled data comes with a cost of time. Such autonomous systems must last a significant amount of time.

- Limiting the cost of the PB system is crucial to offer an affordable solution to beekeepers. For every PB system, there must be a trade-off between the cost of the installation for a single colony, the number of colonies equipped in an apiary and the quality of data that the system records.

2.2.2 Types of sensors

In the IoT field, the choice of appropriate sensors is essential to produce a solid base before data analysis. In PB, sensors can capture apiary-related or colony-related data. In the following paragraphs, we only focus on the latter since data that describe an apiary in its entirety can be collected without any connected beehive.

Weight The weight of a beehive indicates the activity level of a colony. During the foraging season, it reflects pollen and nectar collection as well as the number of bees inside their hive. During winter, there are fewer bees and fewer resources than during summer. Honey stock, which is the food consumed by bees, can be estimated using the weight of a beehive. Abnormal weight variations can also be correlated to incoming swarming events according to [Zacepins et al., 2015]. Swarming refers to the natural process in which a colony of honey bees divides into two groups. It occurs when the colony becomes too large and needs to expand. Swarming can have negative effects on

beekeeping, especially if the swarm leaves the hive at an inconvenient time. A swarm that leaves without a beekeeper's intervention can result in a loss of bees and honey production, as well as potential safety risks in the surrounding area.

For the majority of PB research, the measurement of the weight of a beehive is carried out in the following way: four load cells, which rely on mechanics and resistive theory, placed at each corner of the hive, integrated with amplifier modules that are linked to the microcontroller (in [Anuar et al., 2019], for example). HaBEEtat [Sakanovic and Kevric, 2020] presents an original weight-measuring system, with a connected frame holder inserted between the roof and the base of a beehive, at the very top of the frames. This built-in system enables sensors inside the beehive, without being intrusive, to measure the weight of every single frame.

The system described in [Fitzgerald et al., 2015] sets up a single-point impact cell placed under a supportive beehive platform. A single-point load cell is also used in [Zacepins et al., 2022] together with the HX711 load cell amplifier. Both measurements show a sensitivity of 10 grams which is ideal for beehive weight tracking.

Temperature A bee colony can regulate the temperature inside its beehive. The internal hive temperature is an indicator of the health of a colony. If the temperature is not regulated, the colony is in danger, especially during winter, when bees form a heating cluster to maintain living temperature conditions between 32°C and 37°C [Markovic et al., 2016]. During summer, bees usually ventilate to allow air exchanges. An increase in the internal temperature can be correlated with swarming (shown in [Zhu et al., 2019]).

To measure temperature, [Markovic et al., 2016] uses digital thermometers that have the advantage of communicating with a 1-wire bus, thus enabling the use of several thermometers with a unique microprocessor. The research presented in [Catania and Vallone, 2019] describes thermometers placed inside the beehive, at the center, where the queen bee is located. In contrast, thermometers are placed in the upper part of the beehive close to the roof in [Meikle et al., 2016]. Thermometers mentioned in [Kviesis et al., 2020] are placed above the hive body and outside the beehive, under a cover to avoid sun exposure, to collect both internal and external temperatures. In [Gil-Lebrero et al., 2017], temperature and humidity sensors are installed in the middle of the brood area, on the honey and pollen area, and on honeycombs from non-brood frames. The authors of [Giammarini et al., 2015] choose to place sensors (temperature, humidity, microphone, accelerometer and carbon dioxide) on the inner side of the beehive base, rather than directly on the frames. This has the advantage of being non-intrusive toward bees. However, the quality of data is lower than the on-frame alternative. In the very experimental setup of [Zhu et al., 2019], 36 probes per "space between frames" are placed in a 4-frame beehive to collect the evolution of a 3-D heat-map.

In [Cook et al., 2022], the placement of temperature sensors is analyzed. Data collected by sensors placed closer or further away from the center of the hive is collected. The daily temperature range is positively correlated with the distance of the temperature sensor from the center of the hive. This article validates the choice of existing systems in the PB literature to put temperature sensors close to the center of the hive.

Bees count The number of ingoing and outgoing bees is a crucial parameter correlated with the number of foraging bees, thus providing an indicator of the amount of pollen and nectar to be brought back inside the hive.

Optical, ultrasound and mechanical counting systems are tested in [Reyes et al., 2012] to measure the traffic and assess beehive activity with a modified beehive entrance. In the electronic entrance shown in [Chen et al., 2015], infrared sensors are pointing at the holes that bees use to go in and out, in order to detect each passage.

In [Zhang et al., 2021], the temperature, humidity, and audio data are collected from different beehives, which all have a varying number of frames (from 1 to 23 for bee frames and 0 to 11 for brood frames). Using these data, the trained models have the potential to assess the proportion of bee frames and brood frames inside a beehive.

Sound Sounds emitted by honey bees reflect the state of a colony as a unit. For instance, the audio footprint can be linked with swarm preparation, queen identification, and pest infestation (this topic will be developed in Section 3.2.1). Knowing that bees emit sound ranging from 0 to a few thousand Hertz, the sound sensor's frequency range needs to include that window.

To collect audio samples, the choice of the microphone is essential. There are several types: electret microphones are used in [Qandour et al., 2014] and [Anand et al., 2018], MEMS microphones appear in [Cecchi et al., 2018], and the setups described in [Ferrari et al., 2008] and [Kulyukin et al., 2018] involve clip-on microphones. All existing audio installations are placed inside the hive.

The collection of sound samples is described in [Cecchi et al., 2018]. The main unit containing the microcontroller, sensors, and microphone is placed in a protective box on the backside of a hive. It is unclear whether interior sensors are linked through drilled holes or passing through the entrance. Also, propolisation is mentioned, but it remains unknown whether authors met challenges with internal microphones. Propolisation refers to the process when bees recover a foreign body with propolis to isolate the beehive from this body. The system described in [Ferrari et al., 2008] shows that microphones (and other sensors) are protected from propolisation using a particular net. Sound recording nodes are placed on top of the frames under the roof. The recording node from [Robles-Guerrero et al., 2017] is also protected from propolisation, in this case, with a metallic mesh. In [Ramsey et al., 2017] and [Ramsey et al., 2020], the two accelerometers which record vibrational data are directly placed on a frame. A cavity is made at the center of the frame to place the sensor, and some molten wax is then used to prevent contact between the bees and metal components. Even after two seasons of use, it is reported that electronics are not affected by the colony.

In general, although most of the audio-based PB papers mention the challenges that internal microphones bring to bees' life cycle, the design and position of the recording nodes are not always sufficient to let future research reproduce their setup. Beehive audio monitoring is not converging toward a common framework, but it also guarantees unbiased and exploratory research advancements.

Image Pictures or videos of bees are an indicator of the activity and health of a colony. Even if the place of a camera is restricted to the outside of a beehive (the inside

is dense and dark), pointing at the entrance is the agreement that all computer vision PB researchers reach. Indeed, the most active outside location is the entrance, which regroups an important number of actions performed by bees (foraging, guarding and fanning).

In [Edwards-Murphy et al., 2015a] an infrared camera is used alongside infrared LEDs to monitor the inside of a beehive during times when the beehive is inaccessible (winter, night or bad weather). For now, IR images are only used by beekeepers, not computer vision models. In the same paper, a thermal camera is placed outside and estimates the hot area's size inside the hive. The authors of [Voudiotis et al., 2022] also place cameras inside the hive, more precisely inside the brood box, and use a small LED as a light source.

In [Yang and Collins, 2019], a camera is placed 30 cm above the entrance, and the landing platform is painted green so that bees can be distinguished better. Image samples are also collected at the entrance of beehives in [Sledevic, 2018], with a camera placed 40 cm above the beehives' entrances. Images are then cropped so that they each only contain one bee. Data from several beehives are collected, allowing different backgrounds and more consistency for the training part. The system described in [Ngo et al., 2021] includes an off-the-shelf web camera, and the entrance is enclosed in a black observation box with a lighting control system using a red LED. Compared with the two previous systems, this one allows real-time and long-term monitoring. Those three approaches aim at classifying the same thing: whether bees are carrying pollen sacs, and the last one allows bees counting and tracking too. Nevertheless, the first and third ones focus on the final accuracy, whereas the second anticipates future deployment on the field. Both works described in [Tashakkori et al., 2015] (bees counting) and [Magnier et al., 2018] (bees tracking) have a similar approach as in [Yang and Collins, 2019] using a white background. This setup also brings the constraint of a lower limit for the resolution: on the output images, bees need to appear clearly. For detection purposes, [Chiron et al., 2013a] opted for a limit of 6 pixels per bee. Other computer vision tasks require more pixels per bee (for example, varroa detection).

The authors of [Kulyukin and Mukherjee, 2019] (bees counting) work with cameras placed on top of supers. They focus on cases where one or two supers are added to the hive, making the distance between the camera and the landing platform either 35 cm or 60 cm, giving a field of view of either 50 $\times$ 80 cm or 100 $\times$ 160 cm. In each case, the region of interest retains sufficient pixels per bee to apply computer vision algorithms. The authors of [Kulyukin and Reka, 2016] (bees counting) put the camera above the entrance, in a compartment placed above supers. For research purposes, the experiment can be planned so that the setup is fixed, but supers are added and removed in a real beekeeping context. It changes the camera's distance from bees over time. In this paper, the authors choose to crop images to only retain a region of interest containing the landing pad. More specialized cameras are used in [Shimasaki et al., 2018] and [Shimasaki et al., 2020] (bees tracking), where 1024 $\times$ 1024 images at 2000 and 500 frames per second are captured, respectively. These cameras are placed in front of the beehive, capturing a 4 m $\times$ 4 m scene, which is the widest camera angle found in the PB vision literature. The captured scene aims at tracking wing-flapping movements which happen at a lower frequency than these frame rates.

2.2.3 Network architecture

The type of the chosen network in PB is determined by the goal of the research itself.

In applied research cases, data can be directly sent to beekeepers. For example, the system presented in [Seritan et al., 2018] sends data through SMS.

When the goal is to train AI models, the focus is not to deploy a network or send the data: it can be stored on hard drives or locally transferred through Bluetooth [Catania and Vallone, 2019].

When transmitting data through a network, the network itself shapes the design of the system. Using wireless personal area networks (WPAN) like Zigbee or IEEE 802.15.4 brings the constraints of needing a central station to gather the data and share it through another protocol to the cloud. The range of WPANs is also limited to a few dozen of meters. Low-power wide-area networks (LPWAN) allow sensors to be further from the base station. The range can increase to a few kilometers, which can be ideal when beekeepers choose to move their apiary to different locations in a medium area. Those two groups of networks both have the advantage of consuming small amounts of energy, enabling autonomous systems and low maintenance costs. On the other hand, beehives equipped with cellular networks (GPRS, EDGE) need more energy to transfer data but have the flexibility to share larger quantities of information. In the end, the choice of network is up to the objectives: working with stationary data points in time (temperature and gas data) is much lighter than working with heavy file formats like audio, image, or video samples.

The data can be transmitted to a server from each beehive or a central platform at an apiary. In [Edwards-Murphy et al., 2015c], a star network architecture is deployed. Each beehive has a Zigbee gateway capable of communicating with the base Zigbee 3G/GSM station, located at the apiary, which sends the data (temperature, humidity and gas levels) to the cloud storage server. The data is transmitted six times a day from beehives to the base station, and once a day from the base station to the server. In [Kviesis et al., 2015], data is also transmitted from each hive to the central unit, using the 433 MHz radio frequency band. The systems' structure in [Gil-Lebrero et al., 2017] and [Hong et al., 2020] are also star networks using IEEE 802.15.4 and Wi-Fi protocols, respectively. Each beehive can establish a connection with a local database server to send the data, and each server transfers the information to a global cloud database. The architecture in [Cejrowski et al., 2020] also corresponds to a star network schema, with the central device equipped with a GSM module connected to the Internet, whereas edges of the network communicate with the center through radio modules. In this case, particular attention is given to energy optimization. Each sensor is independent and can enter sleep mode to minimize power consumption. The authors of [Zacepins et al., 2018] chose a LoRaWAN network solution to connect three colonies and monitor their temperature. Since temperature data is single-point values, it is suitable to use LoRa, which is a long-range, low-energy, and low-speed (for data transmission) network. There are two main alternatives when using LoRaWAN: commercial solutions with telecommunication companies or amateur installations for the gateway. In [Zacepins et al., 2018], the first option is chosen, allowing connected beehives to include emitters and letting the Latvian operator take care of the gateway part. In an urban setup,

beehives can communicate data to a gateway placed 2.4 km away.

The design of a wireless sensor network (WSN) applied to precision beekeeping must consider the needs of beekeepers, the environment, and the specificities of the collected data. Each research topic has its combination of constraints so that there is no unique solution that fits all cases. In previously cited papers, most of the data is sent to the center of the network. One alternative to the star network is the peer-to-peer topology, where the network nodes are connected, enabling resource sharing for task parallelism. So far, it looks like none of the network-oriented PB papers opted for this kind of design.

In [Kridi et al., 2016], one of the focuses is only to share abnormal data that is worth informing the beekeeper. The objective is to monitor thermal dynamics inside beehives, so normal states are not sent, reducing packets and saving energy. The authors of [Tashakkori et al., 2021] mention the possibility of performing edge computing, i.e., performing computationally expensive tasks close to the data source, directly near the beehive. By doing so, only the results need to be sent, and LPWAN becomes viable.

2.2.4 Power supply

A self-sufficient approach for a power supply system is to use a solar panel to charge a battery. In [Seritan et al., 2018], [Pérez et al., 2016], and [Ammar et al., 2019]'s examples, solar panels are used. Their energy is converted using DC/DC step-down converters into a standardized 5 V that fuels batteries.

In [Edwards-Murphy et al., 2015c], the battery charge of sensor nodes is analyzed and optimized. There are different optimization levels: the rate of sampling (fewer data reads implies less energy consumed), the orientation of solar panels, and the order of sensor readings.

Other researchers focus on other tasks, simply design their experiments so that the hive is close to a power supply [Chen et al., 2015] and [Howard et al., 2018], or choose to power their system only with batteries that are swapped and replaced when empty [Catania and Vallone, 2020].

2.3 Costs of a connected beehive

In this section, we want to encourage using an approach that focuses not only on the quality of results but also on the inclusiveness of PB solutions. IoT and AI are fields which exponentially developed in the last ten years. Still, the objectives that drive an IoT or AI research article are often based solely on achieving the best performance. Ideally, including the economic, environmental, and social costs and reproducibility are vital. These factors would facilitate access to IoT systems and AI models for all, especially beginners with simple tools. The authors of [Schwartz et al., 2020] state that scientific quality criteria should include the efficiency and related AI solutions measures more commonly. Designing efficient models also enables building applications on low-computation platforms [Howard et al., 2017], which perfectly fits the design of PB systems and services.

Empirical analysis shows that PB research articles cited in our survey describing

either an IoT system or an AI model do not mention energy or price costs 70% of the time (67 articles out of 96 in that category). Among the same subset, there are 14 (15%) articles that mention only energy consumption and 10 (10%) which mention only the price of the hardware. 5 articles (5%) present a reflection on both of these challenges.

2.3.1 Price

The type of collected data directly sets a base for the price of sensors. The goal is then to optimize the return on investment in the quality of information that data reflect. On one side, single-point measures like temperature, humidity and weight, temperature and weight are the most obvious choices for a relatively cheap and basic PB installation that tracks the most crucial colony states. On the other, more complex information contained in sound and image data comes at a price but has the potential, if used correctly, to bring advanced analyses.

The authors of [Zacepins et al., 2016a] estimate their Raspberry Pi-based system to monitor 20 colonies at US$ 140. The complementary network part is US$ 475. Another Raspberry Pi-based system with audio and video monitoring costs US$ 106 in [Tashakkori et al., 2021]. In their case, the system is placed in an urban area where the Internet and power are accessible through wires.

In [Anuar et al., 2019], the weight, temperature and humidity data collecting system is estimated at around US$ 35 per beehive, given that there is Wi-Fi access. In general, we note that supposing that Wi-Fi availability considerably simplifies the costs, as many microcontrollers today include this functionality (ESP8266-based microcontrollers, Raspberry Pis). However, real deployment situations are almost always in areas where there is no Wi-Fi within range.

The addition of audio and video recording nodes adds around US$ 135 to the existing US$ 135 of the temperature, humidity, natural light level and air quality monitoring system in [Hunter et al., 2019].

Overall, sophisticated cameras aside, the investment in sensors and microcontrollers is usually more affordable than network solutions, although the architecture of a PB system often only requires one network component which centralizes the data transfer. In contrast, the number of each sensor must be equal to the number of monitored colonies. Concerning vision-based systems, simple cameras can be relatively cheap: an official Raspberry Pi camera is worth less than US$ 35. It provides 3280×2464 pixels images, allowing a potential long-term integration in a PB system. There are, however, research results that are difficult to integrate and distribute to beekeepers. For example, the camera used in [Shimasaki et al., 2020] which produces high-frame-rate videos, is not in the same price range as a Raspberry Pi camera. Still, such research allows the development of computer vision methods applied to apiculture and biological knowledge about bees.

2.3.2 Energy

As mentioned in 2.2.4, the power system of a PB system can be based on different strategies. In every case, if the final goal is to produce a beekeeper-friendly solution, the system must be autonomous. Whether it includes solar panels or only batteries, the power consumption must be optimized to avoid any kind of breakdown.

A simple power consumption benchmark is performed for two hours in [Anuar et al., 2019]. The system collecting temperature, humidity, and weight can live up to 12 days with the 2600 mAh batteries used. In this case, this confirms that one-dimensional data collected and pushed to a cloud server is not an energy-heavy task.

A power analysis is performed in [Fitzgerald et al., 2015]. The weighing scale's consumption is monitored: 200 mW is required to record weight, whereas 726 mW is necessary to send the data through radio (Zigbee). Even if it is unclear how much time each component needs to operate, we can speculate that weight recording is not an energy-heavy task. The authors of [Gil-Lebrero et al., 2017] also measure the energy consumption of the data recording node and the data transmission node. The results confirm the previous tendency: data transmission requires more than twice the current of the collection. In [Zacepins et al., 2020], the weight and temperature monitoring system is analyzed energetically. The main cost, as expected, is not the measurements themselves but the Wi-Fi transfer. The system can perform measurements every 2 minutes and last 18 days with four 1900 mAh batteries.

In [Edwards-Murphy et al., 2015a], the system relies on a 1000 mAh battery. It consists of an infrared camera, a thermal camera, a microphone and an accelerometer that consumes less energy than the 20 W solar panel production.

In [Kulyukin and Mukherjee, 2019], the computational cost is mentioned in terms of GPU hours. Deep learning methods require a much longer time to train, test and validate than the standard machine learning alternatives: 15 to 30 days for the selected convolutional neural networks, versus 15 hours for the random forests on the same task and dataset.

The battery charge level is tracked in [Edwards-Murphy et al., 2015c]. The sampling rate, sampling power, and solar panel orientation are all factors to be taken care of when tuning the parameters of the PB system. For some cases (maximum sampling rate and no solar panel adjustment), it looks like the battery charge tends toward zero. After some modifications, the charge level goes back to its initial maximum after the computational part. The system in [Edwards-Murphy et al., 2015d] has an audio recording software separated from the temperature and humidity node. This separation has the advantage of shutting down the audio part between sampling events to save energy. In this case, the software is also designed so that the energy consumption fits the budget of the solar panel, taking into account its efficiency. To do so, the authors performed an energy analysis of several data recording cycles. In the end, the daily energy expenditure is calculated (1272 joules) and compared to the solar panel budget (4220 joules).

The energy footprint of an existing PB system is broken down in [Hadjur et al., 2020]. Boot, data collection, formatting, and transmission (Wi-Fi) are benchmarked. This analysis takes into account the current consumption and the duration of each

task to retrieve the total costs, thus identifying the pressure points of the system. The consumption increases by 10% in cold (between 3°C and 5°C) conditions, compared to similar tasks in temperate conditions. A Raspberry Pi requires more energy (+0.5 W) to function if a USB peripheral is plugged in.

In general, energy-heavy operations are the collection of complex data such as video and image, the treatment of those data (computer vision) and the transmission of any type of data.

The power analysis articles cited in the subsection are useful prerequisites for potential future models which could select scenarios given energy budget and evolving constraints such as weather, battery charge and the needs of a smart beehive. For example, the choice between edge computing (performing the calculation with on-hive microcontrollers) and cloud computing (performing the calculation with cloud infrastructures, after the data is transmitted) could depend on several constraints, especially the energy budget.

By other means than the energy measures performed in the articles from this section, the authors of [Pignagnoli et al., 2023] propose a life cycle assessment on beekeeping. Even though it does not focus on precision beekeeping, this study states that the main contributors to greenhouse gas emissions for migratory beehives (when transhumance is done) are transport and supplement feeding. For non-migratory beehives, the contribution to climate change of the category "transport" falls to zero. This is a surprising result because beekeepers still commute regularly to their apiaries even if they do not indent to perform transhumance. However, the authors of [Pignagnoli et al., 2023] still advocate for the development of smart beehives, which minimize transport as a result of remote monitoring.

2.3.3 Open-source access

A trend for some research teams, publishing the code and the data from a paper is always a positive contribution: it allows other researchers to confirm the models on their building own data or test the data with other models. In some cases where the published work contains some weaknesses, it helps to correct it. In this part, we show a non-exhaustive list of diverse open-source initiatives in order to display examples of contributions that can serve as a base for new projects.

Over the years, the authors of [Kulyukin et al., 2018] kept publishing the code, dataset and detailed methodology for their system, alongside their results. They made available the code for the machine learning and deep learning experiments [2] and the rest of the BeePi project code [3].

The authors of [Ammar et al., 2019] published the plans of their connected roof and beehive base [4] as well as the code running inside their PB system [5].

The dataset from [Nolasco and Benetos, 2018a] is constituted with the combination

[2] https://github.com/sarba-jit/EBM_Audio_Classification

[3] https://github.com/VKEDCO/PYPL/tree/master/beepi

[4] https://github.com/emlyon/makers-beehives-fabrication and https://github.com/emlyon/makers-beehive

[5] https://github.com/emlyon/makers-beehives-hardware

of various sound data from individual beekeepers worldwide, who collected their data and added the state of each colony, forming a rich dataset [6].

The authors of [Nolasco and Benetos, 2018b] published the code for their sound classification models [7].

2.3.4 PB-related research projects

Throughout the literature review, we note that apart from a few exceptions, all the cited articles mention a link between their research and a broader project. Whether funded by one university or by a consortium of academics and governments, this brings money to fuel PB research. Each project addresses some specific issue.

For example, SAMS - Smart Apiculture Management Services[8] is a project from the European Union's Horizon 2020 research and innovation program focusing on PB in Europe and its links with countries such as Ethiopia and Indonesia. This shows that even if Figure 2.1 gives a global overview of the locations of PB articles, the underlying projects are sometimes international. On the opposite side of the spectrum, Hiveopolis[9] aims at adapting PB to nowadays' constraints, especially in urban conditions, using environmental data. In the same direction, Apicampus[10] is a university project aiming at studying the impact of stress factors on bees' health, including pollutants, pesticides, pests, or other environmental threats.

Bpractices[11] is another European project. It aims at diagnosing potential honeybee diseases all the while designing connected beehives which are respectful toward bees.

ITApic (Application of information technologies in Precision Apiculture)[12] is a European project which objective is more system-oriented than others. It focuses on developing a beehive monitoring system, on wireless sensor networks and their implementation in Europe.

Swarmonitor[13] is a European consortium whose objective is to develop monitoring tools based on beehive vibrations. The NU-Hive project[14] also focuses on a specific type of data: sound. Every contribution coming from this project explores ways to monitor beehive conditions using audio samples.

In some cases, PB start-ups also participate in research projects, such as Pollenity[15] – a company providing tools to monitor the temperature, humidity, weight, and sound

[6] https://zenodo.org/record/1321278

[7] https://github.com/inesnolas/Audio_based_identification_beehive_states

[8] https://sams-project.eu/wp-content/uploads/2020/06/GA_780755_D.5.3.-Evaluation-report-on-Bee-Mgmt-Health.pdf

[9] https://www.hiveopolis.eu/

[10] https://www.irit.fr/neocampus/fr/on-en-parle/post/95/projet-apicampus-ruches-connectees

[11] https://www.izslt.it/bpractices/en/the-project/

[12] https://www.era-learn.eu/network-information/networks/ict-agri/ict-and-automation-for-a-greener-agriculture/application-of-information-technologies-in-precision-apiculture

[13] https://cordis.europa.eu/project/id/315146

[14] http://a3lab.dii.univpm.it/news/mlsp2016-special-session/2-uncategorised/79-nu-hive-project-details

[15] https://pollenity.com/

of a colony – that is part of Hiveopolis. The authors of [Howard et al., 2018] use Arnia[16] commercial sensors. Moreover, they also add their system to complement and compare their solution to commercial sensors'.

The European Space Agency (ESA) also announced a project in PB: PASST - Precision Apiculture Supported by Space Technologies[17]. It gives access to the satellite data (Earth images) to perform crop analysis, thus selecting the best location for an apiary.

In general, research articles that come from North America are funded by university grants. Outside Europe, most research papers are the results of small-scale projects. European Union's consortiums have been boosting the research in PB within the past few years as shown in Figure 2.1.

[16] https://www.arnia.co.uk/
[17] https://business.esa.int/projects/passt

Services in Precision Beekeeping

Contents for chapter 3

This chapter is the latter part of the literature review. Section 3.1 and Section 3.2 go from how the data is collected to the artificial intelligence (AI) models used to assist beekeepers. Specifically, Section 3.1 describes the types of data collected and the way it is gathered and proposes a bibliometric analysis of the reviewed articles about precision beekeeping (PB) services. Section 3.2 goes through the different approaches used to develop PB learning models thanks to collected data. Finally, Section 3.3 concludes the literature review chapters and shares the lessons learned and directions for future work.

3.1 Data collection as a service in a smart beehive

In this section, we go through every type of data collected in the PB literature and describe the different methods of collection for each category.

3.1.1 Challenges

The main challenge of data collection to build statistical models in apiculture is bias. Most of the time, the duration of the collected data ranges from a few minutes to a

few days, whatever the type of the collected data (as in [Giammarini et al., 2015]). Moreover, each colony is unique, so analyzing just a few beehives for a small proportion of the life cycle of bees is not enough to understand all the subtleties of an apiary. Bias is even more present with models trained on data from one beehive (as in [Sledevic, 2018]). Research initiatives that aim at understanding the life of bees need to include at least data from an entire year, since bees have drastically different lifestyles between summer and winter, and their behavior at the beginning and the end of the high season differ too. The authors of [Davidson et al., 2020] tackle this challenge by combining different sources of data, coming from different apiaries, and different research initiatives, forming a robust dataset where a model can be trained on one part and tested on another one, independent of the first.

As mentioned in Section 2.3.2, energy needs to be taken into account at every step of the development of PB systems and services. When it comes to data collection, the energy cost is primarily correlated with the ratio between running and off (or sleep) states and the power drawn by each state. This topic will be covered using our own measures in Sections 4.1 and 5.2.

There are also different levels when it comes to collecting bee-related data. In [Zacepins et al., 2015], there are three categories: apiary-level, colony-level, and individual bee-level. For this work, we consider the latter as part of the colony-level category since data from counting systems or images of the entrance (considered "individual bee level" in [Zacepins et al., 2015]) are always analyzed colony-wise to assess the health of the entirety of a beehive.

3.1.2 Data collection at the apiary level

The wind is measured and compared to honey production in the work described in [Catania and Vallone, 2020]: wind variations are correlated with beehive development over a month. When long windy periods occurred (several days), bees refrain from going out to collect pollen and the honey production stagnates.

The threats to bees can vary when considering different countries. The flame sensor deployed in Rwanda ([Ntawuzumunsi and Kumaran, 2019]) constantly monitors smoke levels and emits an alarm if a threshold is exceeded, indicating near forest fire which can cause apiary destruction.

The data gathered in [Edwards-Murphy et al., 2015b] include meteorological and environmental metrics. Outside temperature, wind speed, atmospheric pressure, and rainfall are collected using a nearby weather station. Sunlight data is obtained thanks to an airport located 40 km from the apiary. Wind speed is sometimes collected with a nearby weather station, such as [Ngo et al., 2021]'s case where the collaboration of a national weather bureau is mentioned.

Choosing the right place to install an apiary is often decided based on beekeepers' experience. When realized correctly, it allows bees to focus their meals on unique flower types, enabling honey production with distinct flavors. This practice can sometimes be inaccurate, and the authors [Komasilova et al., 2020] propose a data-driven approach rather than experience-driven. Using manually annotated aerial and satellite images of agricultural fields, the optimal position of an apiary is computed. This work is

deepened in [Kotovs and Zacepins, 2023], where interactive digital maps containing the flowering calendar, weather information, plant data and potential connected beehives' data are proposed. It aims at providing beekeepers with the information they need to choose their apiary's location. The main challenges here are to gather such data which often come from different sources and rely on external entities (e.g., the Rural Support Service of Latvia in this paper) and to keep the data up-to-date.

3.1.3 Data collection at the colony level

Inside the beehive

Putting a sensor inside a beehive to gather data from a colony is an advantage over being outside. The concentration of life and the organization inside a hive are much more significant than monitoring individual bees flying outside.

There are, however, several challenges when it comes to measuring colony-level metrics inside a hive. First, it is essential to avoid affecting bees' life with sensors that could disrupt the structure of frames. Bees also produce propolis which they usually use to seal unwanted holes and isolate them from foreign objects like sensors. As mentioned in Section 2.2.2, cases of propolised inner-hive sensors are not always mentioned in the literature, and even regardless of the type of collected data. Finally, modifying the structure of a beehive to fit IoT systems would create empty spaces that bees fill with combs.

Temperature, Weight & Gas levels collection The analysis of the temperature of the cluster that bees form during winter to save heat is performed in [Markovic et al., 2016]. Thermometers are placed vertically on a frame. The collected data in one dimension is correlated to the 2-dimension distribution of the actual temperature on the frame and can help monitor the beehive during winter. In this case, although there are up to six sensors and their wires directly on a frame, the level of intrusiveness and the reaction of bees toward the sensor network is unclear. In [Hunter et al., 2019], internal and external temperatures and internal gas levels are recorded for several weeks during spring. The daily periodic component of the data is highlighted. Inside a beehive, the temperature reaches its maximum in the afternoon, decreases during the night and becomes cool in the morning. Gas level fluctuations have daily and weekly periods explained by the road traffic near the apiary.

The authors of [Edwards-Murphy et al., 2015b] and [Murphy et al., 2016] work with temperature, relative humidity, and gas (air contaminants, CO_2, O_2 & NO_2) data collected over 16 days on a single hive at rates of 3 times per day for gas levels and 6 times for the rest. Their goal is to develop models that can detect any abnormal colony state.

The authors of [Stalidzans and Berzonis, 2013] use temperature data collected on 14 colonies every 15 minutes during a year. Their goal is to model temperature dynamics with regard to the different long-term phases in a colony. Five periods are identified, and each is linked to a linear evolution of the temperature: winter brood rearing, spring brood rearing, summer brood rearing, autumn brood rearing and autumn broodless

period. They mention that direct contact between sensors and bees did not impact measurements. In [Anuar et al., 2019], the weight, temperature and humidity of the inside of a beehive are recorded every 7 seconds for 36 hours. The variation of a beehive's weight is correlated with the moment of the day when bee workers leave and come back inside their hive. It is concluded when there are fewer bees inside the beehive, it is suitable for a beekeeper to perform the checking routine. The work described in [Li et al., 2022] focuses on the in-hive temperature of 30 bee colonies throughout two overwintering seasons. Such long-term data collection allows identifying temperature thresholds for when bees start to go out (5°C) and when they start to significantly increase their activity (10°C). Also, the temperature difference is a good indicator of the activity of a colony.

To detect theft, an alert is sent to the beekeepers when an abnormal variation of weight is detected. Examples of such a service can be found in [Ntawuzumunsi and Kumaran, 2019] and [Anuar et al., 2019].

In a PB system final solution, such information can directly be sent to beekeepers who can apply their expertise to make the right diagnosis.

Sound collection In the following paragraphs, data is less straightforward than in the previously explained cases. It is up to researchers and beekeepers to ally themselves to define objectives and take advantage of artificial intelligence to produce intelligent services.

To publish an open-source dataset, the contributors of [Nolasco and Benetos, 2018a] and [Cecchi et al., 2018] regrouped sound recordings collected from individual beekeepers together with corresponding beehives' states. The resulting dataset is diverse and rich compared to more classical research initiatives. The types of microphones and their place inside beehives vary and bees are different from one apiary to another.

There is still a discussion within the scientific community about the invasiveness of microphones when collecting audio data directly inside the beehive. Although researchers in [Nolasco et al., 2019] and [Cecchi et al., 2019] state that sound analysis of acoustic signals is not invasive, the authors of [Aumann et al., 2017], who opted for an outside alternative, argue that regular microphones systems are still a disruption to the colony.

In [Ramsey et al., 2020], another inside vibrational type of data can be collected with accelerometers placed directly on frames. Their setup is one of the most complete of the apiary literature: 25 beehives are recorded in different apiaries over two years, resulting in roughly 2000 hours of vibrational recordings, making a prediction (as opposed to detection) of swarming events possible.

A posteriori observation of a key event in the life of the colony, such as swarming, is an important dimension to be considered before building learning models. However, beekeepers also gather a certain amount of information during their inspection visits (on average, once per week). For many papers, this data is simplified (prediction of only one trait, such as queen presence, swarming, or pest infestation). The protocol described in [Zhang et al., 2021] involves frame inspection (the percentage of bee or brood coverage is collected) and disease diagnosis in addition to potential rarer and more severe events. This is an original approach for building a model which allows

thinner data labeling.

Outside the beehive

Image collection There are different factors when comparing ways of collecting images of bees: the place and orientation of the camera around the beehive, and whether the beehive is modified to allow better conditions (lightning or forcing bees to a specific path).

During peak hours, dozens of bees can be spotted around the entrance of their hives, providing much information but also making tracking and counting challenging. On paper, one individual bee's trajectory could be misinterpreted, leading to misidentification, especially since bee motion is fast (8 m/s) and unpredictable. Another challenge arises from the outside constraint: the environment. Light and background vary throughout a day or a year. Any computer vision system needs to take into account this external factor.

An artificial setup is established in [Schurischuster et al., 2018] to get around the aforementioned constraints. The entrance of the hive is modified so that a tunnel made of several lanes becomes the new path for bees to commute. Bees are forced into one of those lanes so that they navigate through this tunnel, allowing the perfect angle for video shooting. The modified entrance is also artificially lighted up, and the sides are covered with dark adhesive foil to prevent reflection. This static setup also allows to pre-tune the focus and position of the camera. The authors of [Chen et al., 2012] proposed a modified version of the entrance too, where bees are confined when they go in and out. In their system, bees are tagged and individually tracked using image recognition. This system inspired the one shown in [Bjerge et al., 2019], where bees also go through a particular darkened narrow passageway. Here, a mirror placed under the entranceway reflects bees into the camera and allows capturing bees' images from their ventral side, where the varroa mite can be spotted. The mirror also allows illuminating bees with specific wavelengths since the mite is often well disguised. In this case, on top of the passive way of capturing images (waiting for bees to travel naturally), researchers opted for a more aggressive method which consists of shaking bees off from a frame to force them to go back in.

The authors of [Chiron et al., 2013a] compared 3D cameras (stereo camera and time of flight camera) pointing at the entrance from above, to build an acquisition system without any artificial setup which changes the beehive. Stereo cameras' depth maps are more accurate and have a better resolution than the time of flight alternative. After segmentation and tracking are applied, results show that 3D bee tracking outperforms 2D tracking for numerous bees on simulated data. This technique was promising in 2013, but recent research about it lacks since then. Today, such cameras are still too expensive to be integrated into a cheap deployable PB system. With a Raspberry Pi and its official camera, a 2D 3280×2464 pixels image can be captured, providing a good base for computer vision algorithms.

Vibrations One kind of non-invasive technique of PB that removes interactions between bees and wires, processing units, and sensors is the use of a Doppler radar.

Placed on the outside of the entrance, this type of radar is described by the authors of [Aumann and Emanetoglu, 2016], who built an original non-intrusive setup gathering acoustic data retrieving the typical sounds from a colony, like queen tooting and quacking. In another publication, [Aumann et al., 2017], sounds and vibrations of a hive can be measured in the frequency spectrum using the same type of radar microphone, and looking at the sign of frequency shifts, it is possible to spot individual bees flying out (negative) or back (positive) to their hive. Although only from the entrance of a beehive, the data collected by an outdoor radar still reflects the general health of a colony, as it is known that many events happen at the front of a hive, like foraging, guarding and fanning. The disruption of this activity is directly reflected by a weakness of a colony. Their last published work, [Cunha et al., 2020], retraces the analysis of data collected by their radar system over two years. The correlation between colony health and activity at the entrance is identified with data collected from a dozen colonies and three apiaries. It is worth noting that such radar placed outside the beehive is designed so that its autonomy and its small size are suitable for professional deployment.

Presence detector

Counting the bees going in and out of the beehive can be performed with a modified version of the entrance of the beehive ([Chen et al., 2015]). Thanks to an infrared transmitter-receiver module, the body of a bee as an obstacle to the infrared beam changes the value obtained at the receiver end. This modification is first registered as an analog signal, then transformed into a digital one using a field-programmable gate array to perform computation.

3.1.4 Bibilometric analysis: data types and their associated services

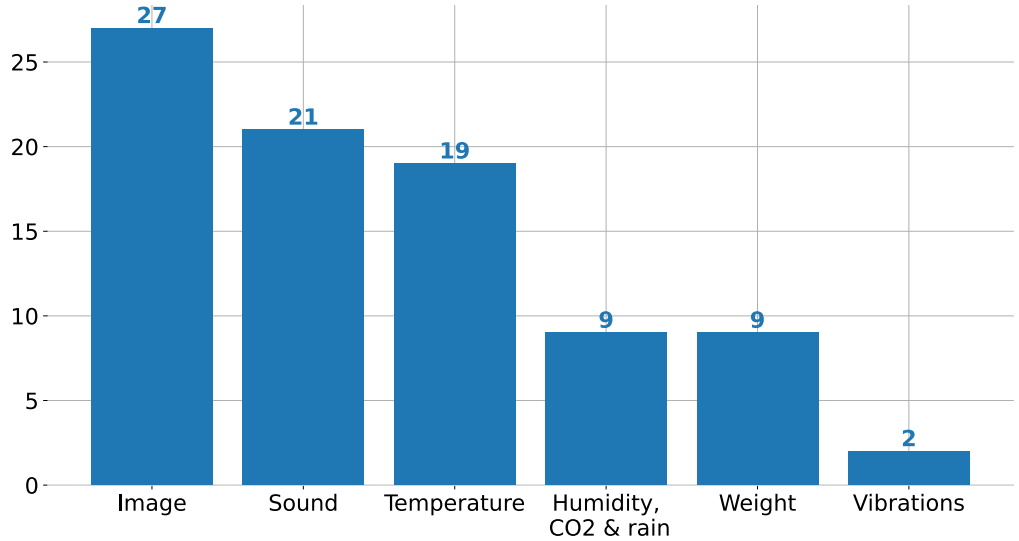
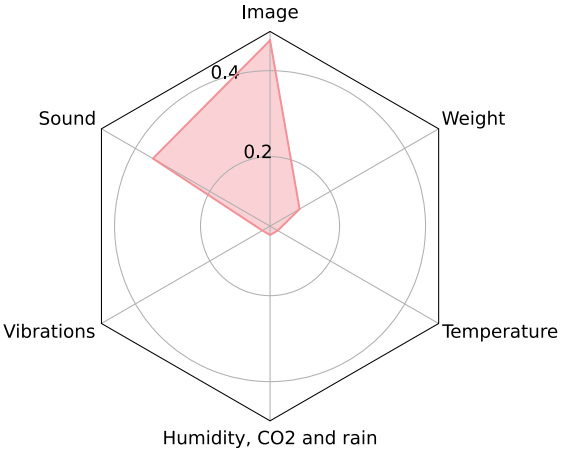
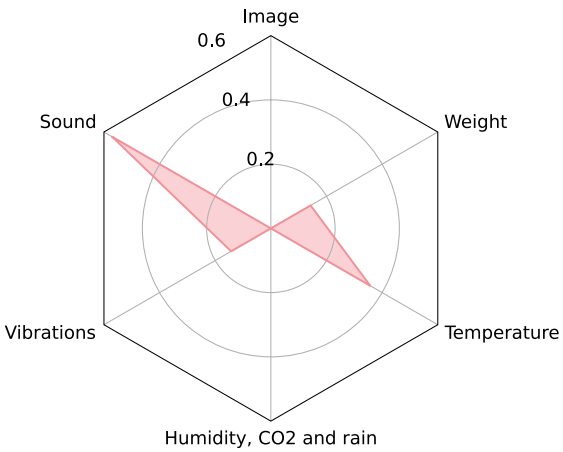


Figure 3.1: Number of cited papers with a learning model, grouped by type of data

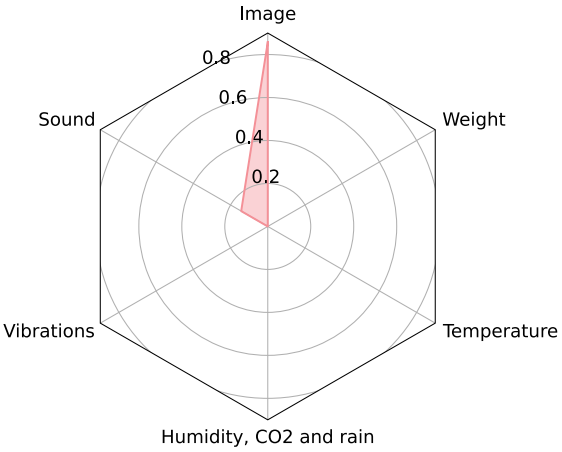
So far, Section 3.1 has introduced the different types of collected data. Figure 3.1 helps to gain an understanding of the proportions and number of articles related to each type of data (image, sound, temperature, humidity, CO2, rain, weight, and vibrations). In this figure, every article using bee-related data in a learning model is listed. Most articles focus on one type of data, but some can be counted several times if more than one of the types is used, like [Ferrari et al., 2008] with sound and temperature. The information gained from this figure is that even though existing installations for beekeepers rely on external sensors like scales, PB scientific research focuses on more complex data like images and sounds using sensors placed closer to bees. Systems used for research purposes are more intrusive than existing commercial solutions. Furthermore, they are more challenging to scale up in terms of the number of equipped beehives but enable a higher quality of data.



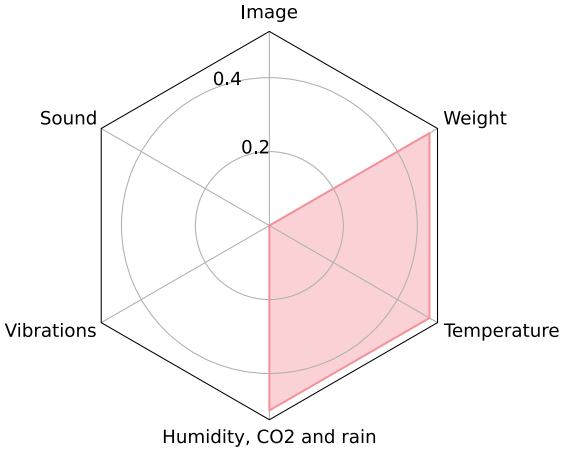
(a) Counting bees & population estimation (17 papers)



(b) Swarm detection & prediction (14 papers)



(c) Pest detection (7 papers)



(d) Honey production (4 papers)

Figure 3.2: Groups of services in PB and the types of data needed to perform them

Figure 3.2 groups papers with a learning model by services. It illustrates the types of data required to perform specific tasks. For example, Figure 3.2a shows that 47%

of articles counting bees or estimating the population of a colony use images for their models. Within the same subset, 35% of papers use sound, 12% use weight, and 6% use temperature, a mix of humidity, CO2 and rain, and vibrations. Swarm is detected or predicted mainly from the sound. In addition to the sound, vibrations, weight, and temperature can also be used (Figure 3.2b). Pest detection (Figure 3.2c) is mainly performed with images and occasionally sound. Finally, honey production (Figure 3.2d) is estimated using weight, temperature, and the mix of humidity, CO2 level, and rain level.

Overall, we focus on eight categories. Figure 3.2 shows four categories of services. The remaining ones are not shown in Figure 3.2 since they use fewer data types, thus less significant to represent graphically.

- **Counting bee & population estimation**: 17 papers (Figure 3.2a).
- **Swarm detection & prediction**: 14 papers (Figure 3.2b).
- **Pest detection**: 7 papers (Figure 3.2c).
- **Honey production**: 4 papers (Figure 3.2d).
- **Health estimation** which are defined differently in each paper: 12 papers — 10 using a mix of temperature, humidity, CO2 level and rain level data, and 2 using those variables and sound.
- **Pollen detection**: 5 papers, all using image data.
- **Tracking**: 9 papers, all using image data.
- **Queen detection**: 7 papers, all using audio data.

3.2 Data analytics as a service in a smart beehive

In this section, different models used to handle bee-related data are exposed. The selection process of the models, the tasks that they perform, and their accuracy are described.

3.2.1 Audio-based precision beekeeping services

Biological knowledge about bees' sound signals

The response from workers to induced acoustic vibrations has been analyzed since the middle of the 20th century (in [Hansson, 1945] & [Frings and Little, 1957]). With vibrations between 100 and 2000 Hertz, bees stop their activities.

Audio and vibration measurements of beehives have been performed since the second half of the 20th century in order to understand interactions within a hive. Sound has a crucial role in the life of bees. They convey such signals through the vibration of their thorax and the muscles of their wings, and the movement of their wings. Sound is transmitted through the air, and vibrations propagate through the wax comb or by

contact between two individuals, as shown in [Kirchner, 1993]. In [Wenner, 1964], one of the first spectral analyses of bee's sound is performed: a microphone and a spectrograph are placed inside a beehive and the sound emitted during the waggle dance and soon after the birth of a new queen are examined.

Using acoustic communication, bees share information with their colony about the distance, direction, and profitability of potential food resources. Bees communicate about food and potential dangers and their relationship together, especially for queens. Table 3.1 shows the associations between frequencies and their significance, which are vital to know for a beekeeper and scientists who wish to analyze beehive states. Because frequencies overlap, beehive sound data analysis is necessary to assess a colony's health and prevent dangerous situations. In all cases from the literature that include learning models, sound analysis involves classification models (rather than regression). The collection of a beehive's sound aims at capturing the general state of the colony. Inside a beehive, bees act as a super-organism. If an individual bee covers the sound of the whole colony, the global pattern cannot be picked up. This pattern changes based on events like swarming (shown in [Anand et al., 2018]) and even over the length of a day, as bees do not emit in the same frequencies in the morning compared to the afternoon (in [Pérez et al., 2016]).

Action	Frequency range	Sender	Significance
Flying	250 Hz	Colony	
Fanning	225-285 Hz	Worker	Regulate hive temperature & Evaporate water from nectar
Hissing	0 to several thousand Hertz*	Colony	Response to danger
Tooting**	300-500 Hz	Queen	Prevent hatching (other queen cups) & Inform workers about other queens
Quacking**	<500 Hz	Queen larvae	Answer to tooting from confined queens to alert about their presence
Waggle	200-350 Hz	Forager	Share location and quality of food

*Although most of the literature pieces state that bees do not communicate at frequencies over a few thousand Hertz, [Papachristoforou et al., 2008] recorded the reaction of Cyprian honey bees to induced hornet attacks, showing a dominant frequency of 6 kHz and harmonics reaching 16 kHz.
** Tooting and quacking are some subtypes of queen piping sounds.

Table 3.1: Types of vibration and sound emitted by bees (according to [Qandour et al., 2014], [Zgank, 2018] & [Terenzi et al., 2020])

Reference	Nb of hives recorded & Duration	Method	Learning Task
[Ferrari et al., 2008]	3 during 270 h	Filtering and spectrogram comparisons	Swarm detection
[Eskov and Toboev, 2011]	12 over 1 spring and summer	POLS with Gaussian kernel	
[Howard et al., 2013]	4 over 1 week	FFT, STFT, S-transform & SOM	Queenless state identification
[Qandour et al., 2014]	*	FFT, SFTF, PCA, LDA & SVM	Varroa mites detection
[Kulyukin and Reka, 2016]	4 over 2 days	FFT & [Tolstov, 2012]	None
[Pérez et al., 2016]	3 over 2 months	Spectrogram comparisons	None
[Ramsey et al., 2017]	1 in UK & 1 in France over a season	Spectrogram comparisons, PCA	None ("whooping" detection)
[Aumann et al., 2017]	1 over 4 days	Doppler radar data and its features	Beehive activity
[Robles-Guerrero et al., 2017]	2 over 45 days	MFCC, SVD & Logistic Regression	Queenless state identification
[Cecchi et al., 2018]	3 over a season	Harmonics comparisons	Characteristics of different moments of a day Characteristics of an orphaned colony Characteristics of the reaction to smoke
[Zgank, 2018]	Dataset: [Nolasco and Benetos, 2018a]	MFCC & HMM	Swarm detection
[Cejrowski et al., 2018]	1 over 7 months	LPC, t-SNE & SVM	Queenless state identification
[Anand et al., 2018]	*	FHT	Swarm detection
[Nolasco and Benetos, 2018b]	Dataset: [Nolasco and Benetos, 2018a]	MFCC, SVM & CNN	Presence of bees in recording
[Nolasco et al., 2019]	NU-Hive data: [Cecchi et al., 2018]†	MFCC, HTT, SVM & CNN	Queenless state identification
[Kulyukin et al., 2018]	6 over 3 to 6 months	CNN, Logistic Regression, k-NN, SVM, Random forests	Classify "bee", "cricket" and "noise" audio samples
[Hunter et al., 2019]	4 over 18 months	Spectrogram comparisons	None
[Cecchi et al., 2019]	NU-Hive data: [Cecchi et al., 2018]†	STFT, MFCC, HHT, CWT & DWT	Swarm detection
[Cunha et al., 2020]	Dozens over 2 years	Doppler radar data	Beehive activity and health
[Ramsey et al., 2020]	25 over 2 seasons	Average vibrational spectra, 3DFT, PCA, DFA	Swarm prediction
[Cejrowski et al., 2020]	2 over 1 month	RMS, ACF, MFCC & SVM	Classify bees day and night
[Zhang et al., 2021]	26 over 79 days	FFT, MLP & GPN	Count bees frames, brood frames and assess disease severity
[Soares et al., 2022]	NU-Hive data: [Cecchi et al., 2018]†	MFCC, time and frequency features, RF & SVM	Queenless state identification
[Dimitrios et al., 2022]	5 around 2 swarming events	MFCC, k-NN, SVM, U-Net CNN (newly proposed)	Swarm detection
[Cejrowski and Szymański, 2022]	4 over 4 months	MFCC & contrastive autoencoders	Anomalies detection
[Abdollahi et al., 2023]	11 (1 empty) over 3 months	Spectrogram, MFCC, DWT & CNN	Beehive activity (full frames of bees)
[Robles-Guerrero et al., 2023]	5 over 1.5 months	MFCC, LR, SVM, KNN, RL, NN	Classify colonies' states

*: unavailable; †: 3 hives over a season

Blue: Features extraction methods; Red: Learning methods

Table 3.2: Selection of PB papers involving sound and vibration analysis

First step of sound data analysis: filtering

Sound filtering is an optional first step when classifying audio samples. In [Qandour et al., 2014], a low-pass Butterworth filter is applied to restrain frequencies from 0 to 4 kHz, since the targeted sounds in this study usually reside below 1 kHz.

The work described in [Cejrowski et al., 2020] focuses on the whole colony's sound, so outlier sound samples where a single bee passes too close to the microphone are removed. Samples are only kept when the frequency of their most significant harmonic component is lower than 600 Hz, or when their Root Mean Square level does not deviate from more than 80% of the dataset average.

Second step: features extraction

Machine learning classification models take numbers, tables, or matrices as input. The first mandatory step in sound classification is to extract some numerical features from audio samples.

Mel Frequency Cepstral Coefficient (MFCC), Mel Spectra, and Hilbert-Huang transform (HHT), which are signal decomposition methods, are tested in [Nolasco et al., 2019]. The authors of [Zgank, 2018] and [Cejrowski et al., 2020] also work with MFCC, as well as [Robles-Guerrero et al., 2017], in which statistical descriptors (mean, standard deviation, kurtosis, skewness, median, etc.) are used to decrease dimensionality. The Fourier transform is used in [Qandour et al., 2014] to extract four features from the frequency domain: spectral centroid, peak frequency, root variance frequency, and bandwidth. After the training part is complete, the weights of the model show that the bandwidth is the least important parameter, whereas the spectral centroid and the peak frequency are deciding features for pest detection. The S-transform ([Stockwell et al., 1996]) is used in [Howard et al., 2013] to extract Fourier-like features, with the advantage of using a variable window length. Due to the high number of dimensions, the self-organizing map method is used to simplify the feature set. The Fast Hartley Transform is used in [Anand et al., 2018] to perform the same task as FFT. However, this variation requires less computation and uses less power, enabling real-time results on limited installations. Wavelet transforms (Discrete Wavelet Transform DWT & Continuous Wavelet Transform CWT) are used in [Rybin et al., 2017] and are studied alongside HHT in [Cecchi et al., 2019], where frequency properties during swarming are highlighted. To extract features from audio samples, the authors of [Cejrowski et al., 2018] choose Linear Predictive Coding (LPC). With 14 coefficients of dimension 3000, the sound is compressed but keeps enough information to describe the signal well. Those vectors are then reduced to a size of 16 using the t-Distributed Stochastic Neighbor Embedding (t-SNE) method. Such low-dimension objects aim at assessing the possibility of future queen presence classification: with a 2-D visualization of the two principal components, clusters can be observed (see Table 3.2).

Training models, accuracy & Conclusions

The quality of the training data is essential to build a solid classification or regression model. In [Abdollahi et al., 2023], the impact of external environmental factors such

as urban sounds, background speech and heavy rain, all impact the classification of the strength of beehives if they are kept in the training samples. The negative impact of such unwanted noise can increase the mean absolute prediction error up to 355%. This paves the way for more focus on cleaning noise out of the audio datasets in future research.

Singular Value Decomposition is applied to MFCC statistical descriptors in [Robles-Guerrero et al., 2017], and then a logistic regression is performed. With the data from two beehives (one healthy with the queen and one unhealthy without its queen) for one afternoon, which is a biased dataset due to its size, the presence of the queen is still detected perfectly.

The author of [Zgank, 2018] compares the swarm detection performances of different parameter tunings for Gaussian mixtures Hidden Markov Models (HMM). 81% of accuracy is obtained with 32 Gaussian mixtures, whereas 4 Gaussian mixtures perform at 68% of swarm/no-swarm classification accuracy. Although not precisely specified, the computational time and energy consumption of the latter alternative are the most compatible with an autonomous PB system.

In [Ramsey et al., 2020], swarming events are successfully predicted up to a month in advance using vibrational data: one of the two methods used involves two cross-correlation products between the averaged spectra and specific discriminant functions. Based on the results, an alarm threshold is defined. This first method only performs well during swarm seasons. The second method follows the same cross-correlation product idea. However, this time the entry is a 3D Fourier Transform obtained thanks to the information contained in the result of the combination of the Fast Fourier Transform (FFT) from several days before the current data point and discrimination spectra functions. In the end, temporality aside, the first strategy outperforms the second in terms of false negatives (no alarm when there should be). This is a considerable finding, as it is known that other swarm-related studies can only detect swarming or predict it right before it happens ([Meikle et al., 2006]).

Support Vector Machine (SVM) The learning methods used in [Qandour et al., 2014] to classify pest infestations are SVM and LDA. The results show that only a few audio descriptors are required to obtain a good performance (spectral centroid is giving a 70% accuracy alone). With the four descriptors selected, LDA can detect varroa presence at 81%. This study does not specify how many colonies are involved in the sound recording process, but we can speculate that a unique beehive is the only source of the dataset. SVM is also used in [Cejrowski et al., 2018] to detect queen presence and make the difference between two queens introduced to a swarm one after the other. For a specific colony (focusing on one queen), SVM is between 97% and 98% correct at classifying the presence of the queen. SVM is also efficient when the task is to make the classification between the swarm living with the first queen and the swarm living with the second queen. However, the classification model for queen presence, which is trained on the data from the first queen's state, is not accurate at detecting queen presence in the second queen's state. In the following paper, [Cejrowski et al., 2020], SVM is used, this time to differentiate day and nighttime for a colony. Several night lengths are considered, and the model's accuracy is shown for different times when bees

start their night. At best, SVM predicts the phase of the day at an 81% accuracy using the main dataset. It is concluded that bees' deep night is between 11 pm and 4 am. The experiments described in [Nolasco et al., 2019] compare CNN with SVM in a sound classification task to determine the presence of the queen. In all setups, SVM performs better than the deep learning alternative. However, the common weakness of those two techniques is the inability to generalize results to new colonies. This also confirms the trend described in the previously cited article.

SVM is a popular classical machine learning approach used in the PB literature for classification tasks. However, the authors of [Soares et al., 2022] compare this method with random forests (RF) for a queen presence model and find better their best results with RF with 31 MFCC features. It should also be noted that this paper focuses on developing a low-computation approach and shares some results with even fewer MFCC features (15). These results are worse than the 31-feature alternative, which reaches a 99.6% accuracy, but it still performs well with a 98.3% accuracy. The exact computation cost or energy cost for each case is not measured, this would be an interesting direction for future works in low-computation PB. In this paper, the authors choose to use more precision metrics to test their models than the rest of the literature with accuracy, kappa, AUC, specificity and sensitivity. Such density of testing should be standardized to compare different approaches.

In [Robles-Guerrero et al., 2023], the performances of predicting the presence of the queen bee of five machine learning algorithms are compared. More precisely, the three chosen states are healthy "queenright" colonies, "queenless" colonies where the queens were removed, and a "queenless" colony with a low population. SVM, KNN and NN (lightweight) have higher performance scores than LR and RF. However, a computational time comparison is also performed and shows that SVM and NN's inference time are shorter than KNN's (3 ms, 2 ms and 48 ms, respectively). SVM and NN are the most suitable solutions for deployment on low computational resources.

Deep learning The authors of [Kulyukin et al., 2018] decide to split training and testing samples by beehive and location in order to validate the ability of models to extrapolate their knowledge. Testing models with data from the same apiary or even colony would introduce bias, and results could have been good due to the footprint of a colony rather than the general properties of bees, which are what we want the models to learn. The latter brings more knowledge as models which can generalize are always important contributions. In this research, a second dataset is tested and separated by beehives, location, time, and bee race to try an even more challenging setup. The comparison between standard machine learning and deep learning models in [Kulyukin et al., 2018] shows that distinguishing between "bee", "cricket" and "noise" labeled data can be performed evenly if the data is separated by beehive and location using a custom Convolution Neural Network (CNN), logistic regression, SVM, k-NN or random forests. In their second dataset, where the training and testing data is split by beehives, location, time, and bee race, deep learning outperforms standard machine learning approaches. CNNs also perform better when trained on raw audio samples (rather than spectrogram images). A key question answered in [Kulyukin et al., 2018] is: is it worth embedding a CNN in a PB system? There is no doubt about the

accuracy of deep learning, especially with no necessary feature extraction needed and in a more challenging setup where generalization is required. However, the actual cost of such a model resides in the training part, which is longer and more energy-heavy. Even if the experiment cited involves sound classification between "bee", "cricket", and "noise" samples, which is not useful for the beekeeper, the conclusions about the model's efficiency may be transferred to more practical issues.

In general, advanced beehive audio analyses involving machine learning, which are growing at a fast rate, still rely on the knowledge from experimental studies from the past century. One challenge remains: the length and diversity of data always pose the question of the biological sense of such studies. Of course, models are validated because accurate and efficient in specific situations, but general results about the life of honey bee colonies are too far from the present isolated results.

3.2.2 Image-based precision beekeeping services

Image pre-processing

Pre-processing for image classification in PB involves removing backgrounds and isolating individual bees to count them, track them, and find pollen or parasites on their bodies. In general, the camera setup gives simple enough images (bees in front of a plain background) to prefer color-based segmentation methods rather than deep learning alternatives. Under those conditions, the main difficulty becomes the shadows of the bees in the background, which can lead models to count each bee twice or to count a bee even if it is not inside the camera's angle. To solve this issue, the dominant blue color of natural light is taken into account. It is the reason why shadows have high saturation in blue channels. Since bees are more yellow and brown than blue, filtering their shadows remains an easy task ([Magnier et al., 2018]). In [Stojnic et al., 2018], the approach starts with considering images in the CIE LAB space and considering the two non-illumination channels, so that differences in light levels are not seen. This segmentation method is compared with k-means. In [Schurischuster et al., 2018], from an original picture of bees to the training features, a foreground detection technique (median image and Gaussian mixture models are compared) and color thresholding (simple region threshold in CIELab and HSV color spaces are compared) are applied. Then, once the background is removed, parts of the image where the bees are located are extracted (patch extraction, using connected component analysis). At the end of the processing part, images that are delimited by the shape of individual bees are generated. In [Sledevic, 2018], the same training is performed with RGB and HSV color spaces to compare their effectiveness.

To create a background to be subtracted from pictures where bees appear, synthetic images without bees can be created with the average of many other images where bees can appear ([Tashakkori et al., 2015]). In [Yang and Collins, 2019], mixture of Gaussians models (MOG) are used to subtract the background. HSV color space is also explored here, with the potential of removing shadows.

Reference	Resolution (FPS if video)	Method	Task
[Ramìrez et al., 2012]	708 × 480 (30 FPS)	MAA detection	Varroa mites detection (without bees)
[Chiron et al., 2013a]	752 × 480 (50 FPS), stereo camera	GNN, Kalman filters	Bee tracking
[Chiron et al., 2013b]	752 × 480 (47 FPS), stereo camera	HIDS, GNN, MHT, Kalman filters	Bee tracking
[Tashakkori et al., 2015]	*	CC & OF	Bee tracking
[Sparavigna, 2016]	*	Image Segmentation	Honeycomb analysis
[Chazette et al., 2016]	320 × 240 per bee	RL/CI & CNN	Varroa mites detection
[Kulyukin and Reka, 2016]	720 × 480	OpenCV algorithms	Bee counting (omnidirectional)
[Schurischuster et al., 2018]	1920 × 1080 (30 FPS)	GMMs, Color Histograms, Dense SURF, Naive Bayes, SVM & RF	Varroa mites detection
[Magnier et al., 2018]	360p, 540p, 720p & 1080p (60 & 30 FPS)	Edge detection, fit ellipses, concordance	Bee tracking & counting
[Stojnic et al., 2018]	1280 × 720	SIFT, VLAD & SVM	Pollen sacs detection
[Sledevic, 2018]	1920 × 1024 (3 FPS)	CNN	Pollen sacs detection
[Shimasaki et al., 2018]	1024 × 1024 (2000 FPS)	STFT	Bee tracking & wing-flapping freq. measures
[Yang and Collins, 2018]	1920 × 1080 (50 FPS)	Hough transform, Kalman filter, Hungarian algorithm	Bee tracking (directional)
[Yang and Collins, 2019]	100 × 100 to 200 × 200 per bee	MOG & Faster RCNN	Pollen sacs detection
[Kulyukin and Mukherjee, 2019]	360 × 240 & 1920 × 1080	MOG, k-NN, SVM, RF & CNN	Bee counting (omnidirectional)
[Bjerge et al., 2019]	1296 × 966 (31 FPS)	SIFT, SURF, ORB, LDA & CNN	Varroa mites detection
[Shimasaki et al., 2020]	1024 × 1024 (500 FPS)	STFT	Bee tracking & wing-flapping freq. measures
[Mukherjee and Kulyukin, 2020]	160 × 120, 320 × 240 & 640 × 480 (25 FPS)	DPIV	Bee tracking (directional)
[Ngo et al., 2021]	640 × 480 (25 FPS)	Kalman filter, Hungarian algorithm & CNN	Bee detection, tracking, recognition of pollen and non-pollen bearing bees and counting
[Voudiotis et al., 2022]	800 × 600 (camera) into 640 × 640 (CNN)	CNN: MobileNet V2 & V3 and ResNet50	Varroa mites detection
[Sledevič et al., 2022]	512 × 128 (113 FPS) & 256 × 64 (1250 FPS)	CNN optimized for FPGA	Pollen sacs detection
[Wachowicz et al., 2022]	960 × 540 & 480 × 270	k-NN, MOG2 & CNN	Bee identification & Varroa detection
[Kulyukin et al., 2023]	1920 × 1080 (24 FPS)	RF, SVM	Bee tracking

* : unavailable

Blue: Image transformation without Machine Learning, Red: Classical Machine Learning, Green: Deep Learning

Table 3.3: Selection of computer vision PB papers

Models & accuracy

Classical machine learning Naive Bayes, SVM and RF are tested in [Schurischuster et al., 2018] to classify a balanced 200-image dataset with "mite" and "no mite" labels. As input, image features are obtained with color histogram, color histograms moments, local binary patterns, and dense SURF. This study compares the accuracy and F1-score of different combinations of color models, image features, and classification methods and obtains a maximum accuracy of over 80%. In [Stojnic et al., 2018], pollen-bearing bees and non-pollen-bearing bees are classified using SVM mixed with PCA using SIFT and VALD to compute image descriptors. A maximum of 0.915 AUC score is obtained (see Table 3.3).

Deep learning In [Sledevic, 2018], a CNN is applied to individual bee images to detect pollen-bearing bees. The size of the filter in the convolutional layers has to be larger than the pollen-bearing targets (5 pixels) and smaller than 15 pixels because of computational cost in embedded systems. The number of layers is also crucial as models with 3 layers outperform 1-layer and 2-layer models overall, giving a maximum of 94% classification rate with RGB images and 92% with HSV images from different hives. In [Yang and Collins, 2019], another CNN is tested with the same pollen detection task, using 13 convolutional layers and 3 fully connected layers. This model (7% error rate) is outperforming the statistical two lines model (33% error rate) on the same dataset. One common challenge of the two above-mentioned studies is the distribution of pollen-bearing bee samples (positive). There are 1000 positives and 1000 negative samples in the first analysis, and 10% of them are selected for validation. In contrast, the validation set for the second analysis contains roughly 20 times fewer pollen-bearing bees than non-pollen-bearing bees. In this case, we question the information given by the error rate, because the model is effective at not finding any pollen sacs but struggles when there is some.

The authors of [Kulyukin and Mukherjee, 2019] opt for a method where a motion detection model suggests areas of interest where potential bees are. Then, CNNs, SVM, and RF are applied to classify images with and without bees. Results from hand-designed and auto-designed (designed with automation) CNNs are compared, and hand-designed alternatives marginally outperform auto-designed CNNs. RF perform better than other standard machine learning approaches like SVM. They also represent a middle ground between the maximum accuracy of DL techniques added to the computational resources, and the standard energy-efficient machine learning alternatives, which often fail to produce comparable results. In [Voudiotis et al., 2022], two scenarios "online" (computation in the cloud) and "offline" (computation at the edge) are shown.

The main drawback of deep learning applied to PB is the reactivity of models. Even after the training is completed, results often cannot be obtained instantly, especially with CPU-based mini-computers. Although results given a few minutes after the images are collected can still be viable, Ngo et al. present in [Ngo et al., 2021] a real-time pollen sacks detection model using GPU-based hardware and the YOLOv3-tiny architecture for its deep neural network, known for its fast detection speed. In this paper, the F1-score is 0.94 for pollen and non-pollen bees recognition, and the

mean absolute percentage error (MAPE) is 8.45±2.72% for the pollen-bearing honey count and 10.55±2.10% for the total incoming bees. After pollen sacks recognition, a Kalman filter and Hungarian algorithm are used to perform tracking tasks. The authors of [Sledevič et al., 2022] propose a CNN accelerator architecture optimized for field-programmable gate arrays (FPGA). They measure the computation performance of their implementation and achieve 8.8 ms and a 2.4 ms pollen detection for 512×128 px and 256×64 px images respectively, together with a 92% accuracy for the model. This study is so far the first PB on-edge real-time implementation of a computer vision task able to perform at least 90% accuracy between 113 and 1250 FPS, depending on the resolution. Previous ones either predict on a PC-like system ([Ngo et al., 2021]) or just use classical machine learning models like SVM on Raspberry Pi ([Babic et al., 2016]), which limits the frame rate to 1 FPS.

Other statistical methods The authors of [Shimasaki et al., 2018] used Short-time Fourier Transform (STFT) to analyze video data at high frames per second. By doing so, the flight path of bees can be seen as the undulation of the brightness of pixels and tracked with precision. A 180 Hz frequency component in the spectrum corresponds to the wing-flapping frequency. In [Shimasaki et al., 2020], the evolution of the flapping frequency is added. It ranges from 150 to 250 Hz.

Using stereo cameras (2D image & depth maps), the bee tracking techniques described in [Chiron et al., 2013b] rely on a novel hybrid intensity depth segmentation, Kalman filter, and Global Nearest Neighbors (GNN). In this case, each target is linked to a filter used to estimate the trajectory, and GNN used available information to help with uncertainties. With these tracking models, false positives mainly arise when the flight boards are crowded.

In [Magnier et al., 2018], the movement of bees at regular video frame rates (30 and 60 FPS) was analyzed. In their setup, the challenge is to map an individual image per image when two or more bees appear in the video. To do so, the two previously known images are used to predict the next one. For each frame, each bee is represented as an ellipse representing a position and an orientation. Once trajectories are found, entering, departing and passing bees can be counted based on the starting point and finishing point of paths. In this study, four background removal methods are compared based on the counting accuracy.

The use of color distribution in an image (histogram analysis) can be used to detect anomalies such as varroa mites. However, such methods need to add the position of the mites to eliminate them. [Chazette et al., 2016] chooses Region Labeling to detect the position of mites and describes a real-world scenario with CNNs and lasers that locate and eliminate varroa mites.

In [Mukherjee and Kulyukin, 2020], a method that computes directional vectors rather than entire flight paths is performed (DPIV: Digital Particle Image Velocimetry) to measure traffic with entrance images. On top of directions estimation, it performs as well as state-of-the-art techniques to measure omnidirectional traffic ([Kulyukin and Mukherjee, 2019]). In comparison to CNNs, DPIV also does not need a long and dedicated training part.

3.2.3 Other types of data for precision beekeeping services

In [Davidson et al., 2020], swarms are detected thanks to temperature passed into a rule-based algorithm and autoencoders, which consist of a pair of neural networks (encode and decoder) that are, in this case, two long-short-term memory networks (LSTMs). The results show that all swarms are detected with the autoencoders method, and it is even triggered by other anomalies (mainly from beekeeper's intervention: varroa treatment, control visit). Loss of temperature control, which can be correlated with swarming, is predicted using homeostasis, hive mass, and acoustical and external meteorological conditions in [Braga et al., 2021]. To do so, they also use LSTM-based neural networks.

The authors of [Ngo et al., 2021] show the correlation between environmental data and the activity of a colony (in this case, pollen foraging). The results emphasize that temperature, relative humidity, wind speed, rain level, and light intensity influence pollen foraging. For example, heavy rain or a gentle breeze negatively affects the harvest. [Robustillo et al., 2022] also shows a prediction of internal hive variables (temperature, humidity and weight) using external variables (temperature, humidity, rainfall, wind speed, air pressure, particulate matter and brightness). The tested methods are vector autoregressive models, dynamic linear models and generalized additive models. Vector autoregressive models and their variants — time-varying vector autoregressive models — show the best results for internal variables' prediction. In [Catania and Vallone, 2019], the collected temperature is compared with the data from [Meikle et al., 2016], where thermometers are in the upper part of the beehive, close to the roof, in a less dense area of the hive. The first case finds a weaker correlation between inside and outside temperatures, compared to the second. The authors of [Ziegler et al., 2022] also study the interrelationship between bee-related variables (weight of hive) and external variables (temperature, dew point, humidity, wind speed, precipitations) using vector autoregressive models. They find that the maximum temperature variable and the minimum dew point are the two variables that explain the best beehive's weight with respectively 7.5% and 3.2% of the beehive's weight variable's variance.

Bee traffic is predicted with electromagnetic radiation and weather using SVM and RF in [Kulyukin et al., 2023]. As reported in other studies, RF perform better than SVM and this paper also states that it is more energy-efficient. The main finding is that electromagnetic radiation is as good as weather for traffic prediction, and both electromagnetic radiation and weather are better predictors than time. The authors of [Andrijević et al., 2022] compare three models for bee traffic estimation with 24 internal and external parameters. LSTM recurrent neural network outperforms ARIMA and Facebook Prophet models.

The authors of [Kviesis et al., 2020] have a fuzzy logic approach for state detection. In their fuzzy inference system, they take into account the inside temperature, the ambient temperature, the difference between to two, the evolution of the inside temperature, and the season to detect normal, abnormal and death states. The model obtains an F1-score of 98%. In [Edwards-Murphy et al., 2015b], a threshold-based algorithm is developed to assess the general health of a colony. For example, the authors state that the variation of temperature over 24 hours should not be greater than 20

degrees. All thresholds and intervals have been validated with accurate alarm triggering. Advancement of the threshold-based algorithm is proposed in [Murphy et al., 2016], where a decision tree is used to evaluate a beehive's state using heterogeneous data. One of the advantages of such a lightweight program is that it fits on a microcontroller with low capacity and low energy consumption (the ATmega1281 for example). To design the algorithm, a biological analysis is first performed to bring 10 important beehive states out. Some describe normal activities while others, when detected, must trigger an alarm sent to the beekeeper. Then, the training of an iterative dichotomiser 3 decision tree algorithm is performed with labeled data as input. In the end, over 95% of states are accurately classified. In comparison, the previously cited threshold-based method reached at most 89% of accuracy with the same dataset.

The work described in [Cejrowski and Szymański, 2022] shows an approach that mixes audio, temperature and humidity data with the aim of detecting anomalies using contrastive autoencoders models. In this study, the anomaly is a feeding of glucose syrup, so it is induced by human action. This is a first step toward more general models which are not specific to one task, but such methods should be implemented in the future with more various sources of anomalies, in order to obtain a more general anomaly detector.

3.2.4 Discussion on the precision beekeeping services

Sections 3.1 and 3.2 show that every type of data has the potential to bring intelligence to a connected beehive. The differentiating factor between models that confirms results from the past and a significant contribution is the data quality. Whether a study uses a single data source or heterogeneous data, the collection over several full seasons over different apiaries and precise labeling are necessary to produce unbiased and general biological results.

To collect data, only a few articles document their challenges when installing in place their sensors. For better traceability, PB research papers need to specify why and how sensors should be placed, but maybe more importantly, explain the constraints that led to their choices and the reasons why other options are not selected, if possible.

Is it worth embedding an AI model inside a PB system? The vital aspect to be taken care of is energy consumption, finding the tradeoff between intelligence and energy consumption becomes the issue that beekeepers and researchers must agree on. From a research perspective, the final product and the preliminary data collection tools must be distinguished. The first one must fit in a PB system as described in Section 2.2: non-intrusive, efficient and connected. For the latter, non-intrusiveness remains essential to respect the bees. Efficiency and connectivity are not determining factors: training data can be manually downloaded and the data collection process can be monitored closely as opposed to the final product. From a beekeeper's point of view, knowing the needs depends on the size of the apiary. It also depends on the number of apiaries, whether transhumance is performed, and the money that can be invested. The simple and cheap solutions regroup systems that collect and transfer raw data directly to beekeepers, who then extrapolate to make their opinion thanks to their apiary experience, which is possible. On the other hand, more advanced services provide results that are not

manually reproducible but require extra energy resources to compute AI models on-site.

The PB literature shows that many papers choose to focus on common problematics. They show their method and their results through metrics that they each select, and such metrics vary from one paper to another, although the initial question is the same. Also, the quality of validation is sometimes lacking. The sound and vibration-oriented literature review presented in [Uthoff et al., 2023] states that there is a lack of a common evaluation metric for the queen bee and swarming detection. As shown in Table 3.2, the number of unique hives is often close to one — future works should focus on the robustness of the analyses by using a larger number of unique beehives for training data and explore more features apart from MFCC, which is not initially designed for bees' sound analysis. Also, in general, PB data analysis works should collect data from hives over several seasons, in order to gather insights about the long-term dynamics of bee colonies.

According to [Borlinghaus et al., 2022], 63% of bee counting solutions proposed in the literature are not validated at all. This paper proposes a protocol to evaluate bee counting methods. It answers the call of [Odemer, 2021], the survey on bee counting devices which calls for a method to validate a counter's precision. It mixes existing approaches, it is time-consuming but only needs to be performed once per system. Such a protocol could guarantee the accuracy of future solutions. This contribution is an important step toward reproducible PB research and should be replicated for other types of services.

3.3 Conclusion of the literature review

3.3.1 The state of precision beekeeping research

Precision beekeeping (PB) is still a new and emerging research topic: there are many publications in recent years, and those come from various scientific backgrounds, from computer science to agriculture.

It is not easy to generalize results found in the literature, first, because the life cycle of bees continues to evolve after the data is collected. Then, the experimental conditions are unique for each project since they are conducted at different times, places, and with variations in the design of the beehive, or even the subtype of bee species.

However, all the cited sensors, network types, energy nodes, data and artificial intelligence (AI) models show that there are some coherence and standard bases between PB literature pieces. Thanks to research initiatives and their accessibility, experiments that would have been instantaneously moved on from in the past, can be the source of new ideas and advancements in this field.

Tracking the research in PB remains tricky because the systems, services, and data are directly linked to business considerations. Several working groups appear to be intensely productive during a short time window and then disappear from the scientific community, most often because they stop publishing their findings, protect them, and start a business (ApisProtect from [Murphy and Whelan, 2017]).

3.3.2 The needs for precision beekeeping research

Looking at beekeepers' needs, it is clear that more affordable commercial solutions are to be developed. There is, of course, a duality between developing a unique solution that could regroup data from all beekeepers and build AI models, and the economic competition. The latter can still push PB in the right direction if many actors strive for cheaper and more adapted solutions. This survey helps to better understand this challenge with Section 2.3 which helped to understand the expenses when adopting precision beekeeping services, and the two following sections (Sections 3.1 and 3.2) which explained the potential benefits in terms of services in surveyed research papers. It is however difficult to imagine the return on investment for beekeepers when adopting the solutions described in the latter two sections and scaling them up in terms of quantity since every service described uses DIY installations that are hard to mass-produce.

Bees are exposed to many stress factors, some better understood than others, and PB is considerably improving experimental conditions, particularly with the increasing data volumes. The state of PB research today shows solid advances with data like weight and temperature. During the past few years, sound and image-based methods came into play and opened new perspectives. The question is, how far can it lead us? So far, most of the projects are not reproducible. The next step is to integrate this expanding theoretical knowledge into beekeepers' everyday life. However, is it worth equipping all beehives with sensors? Probably not, first because the return on investment is not worth it, and then because different colonies still share the same environment.

Electromagnetic fields are also correlated with honey bees' health. Although there is less research on it than other stress factors, [Shepherd et al., 2019] showed an increase in aggressiveness and a decrease in the ability to learn when bees are exposed to low-frequency electromagnetic fields, at rates that are comparable to those found around power lines. [Henry et al., 2019] compared the internal temperature and relative humidity between wireless-connected beehives (2.4 GHz Wi-Fi) and their wired twins for 30 days. No effect on bees was observed during that period. It remains unknown whether PB research results are biased because of this reason.

3.3.3 The future of precision beekeeping research

From a biological perspective, the 20^{th} century has seen manual observations emerge, proving fundamental apiary knowledge. From the beginning of IoT systems around 2010 to today, digital technologies helped to exponentially improve the quality and quantity of data collected, as well as the growth of apiary research all over the world. It confirmed previous biological conclusions. From now on, the hope is that more and more connected beehives will arise and help to produce large-scale studies searching for new apiary knowledge.

This survey highlights the fact that PB research as a system and PB research as a service are often two different things: some papers focus on IoT technologies and how to efficiently deploy a PB system, whereas others focus on machine learning tasks and classification performances. We encourage PB research initiatives to cover the full spectrum and take into account the constraints of their services for the design of their

system, and vice-versa. The artificial intelligence techniques used as parts of PB services are often energy-consuming, thus making their deployment difficult. The future of PB lies in the integration of efficient, operational, and deployed AI models. To do so, the energy footprint of end-to-end systems needs to be analyzed to enable autonomy for a complete solution.

The following chapters propose answers to the questions raised in this last paragraph. In Chapter 4, we contribute to efficient precision beekeeping with the design, implementation, and deployment of an energy-aware precision beekeeping system and its services. Chapters 5 and 6 propose solutions for energy efficiency and autonomy in IoT systems similar to smart beehives with computer simulations and the optimization of energy consumption for the allocation of resources and tasks.

Design and deployment of a precision beekeeping system

Contents for chapter 4

This chapter shows a power consumption analysis, presents the design of our precision beekeeping (PB) system, and analyzes the data collected by our system. First, the power consumption of the precision beekeeping system introduced in [Ammar et al., 2019] is analyzed in Section 4.1. This system will be referred to as the *Makers' Beehive*. The power consumption analysis allows to identify the limits of that system and to formulate improvements. Secondly, our proposed system is introduced in Section 4.2, which is an energy-aware precision beekeeping system and which will be referred to as the *EAPBS*. Its architecture and its deployment are presented. Thanks to the data collected in 2022, which we share, Section 4.3 contains the first exploration of the multimodal dataset, and shows the potential for AI-based services. Finally, Section 4.4 concludes this chapter with a summary of the results.

4.1 Energy benchmark of an existing PB system's hardware

In this section, we aim at identifying the areas of improvement of the *Makers' Beehive* in terms of value for energy consumed. The energy consumption of different steps (idle, boot, shutdown and stress) and specific conditions (different temperatures) are measured.

4.1.1 System and data collection cycles

The data collection node of the *Makers' Beehive* system (Figure 4.1) consists of a Raspberry Pi 3 Model B microcomputer, a Raspberry Pi camera Rev 1.3, and an Arduino Uno board which is based on the Microchip ATmega328P microcontroller together with its sensor hat and sensors. When powered by the battery, the Raspberry Pi 3b powers the Arduino Uno and is able to receive data through its USB port.

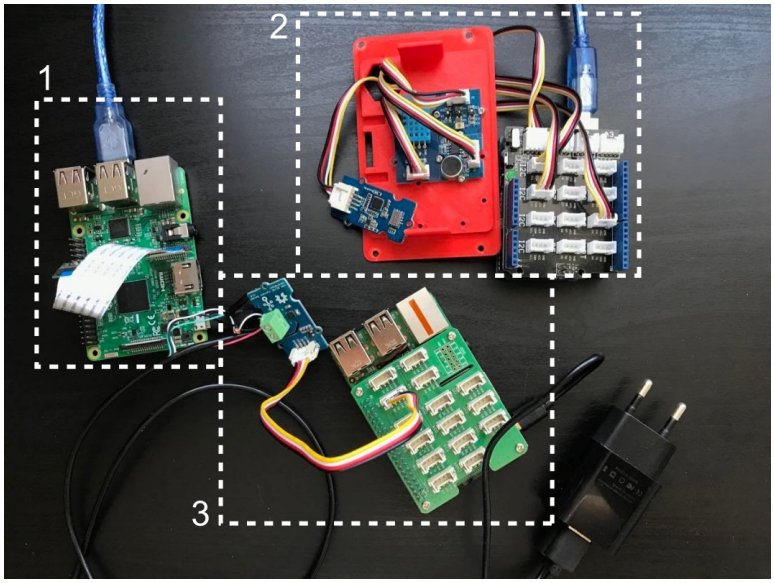


Figure 4.1: Picture of the system introduced in [Ammar et al., 2019] (Box 1: Raspberry Pi 3b and Raspberry Pi camera; Box 2: Arduino Uno, sensor hat and sensors). On top of this system and for the purpose of this work, a current-measuring node is added (Box 3: a second Raspberry Pi and a current sensor).

When deployed, the *Makers' Beehive* performs the following routine at regular intervals:

Step 0: Raspberry Pi 3 b's boot;

Step 1: Collect environmental data (temperature, humidity, noise, light, Carbon monoxide, Nitrogen dioxide and the beehive's weight);

Step 2: Capture a series of 20 top-view pictures (one per second);

Step 3: Convert the images into one GIF file. It is then uploaded to an image-sharing website and allows beekeepers to inspect the entrance of their beehive at a glance;

Step 4: Upload the measurements and the URL of the GIF file and pull the current state of the script's repository;

Step 5: Raspberry Pi 3 b's shutdown.

In practice, the system is granted a time interval of 5 minutes to accomplish the data acquisition process. Once the data is successfully uploaded, the Raspberry Pi shuts off completely. However, if the data acquisition process is not accomplished within the granted time, the system is forced to shut down. It is worth noting that the frequency at which the deployed system used to acquire data in 2019 and 2020 (years of operation) was set at one hour. This frequency impacts the overall energy consumption. However, we will focus here on the cost of the data acquisition routine which is independent of the wake-up frequency.

4.1.2 Current collection node

The current-measuring node added on top of the *Makers' Beehive* consists of another Raspberry Pi 3b which is independently powered, and its current sensor is attached to its sensor hat. This extra cost of hardware ensures that the measurements do not impact the measured system.

The current sensor itself is a Grove ±5A DC/AC current sensor[1] based on the Allegro ACS70331 current sensor[2], which is based on giant magneto-resistivity sensing technology. This sensor is selected because of its adapted sensing range (0 to 5 A) and its wide temperature operating range (-40°C to 85°C), as the later-introduced setup operates in a sealed case in an outdoor environment. The current collection script is written in Python 3 and relies on Grove's package functions.

4.1.3 The cost of the Arduino

Energy profiling of the idle phase

A first step toward analyzing the energy consumption of the *Makers' Beehive* and its tasks is to understand the fixed costs of the hardware. The consumption of the data-acquiring Raspberry Pi is recorded while in idle mode — the Raspberry Pi being logged in and the operating system launched at a resting state. This experiment repeats itself four times, to determine the power consumption of all combinations of peripherals. First, the full setup is tested (Arduino sensor node plugged in and Pi camera plugged in). Then, each peripheral is plugged in alone. Finally, the idle consumption of the setup without any peripheral is recorded.

Figure 4.2 shows the extra 0.5 W of power cost by the connection of the Arduino sensor node via USB. It is important to note that the energy impact of the camera peripheral (not in use actively, like the Arduino) is negligible.

[1] https://wiki.seeedstudio.com/Grove-5A_DC_AC_Current_Sensor-ACS70331/
[2] https://www.allegromicro.com/-/media/files/datasheets/acs70331-datasheet.pdf

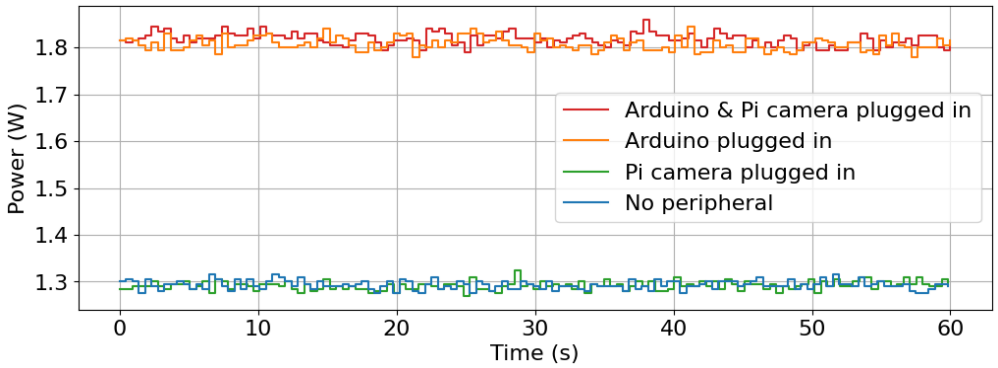


Figure 4.2: Idle power consumption of the *Makers' Beehive*'s Raspberry Pi 3b over a 60-second time window

Energy profiling of the boot and shutdown

The boot and shutdown are two prominent phases as they always occur once per routine. Figures 4.3a and 4.3b show the power evolution of a standard boot-up phase (with Arduino plugged in) and the comparison of two shutdown phases.

Boot For the boot-up phase, the recorded Raspberry Pi is turned off (0 to 10 s window in Figure 4.3a), plugged out of the power supply (10 to 20 s) and plugged back in (20 s mark).

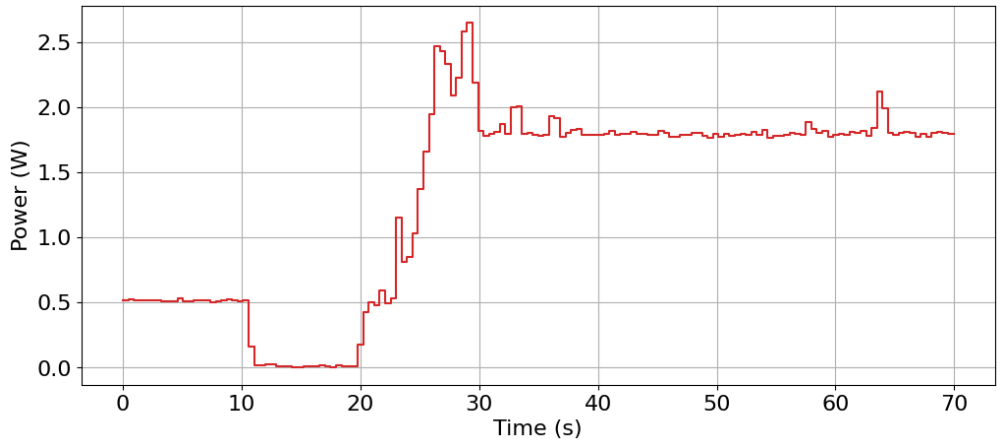
The boot-up phase, due to its quickness (10 s, between the 20 s and 30 s mark on Figure 4.3a), requires 14.5 J: the time window where the power is greater than the idle phase is only 4-second long (26 to 30 s).

Shutdown For the shutdown phase, a graceful shutdown approach is used. During a graceful shutdown phase, all running processes are sent a shutdown message. It is only once those processes are closed that the interface is turned off and the filesystems unmounted (described in [Amirtharaj et al., 2018]). A forced shutdown alternative also exists. It does not notify the running processes of the incoming shutdown. In our case, both were evaluated and although the forced approach turns off the system in less than one second, the small gain of energy is not worth the risk of potential memory errors. The presented results focus on the graceful approach.

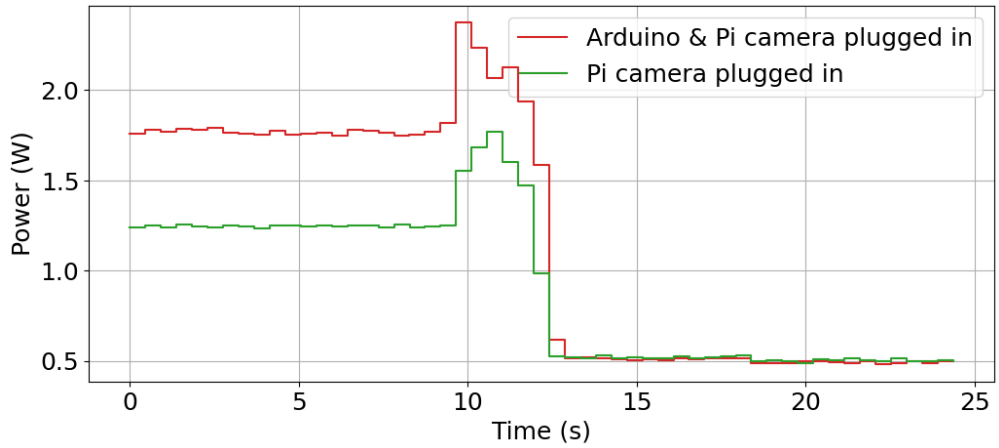
The results of Figure 4.3b follow the previous section's findings, as the extra USB peripheral adds a fixed cost of energy while turned on. If the Raspberry Pi is turned off and still plugged in, there is no effect from the Arduino. The slight peak of power usage at the moment of the shutdown (10 s mark) is explained by the messages sent to running processes and the stop of these processes.

Comparing the value and the cost of the Arduino

In the *Makers' Beehive*, the Arduino is here to act as a data collector. It is the intermediary between the Raspberry Pi 3b and the sensors. However, a Raspberry Pi-based



(a) Power consumption of the full setup during boot-up



(b) Power consumption of the full setup and Arduino plugged out during shutdown

Figure 4.3: Energy profiling of boot-up and shutdown of the *Makers' Beehive*'s Raspberry Pi 3b

system could function identically without an Arduino, by using the same sensor hat as the current-collecting Raspberry Pi and plugging the same sensor into it. The value of the Arduino is negligible, as it does not perform any other task than collecting data. In conclusion, the Arduino and its energy consumption cost will be removed in the upcoming proposed system.

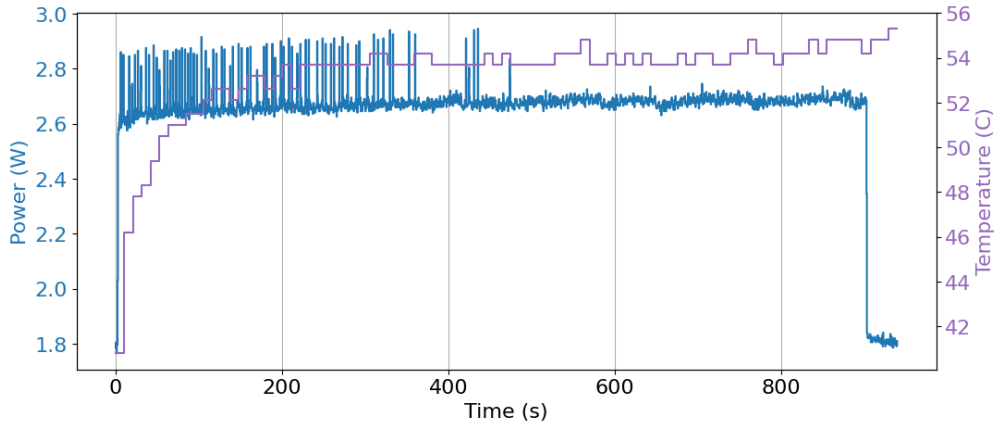
4.1.4 Validation of the use of a Raspberry Pi 3

Energy profiling of stress tests

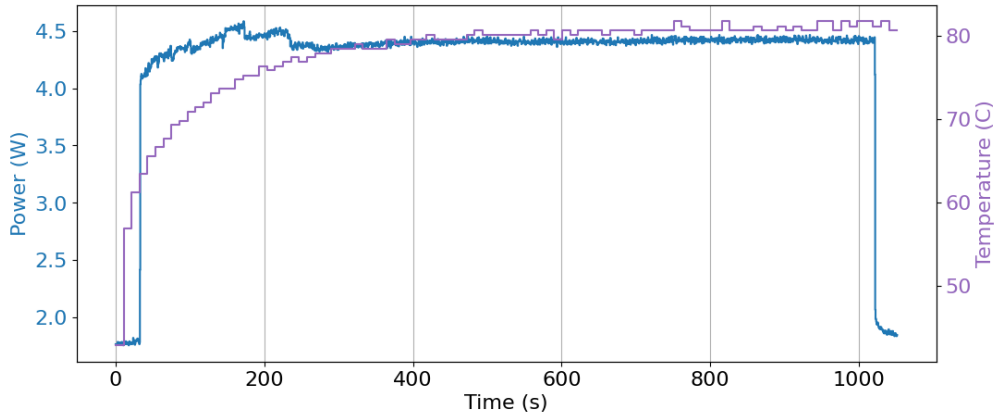
To better understand the limits of the services of a Raspberry Pi 3b, its performances in extreme situations are analyzed. Stress testing aims at pushing a CPU to its full power.

Two stress tasks are benchmarked, they are both available on a Raspbian Lite[3] operating system:

- Stress (`stress`): here, the CPU is run at its full power, all the while controlling the temperature of the chip.
- CPU Burn (`cpuburn-a53`): here, the Raspberry Pi is maxed out completely without any restriction.



(a) Power consumption of stress task for the *Makers' Beehive*'s Raspberry Pi 3b



(b) Power consumption of CPU burn task for the *Makers' Beehive*'s Raspberry Pi 3b

Figure 4.4: Energy profiling of stress and CPU burn tests of the *Makers' Beehive*'s Raspberry Pi

Figure 4.4 shows the temperature and power consumption for the two tasks over several minutes. The absolute maximal power used by the Raspberry Pi 3 lies around 4.5 W, whereas the safer approach only requires between 2.6 W and 2.7 W, which is comparable to the energy of some tasks performed in the field.

[3] https://www.raspberrypi.com/software/operating-systems/

System at different temperatures

Precision beekeeping systems are deployed on-site in wild outdoor zones. Such systems are supposed to run without interruption. How can varying temperature conditions impact the energy consumption of the system?

Experimental Conditions The Raspberry Pi 3b is placed in two different temperature conditions: at a cold temperature (inside a fridge; between 3°C and 5°C) and at room temperature (between 19°C and 22°C), all the while maintaining power supply connection. For each one, ten iterations of the routine's script are executed, each of them followed by a 3-minute break, to retrieve a resting state.

Since the pictures captured in an open place (room temperature situation) are different from the ones taken inside a closed dark fridge (cold temperature situation), the size can differ too, leading to different performance for compressing images and converting them into a GIF file. To avoid this potential issue, the same batch of images is always chosen to create the GIF file.

Results Over 10 iterations of the main script, the average energy consumption is on average 169.6 joules at room temperature, with a standard deviation of 3.1 joules (average temperature of Raspberry Pi chip: 45.7°C; standard deviation: 0.6°C). At cold temperatures (between 3°C and 5°C), the average consumption of energy is slightly impacted with an average consumption of 186.4 joules and a standard deviation of 4.6 joules (average temperature of Raspberry Pi chip: 30.4°C; standard deviation: 0.5°C). One iteration can be observed in Figure 4.5.

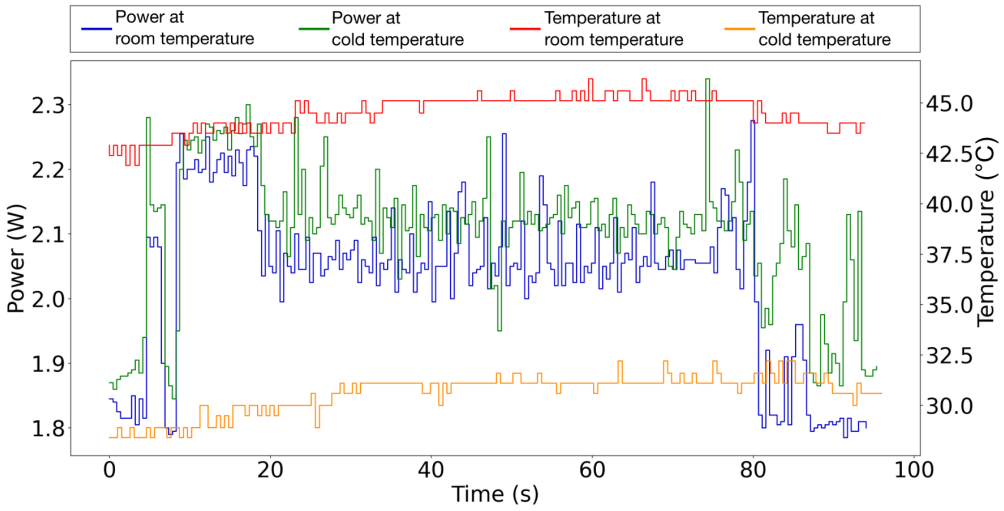


Figure 4.5: Power and temperature of the Raspberry Pi chip for one script run at different temperature conditions

Practically, during its past deployment, the *Makers' Beehive*'s system (full setup with Raspberry Pi and Arduino plugged in) used to alternate between a 60-minute OFF mode and the 5-minute data collection window. The latter is a combination between

the Raspberry Pi boot-up, the execution of the script, and the shutdown. According to the observations in the previous sections, the total cost of energy is 1800 joules for the 60-minute OFF mode phase and 548 joules for the 5-minute script window. All in all, that is an average of 2167 joules per hour: 120.4 mAh at 5 V. In the example of the deployed system in [Ammar et al., 2019], the capacity of the solar-powered battery is 33 000 mAh. If a fully charged battery were to lose its charging source, the system would still function for around 11 days and 10 hours (274 cycles of 1 hour).

Raspberry Pi's maximal capabilities compared to our needs

Considering the temperature reaching 80°C, we do not wish to replicate the same computation load as in the CPU burn in our future system. On the other hand, stressing the Raspberry Pi in a setting close to the stress test is acceptable because it is 10 degrees apart from the chip when the system is performing its routine at room temperature. The long-term deployment in outdoor conditions leads to variations in temperature. Perhaps that actual ambient temperature could be warmer than room temperature, so the chip's temperature during the normal routine could approach the stress test's temperatures. In France, the summer highest temperatures can sometimes reach close to 40°C, and go down to below 0°C in winter. The coldest conditions will penalize the energy consumption, but only by 10%. These variations are variable to accept for the design of the future system.

4.1.5 Energy optimization the data acquisition phase

Energy profiling of the main routine

Here, we analyze the energy consumption of the data acquisition process. Over 10 independent measures, here are the following averaged energy values for each step: *Step 1:* 6.2 J; *Step 2:* 23.7 J, *Step 3:* 126.1 J, *Step 4:* 13.5 J.

Figure 4.6 shows the overall consumption of the execution of one in-hive routine script. It is divided into four energy phases, one for each of steps 1 to 4 as introduced in Section 4.1.1, allowing to estimate the power cost of each. The most energy-consuming phase is the longest: the third one (20 to 80 s window) — step 3 — which converts images into a GIF and uploads the GIF file to an image-sharing website. For an improved version of this routine, it is key to focus on optimizing the conversion or modifying its method, and on decreasing the load created by the transfer of a such large file.

Focus on the network tasks

The flow of data transmitted in the *Makers' Beehive* is bidirectional. On the one hand, the Raspberry Pi updates the software at each iteration of the main loop: a download triggered at step 4 of the routine as seen in Figure 4.6. On the other, data from environmental sensors and the GIF file are uploaded to a server (end of step 3 and step 4 of the routine).

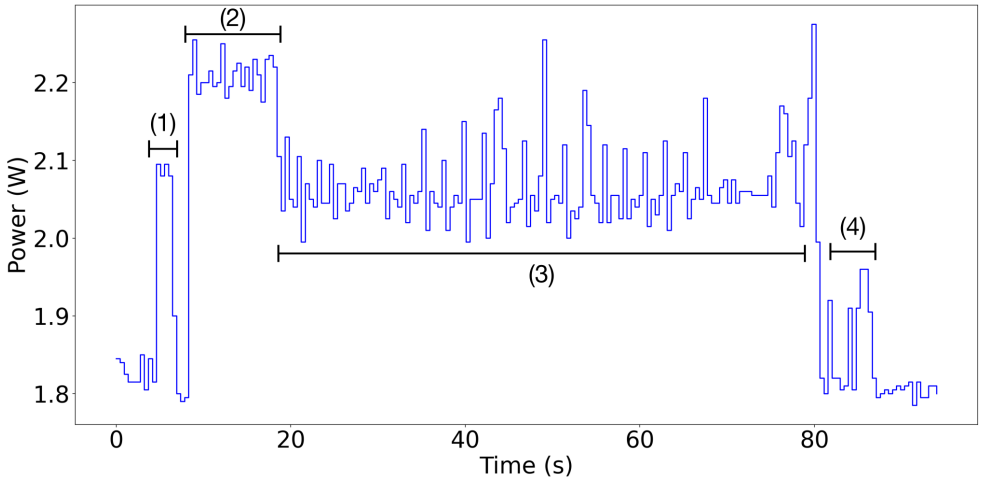


Figure 4.6: Power usage profile of the *Makers' Beehive*'s Raspberry Pi 3b for one execution of the in-hive script

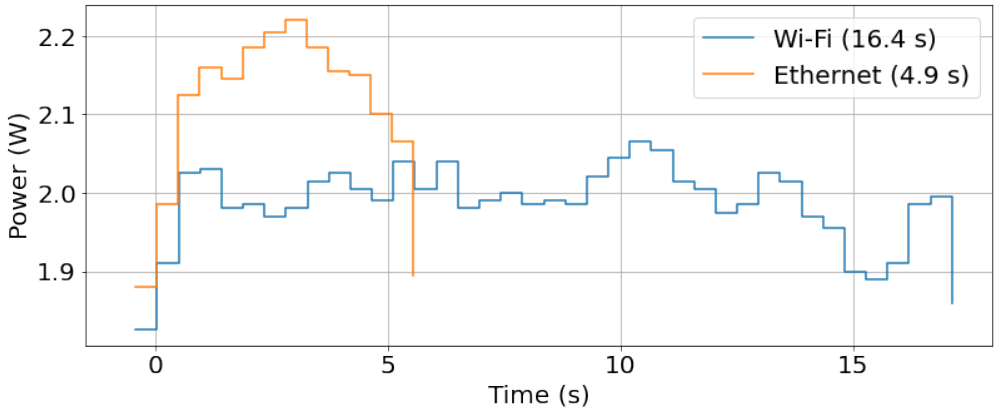
The goal of this analysis is to profile the data transmission from an energy standpoint for both wired and wireless connections. All scenarios are measured using the iPerf3 tool [Dugan et al.,], which is a network performance measurement tool and allows the transfer of network packets at the maximum achievable bandwidth.

Experimental Conditions The replica of the in-hive Raspberry Pi is connected to a local network, located in Lyon, France (Internet provider: SFR). This network has a maximum download speed of 190 Mbits/s and a maximum upload speed of 20 Mbits/s. The Ethernet cable has a maximum speed of 100 Mbits/s. The server which is used to test sending and receiving data is a public server located at `bouygues.iperf.fr`[4]. Its bandwidth is 10 Gbits/s.

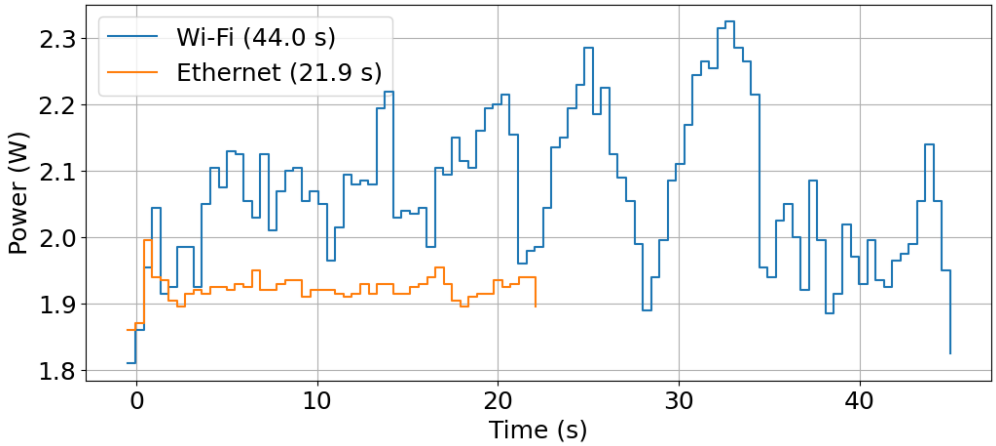
Once the Raspberry Pi is connected to the network (either through Ethernet or Wi-Fi), a 5 MB upload is tested with the command `iperf3 -c bouygues.iperf.fr --bytes 50M`, as well as the download equivalent (`-R` added at the end of the command). 50 MB is enough to get at minimum a few seconds of execution for the fastest rate of transfer (downloading through Ethernet). The Internet transport protocol used is TCP (Transmission Control Protocol) and there is no bandwidth restriction.

Results The results in Figure 4.7a and in Figure 4.7b show the comparison between the types of connection. The wired connection is faster than the wireless: 4.6 seconds versus 15.6 seconds on average for the 50 MB download and 20.8 seconds versus 42.1 seconds on average for the 50 MB upload. Those values are similar to the averages in Table 4.1 but different from the unique runs in Figure 4.7. However, downloading data with a wired connection reaches a higher peak of power than the latter, while uploading data displays the opposite results.

[4]The experiment took place in 2020 and this server is not available anymore. A list of equivalent servers can be found at `https://iperf.fr/iperf-servers.php`.



(a) Power profile of one 50 MB download



(b) Power profile of one 50 MB upload

Figure 4.7: Energy profiles of iPerf3 upload and download tasks performed by the *Makers' Beehive*'s Raspberry Pi 3b on the `bouygues.iperf.fr` server.

Table 4.1 helps to identify the limiting factor and the pressure point in each scenario:

- While downloading through Wi-Fi, the speed is limited by the Raspberry Pi's Wi-Fi card.
- While downloading through Ethernet, the speed is limited by the Ethernet cable.
- While uploading through Wi-Fi, the speed is limited by the Raspberry Pi's Wi-Fi card.
- While uploading through Ethernet, the speed is limited by the local network's upload bandwidth. This could explain the small increase of power in this case.

Table 4.1 also shows that it is more efficient to use an Ethernet connection while exchanging data. However, while in idle mode or performing other tasks that do not stress the network, having an Ethernet cable plugged into a Raspberry Pi introduces an extra energy cost of 0.085 W. It represents the energy required to handle the Ethernet peripheral.

	Throughput		Power consumption	
Network	download	upload	download	upload
Wi-Fi	25.7 Mbits/s	9.5 Mbits/s	34.0 joules	93.9 joules
Ethernet	86.4 Mbits/s	19.2 Mbits/s	10.8 joules	42.5 joules

Table 4.1: Average energy consumed for network tests (transferring 50 MB)

Even though the Ethernet connection is a more energy-efficient option for data transfer, the practical constraints of the deployment of a precision beekeeping system force, in our case, to use a Wi-Fi Internet connection. Indeed, the newly deployed system will be placed on the roof of the building of a university, where no wired Internet connection can be established.

The modifications needed on the *Makers' Beehive*'s routine

The system analyzed in this section is a functioning data collection tool. It can be deployed inside a beehive and provide meaningful information about bees. However, this thesis aims to provide a more energy-efficient approach, so the system tested in this section requires some improvements.

First, the Arduino part of the system increases the idle power consumption by 0.5 W and only serves as a data collector, which is a role that the Raspberry Pi itself can guarantee. Therefore, this part will be removed from the new proposed system.

The other main line of improvement with the *Makers' Beehive*'s routine is *Step 3* which accounts for 75% of the energy consumed by the Raspberry Pi during the routine. This step is an image processing phase. One could think of positioning this service in a distant server rather than directly at the precision beekeeping (PB) system, extending the life expectancy of the system. Also, creating a GIF file is purely a gain of aesthetics, it does not offer more information than the initial images used to create it. Additionally, the size of a GIF is at least the same as the sum of the sizes of the images which served to create the given GIF, if the resolution is not affected. Another approach is then to anticipate the needs of the models used to provide PB services and perform the right image processing step, if any. In the newly proposed system, depending on the scenario, the choice is to either transfer images as they are or pass them directly into the computer vision models.

4.2 Energy-aware precision beekeeping system

This section aims to describe a newly-proposed autonomously running solution for collecting bee-related data capable of monitoring its energy consumption, which will be referred to as the *EAPBS* (energy-aware precision beekeeping system). Bee-related data is used to gain an understanding of the monitored colonies and more generally apidology. On the other hand, energy data is used as a basis for the orchestration of energy-aware services.

The main goal is not to propose a system capable of answering all beekeepers' needs or a system optimized for deployment in the context of pure colony monitoring. Thanks to its distinctive feature of current measurement, we rather want this system to help to gain knowledge on the energy consumption of precision beekeeping (PB) systems, and even Internet of things (IoT) systems close to PB systems.

4.2.1 Hardware

The *EAPBS*, as shown in Figure 4.8, relies on two Raspberry Pis, which are single-board mini-computers.

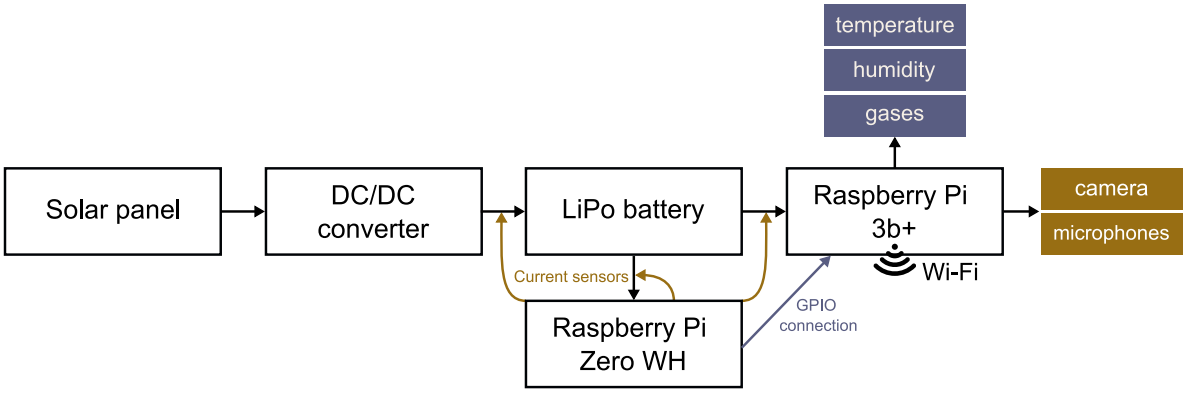


Figure 4.8: Architecture of the *EAPBS* showing essential hardware, energy node, and network

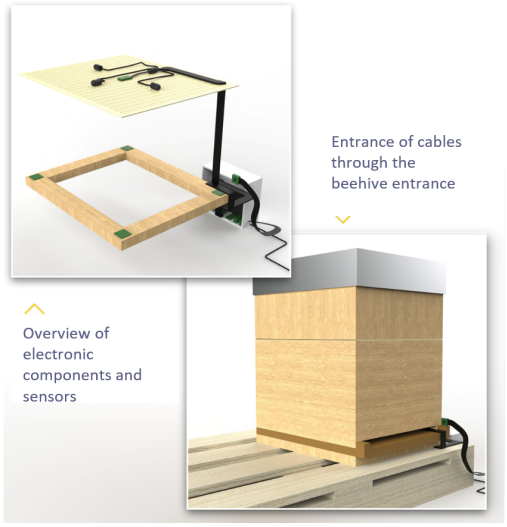
First, the Raspberry Pi 3b+ is the most complete of the two as it includes four USB 2.0 ports, a port for a Raspberry Pi camera, 2.4 gigahertz and 5 gigahertz IEEE 802.11 wireless LAN and Bluetooth connectivity, as well as a quad-core 1.4 gigahertz 64-bit processor and 1 gigabyte of RAM. In comparison, the second one, a Raspberry Pi Zero WH, is a lightweight version of its sister model with one micro-USB port, a camera connector, the same Wi-Fi and Bluetooth connectivity, a 1 gigahertz single-core CPU, and 512 megabytes of RAM. These kinds of computers are selected because they are widely known solutions for IoT devices. They run on a Unix operating system enabling flexibility in terms of coding languages and libraries, and because each of them is pushed to its connectivity and computation limits in this system. Later, it will also stand out because of its ability to combine multiple artificial intelligence models.

On top of each Raspberry Pi, a Grove Base Hat [5] is installed to wire sensors.

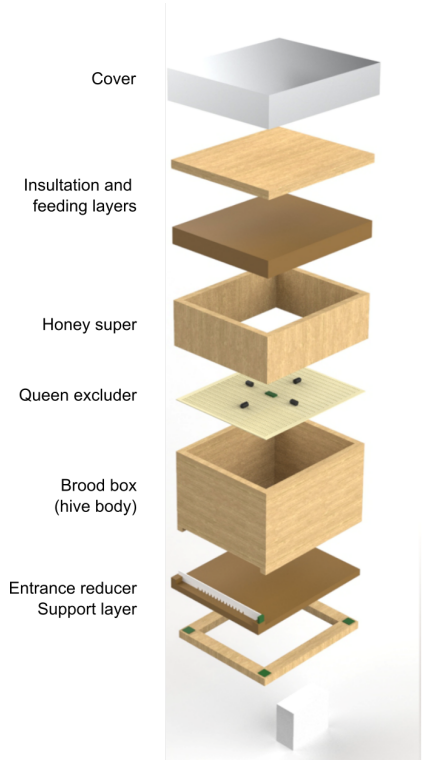
For the Raspberry Pi Zero WH, its hat sensors are three identical ±5 A DC/AC current sensors (introduced in 4.1.2). Each of these sensors is linked to an electric wire: the two Raspberry Pi's power supplies and the wire that goes from the solar panel to the battery. For the Raspberry Pi 3b+, a temperature and humidity sensor (SHT31) [6] and gas sensor (carbon monoxide, nitrogen dioxide, ethyl alcohol, and volatile organic compounds) [7] are used. In addition to hats' sensors, the Raspberry Pi 3b+ collects

[5] https://wiki.seeedstudio.com/Grove_Base_Hat_for_Raspberry_Pi/
[6] https://wiki.seeedstudio.com/Grove-TempAndHumi_Sensor-SHT31/
[7] https://wiki.seeedstudio.com/Grove-Multichannel-Gas-Sensor-V2/

sound through three USB microphones (range: 20 hertz to 16 kilohertz) and images from a Raspberry Pi camera module 2 [8]. The single-board computers, their power cables, the battery and the current sensors are all placed inside a sealed box that only allows cables to go through. The wires of the temperature and humidity sensor, the gas sensor, and the microphones are placed inside the beehive on the queen excluder. These wires go from the hardware box to the hive through the bottom part of the hive, go up along a side wall, and are placed close to the middle of the queen excluder grid (see the top part of Figure 4.9a). The camera is isolated from outdoor disruptions with a 3D printed case (see Figure 4.10). It is the only bee-related sensor of the *EAPBS* placed outside the hive, on one edge of the entrance of the hive (see Figure 4.11). This case only allows the camera's cable and lens to pass through the plastic. Its support (left side of Figure 4.10) is designed to fit the depth of the entrance of a Dadant beehive, which is a standard beehive model, and it is used for our cases. The middle and right parts of Figure 4.10 act as a shell for the camera. In practice, the camera faces the other end to take pictures of the whole hive's entrance area.



(a) Diagram of a connected beehive and its sensor node, showing cables going inside along the side of the hive



(b) Layers of the beehive part of the *EAPBS*

Figure 4.9: Diagrams of the *EAPBS*

[8] https://www.raspberrypi.com/products/camera-module-v2/

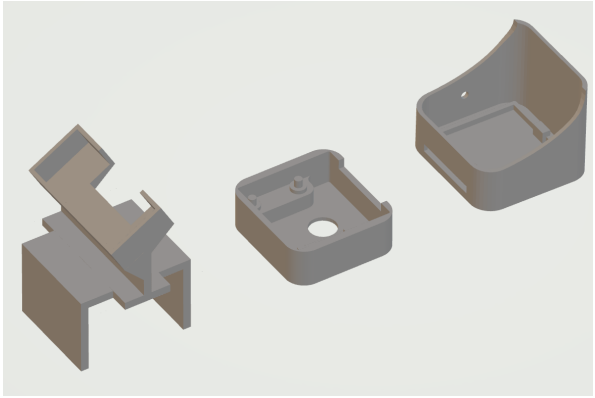
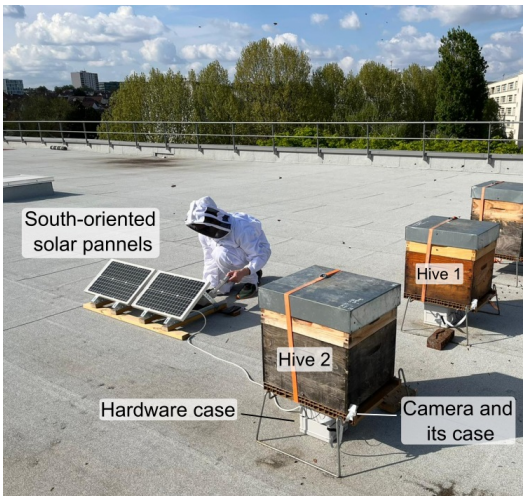


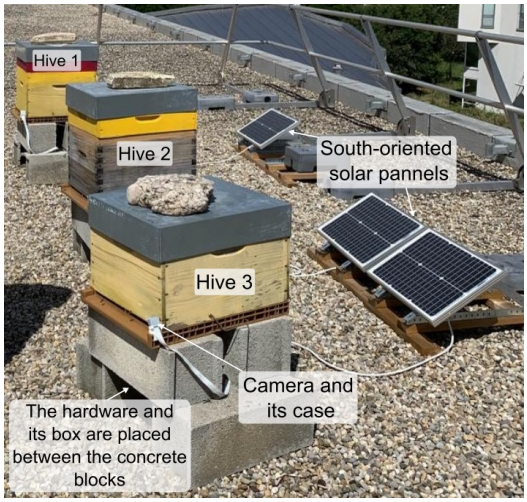
Figure 4.10: 3D model for the Pi camera's case

4.2.2 Deployment

Location Five smart beehives are currently deployed. Two are located to the South of Paris in Cachan, France (see Figure 4.11a), and the others are in Lyon, France (see Figure 4.11b). All of them are positioned on a university building's roof (see Figure 4.11) and are in an urban area with parks bordering both campuses. Such locations make the deployment safe, as theft is a risk for beehives and especially beehives with digital hardware in the wild. It also facilitates adjustments and maintenance.



(a) *EAPBS* on the roof of aivancity's campus (Cachan, France)



(b) *EAPBS* on the roof of ENS de Lyon (Lyon, France)

Figure 4.11: Pictures of the *EAPBS* deployed in Cachan and in Lyon

Energy Each *EAPBS* relies on a solar panel, current converter, and battery for its energy. The solar panel is a monocrystalline 30 W panel. The battery is a 20 000 mAh power bank. Between the two, the current is converted to 5 volts (the adequate voltage for the battery) with a DC/DC Step-Down 5 volts/3 amperes converter.

Price The price listing of one *EAPBS* can be found in Table 4.2. The whole system costs 373 EUR (data of January 2023), and its price is distributed over the energy items (92 EUR), the sensors (108 EUR), the Raspberry Pis and their hat (143 EUR) and the waterproof cases (30 EUR).

Component	Quantity	Price per unit in euro
Solar panel	1	37
DC/DC Converter	1	10
Powerbank battery	1	45
Hardware Box	2	15
Gas sensor	1	36
Temperature/Humidity sensor	1	15
Current sensor	3	6
Raspberry Pi camera	1	28
Microphone	1	11
Raspberry 3b+ kit	1	82
Raspberry Zero WH kit	1	40
Raspberry 3b+ sensor hat	1	11
Raspberry Zero WH sensor hat	1	10
Total		373

Table 4.2: Price of components for one *EAPBS* (January 2023)

Time frame The *EAPBS* was first tested in early 2022 with an empty beehive for it to run for several weeks and check whether data was correctly collected and transferred. Then, systems from both locations were deployed inside beehives in Spring 2022, which received new colonies of bees. Before each colony was moved inside its hive, sensors, and wires were installed inside the hive's bodies (as seen in the exploded view from Figure 4.9a) to reduce the disturbance toward the development of young colonies.

Data collection routines The Raspberry Pi Zero, which records the current, is always switched on. At regular intervals, it sends a signal through GPIO to wake up the other Raspberry Pi, the beehive data recorder. Once the Raspberry Pi 3b+ wakes up, it collects data from all its sensors, including three 10-second audio samples collected at the same time and five 800×600 pixels images spread over five seconds, and it transfers the data through Wi-Fi to a remote data storage cloud server. In addition, at regular intervals, the Raspberry Pi Zero WH transfers the latest consumption data. To complete the data, the weather data from both locations is collected at regular intervals.

4.3 Dataset exploration

In this section, we share the dataset acquired by the *EAPBS* in Lyon and in Cachan over the high season of 2022 and show the potential of building PB services using this data. We choose to share the in-hive temperature and humidity data, and the energy consumption data, as the sound samples and images have not been analyzed yet.

4.3.1 Open-source bee and energy consumption dataset

In an open-source approach to research, the data from the high season of 2022 across the five beehives deployed is made available. The dataset is accessible at `https://doi.org/10.5281/zenodo.7880085` and can be cited as [Hadjur et al., 2023].

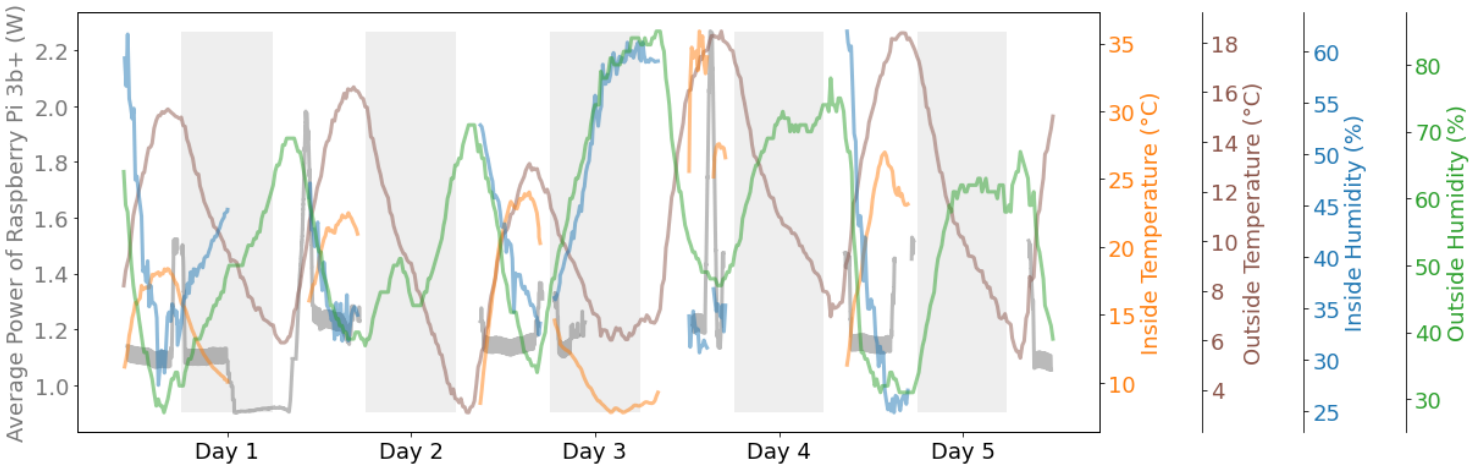
This dataset regroups the temperature and humidity, and energy consumption data. In total, 7786 environmental data points (temperature and humidity) are gathered across all the bee-related parts of the dataset. All bee-related data was gathered every 12 minutes, except for Cachan's hive 2, where this frequency varies. On the other hand, the consumption data is continuously collected as long as the system is functioning.

When all types of bee-related data (sound, images and environmental) are considered, the three EAPBS in Lyon did not gather an equal amount of data due to repeated maintenance on hive 1. Hive 1, 2 and 3 respectively gathered 0.1 GB, 11 GB and 11 GB of multimodal data. In Cachan, hive 1 gathered 0.8 GB, whereas hive 2 gathered 6.5 GB of data. This difference is also explained by the technical difficulties met for the deployment of hive 1. In total, 19 786 audio samples and 38 302 images were collected.

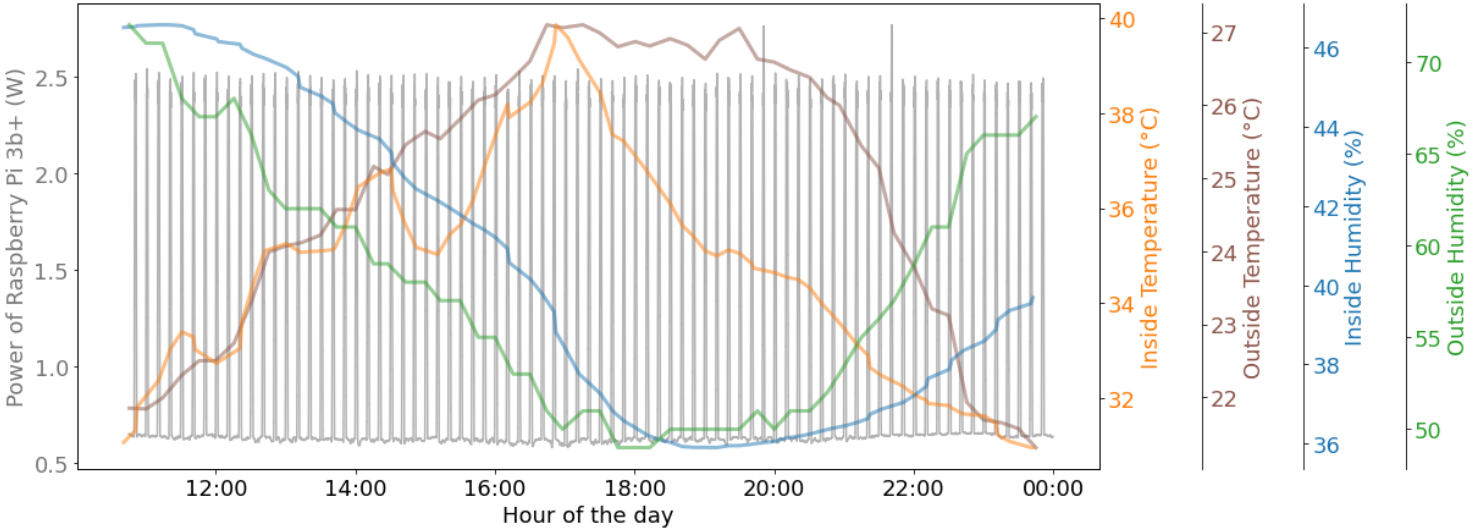
4.3.2 In-hive, current and environmental data

Figure 4.12a displays the activity of the Raspberry Pi 3b+ of one beehive together with the in-hive temperature and humidity collected by the *EAPBS* (inside the hive), and the meteorological data (outside) from a weather API at the same moment.

At this moment, the colony of bees was yet to be introduced inside the beehive, hence the abnormally low inside temperature. The data is shown over one full week to emphasize energy consumption dynamics. In this graph, the gray background shows periods after sunset, whereas the white background shows daytime periods. This graph also shows moments when the system is not running due to the lack of light at night (discontinuous orange and blue curves). The low luminosity is presumed to take the solar panel's output voltage to uncontrolled values, thus affecting the batteries and the electronics. Figure 4.12b highlights the intervals at which the Raspberry Pi 3b+ wakes up during a day when the *EAPBS* was working without interruption. Its energy consumption is shown alongside the in-hive temperature and humidity (the bee colony was inside at that moment), and the meteorological data. The spikes of consumption correspond to the moments between turning on and shutting down the Raspberry Pi 3b+. In this graph, the frequency of the wake-up GPIO signal is set to 10 minutes. This graph shows internal data during the high season in a beehive whose colony was weak at this moment. This could explain the high variability of temperature compared to similar existing measures over a full day in the literature [Zacepins et al., 2016b].



(a) The in-hive temperature and humidity, the external temperature and humidity, and the averaged Raspberry Pi 3b+ consumption during one week of Spring 2022 in Paris (France).



(b) The in-hive temperature and humidity, the external temperature and humidity, and the Raspberry Pi 3b+ consumption showing consecutive wake-ups during 13 hours of one day in Summer 2022 in Paris (France).

Figure 4.12: The collected hive-related and environmental data

4.3.3 Sound samples

As shown in Chapter 3, there is a potential for audio data analysis for PB services, especially for population estimation, swarm detection and queen detection. Our audio samples collected over the data acquisition period reveal the same audio footprint as the ones described in the past literature. Figure 4.13 shows the amplitude of frequencies of one sound sample over time and Figure 4.14 shows the averaged amplitude of frequencies. The latter better summarizes the whole sample: peaks are visible at 142, 264, 346, 429, 673 and 860 Hz. The one at 346 Hz is the least prominent, but the others are visible on the two figures, although lower frequency peaks are less sharp due to the logarithmic scale.

Audio captured by the *EAPBS* [9]

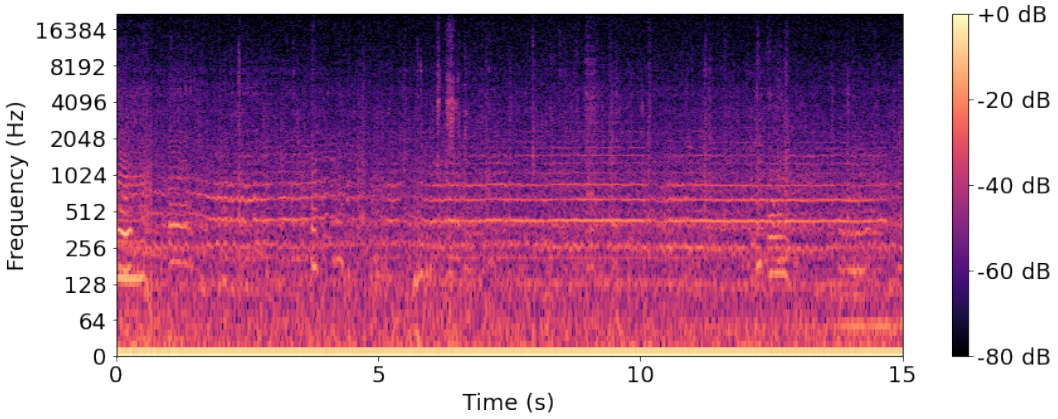


Figure 4.13: Log-spectrogram of a 15-second in-hive audio sample

4.3.4 Images

Images in Figure 4.15 show a selection of noteworthy events caught by the camera of the *EAPBS*: a significant number of bees staying around the entrance (Figure 4.15a), pollen grains being brought back into the hive (Figure 4.15b), a threatening wasp (Figure 4.15c), and a threatening Asian hornet (Figure 4.15d). Although a professional beekeeper would be needed to interpret the first image, the three others' consequences are more straightforward: pollen being brought back is a net positive event, whereas the presence of intruders creates at least stress toward bees and at worst decimation of a colony.

[9] This audio sample, which can be played within this PDF if it is opened, for example, with Acrobat Reader, can also be found at https://perso.ens-lyon.fr/hugo.hadjur/manuscript/audio_sample_ens_beehive.mp3.

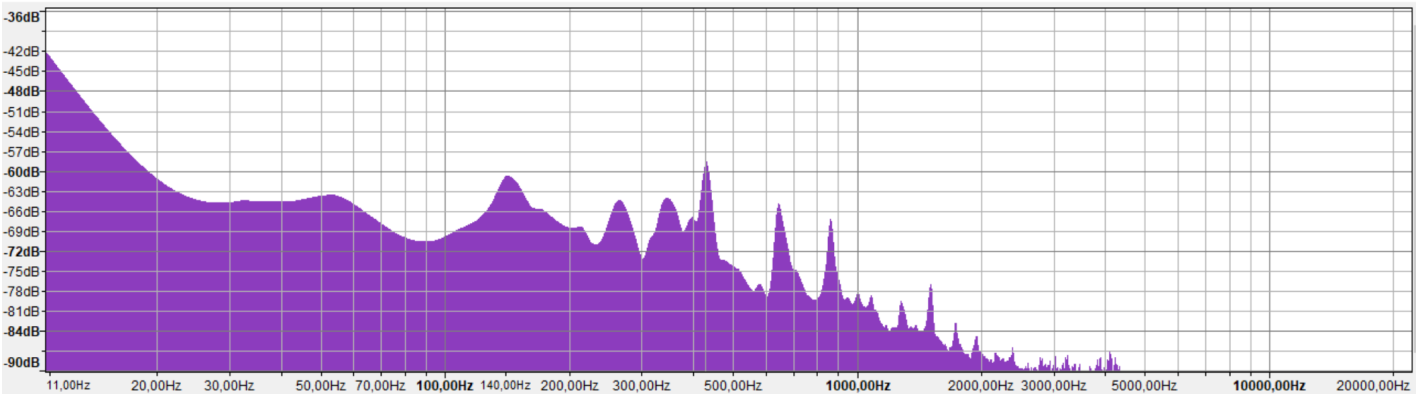


Figure 4.14: Dominant sound frequencies of a 15-second in-hive audio sample

4.4 Conclusion of the design, development of the PB system

The work described in 4.1 is an important prerequisite for all the following work presented in the thesis. It allows criticism of an existing solution to optimize the upcoming use of the hardware. Removing the Arduino from the initial system decreases its energy consumption by 0.5 W during the execution of the data acquisition routine. On the other hand, we validate the use of the Raspberry Pi 3 as the central edge device because of its better computation and connectivity capabilities compared to other single-board computers. We also show that a Raspberry Pi 3 is capable of holding for 10 minutes the computational intensity of a stress test while keeping its chip temperature below 55°C. This allows to safely select this single-board computer as the one to be deployed at potentially high temperatures, especially inside a box during summer, which will be the case. Also, the most energy-consuming phase of the data acquisition and transfer routine, the processing of images, does not bring value to the data yet, so this step is removed from the routine of the proposed system.

Our presented system relies on one Raspberry Pi 3 and one Raspberry Pi Zero. The Raspberry 3 gathers multimodal data from inside a beehive with a temperature and humidity sensor, a gas sensor and microphones. Additionally, a camera is placed right at the entrance of the beehive and lies inside a sealed case. The Raspberry Pi 3 wakes up at regular intervals to gather data from its sensors and sends it to a remote server through Wi-Fi. In addition to the capability of collecting several kinds of bee-related data, our system has a unique feature that so far, has not been seen in the PB literature: thanks to its Raspberry Pi Zero and three current sensors, it continuously collects the energy consumption of the Raspberry Pi 3, the Raspberry Pi Zero and the energy production of the solar panel. This system is the fundamental element to later apply the energy optimization models, which focus on a wider range of IoT objects, to a PB-specific case.

The following chapters will primarily focus on the optimization of energy in IoT systems applied to PB, but this chapter also offers a playground for future bee-related

(a) Bees staying around the entrance (Lyon Hive #3)

(b) Bees bringing back pollen (Lyon Hive #3)

(c) A wasp prowling around the hive (Lyon Hive #3)

(d) An Asian hornet prowling around the hive (Cachan Hive #2)

Figure 4.15: Images taken by the *EAPBS*

data analysis. Indeed, the audio samples collected by our system have the potential to be used for classification models to build PB services. The images collected over the high season of 2022 also represent a rich dataset and have the potential of developing computer vision models around them. To develop such models in future works, there will be a few steps to follow: a first phase of labeling which would require some manual work to identify certain categories of audio samples and areas of interest in images, a second phase where data can potentially be preprocessed, and then classification model trained to classify audio sample or segment images. The main challenge for labeling data is the required apidology knowledge. On the one hand, images are straightforward as it is easy to observe intruders or pollen with the eye. On the other, audio, temperature

and humidity data can rarely be interpreted by just listening or looking at them. The best strategy for labeling such data is to ask a beekeeper to regularly write down the observations about a colony during visits, and then join these observations to the corresponding EAPBS's collected data.

In practice, we shared the results from Section 4.3 with professional beekeepers who validated the added value of this analysis. They particularly showed attention to the potential of sound and image analysis to later build diagnosis tools.

The next chapter aims at analyzing the energy consumption of our system and at introducing an energy-efficient deep learning-based queen bee detection service on top of our system. We will propose a simulation of IoT systems close to PB systems at a large scale in order to optimize the orchestration of services between the edge and the cloud.

Optimizing Resource Allocation for Service Orchestration at the Edge and in the Cloud

Contents for chapter 5

In this chapter, we aim to find the best orchestration of IoT systems' smart services between the edge and the cloud. To do so, we first define three important terms for this chapter in Section 5.1. Then in Section 5.2, an analysis of the energy consumption of our energy-hardware precision beekeeping system (EAPBS) is conducted to weigh the different steps of its routine. Section 5.3 introduces the two main scenarios considered and presents the work on the optimization of one precision beekeeping (PB) service in terms of performance and energy consumption. Section 5.4 shows the simulation results with different variations of parameters to provide insights about the best scenario. Finally, Section 5.5 shares the conclusion and future works of this chapter.

5.1 Definitions

The definition of the terms edge and cloud are first specified. The edge refers to the devices and sensors located in or near the physical environment where data is acquired (for example, a beehive). These devices are responsible for collecting and processing data processing before transmitting the data to the cloud for further analysis.

The cloud refers to the remote servers used for storing and processing data generated by the edge. Cloud infrastructure, compared to the edge, provides larger data storage capacity and computational power for analyzing data and running machine learning algorithms that can provide insights about the environment.

In this chapter, the goal is not only to optimize smart beehives in terms of their energy consumption but also to generalize results to all similar IoT systems. This group of edge systems, which is introduced here and will be referred to as *autonomous energy-harvesting stationary IoT systems* (AEHSS), can be described as follows:

- **Location:** The edge device lies immobile, often next to other devices, in an outdoor environment subject to temperature and weather variations. It is however often protected by a sealed box.
- **Hardware:** The edge device relies on a microcomputer able to handle dense amounts of data like audio samples and images. The handling part can be broken down into the collection, the processing and the transfer of data. Because of these constraints, microcontrollers in the category of the ESP32 lack the processing power and the required connectivity and, in the practical applications of this work, must be left out. Suitable microcomputers include solutions like Raspberry Pi which possess multicore processors, RAM, ports for sensors, Wi-Fi and Bluetooth connectivity, and the ability to use Unix-based operating systems.
- **Energy:** One system includes one microcomputer, several sensors and one energy node. The latter answers the important constraint of not being able to use a power outlet. Thus, a system must work autonomously under a limited energy budget which varies over time because of external parameters like the amount of sunshine and the weather. Using its energy node, it has to harvest its energy and store it.
- **Network:** On the communication side, systems must be able to carry close-range device-to-device communication and device-to-cloud communication. The services provided by an AEHSS can be in some cases supplemented by the help of a cloud server.

5.2 Energy consumption of the deployed system

In this section, the aim is to calibrate the frequency at which the Raspberry Pi 3b+ wakes up and performs a set of tasks at regular intervals.

The energy of the Raspberry Pi 3b+ is monitored at different wake-up frequencies: every 5, 10, 15, 30, 60, and 120 minutes (shown in Figure 5.1). For example, Figure 4.12b shows the frequency of 10 minutes, where two consecutive wake-up calls from the Raspberry Pi Zero are spaced by 10 minutes. The frequency parameter's values are realistic and could be implemented in the system, depending on the chosen beehive-related services. For example, in the case of a service tracking the temperature of the beehive, collecting data every 60 or 120 minutes suffices. On the other hand, in a period of collection of large datasets, collecting data every 5 minutes becomes reasonable. The experiment is designed so that each setting is active for at least nine hours in a row to collect substantial energy data. This duration is chosen because it corresponds to daylight time, where our energy node is more stable and able to provide constant power to the electronics. The collected energy is a sum of the active power (when the Raspberry Pi boots, performs tasks and shuts down) and the sleep power of the sleep

state, which consumes non-null resources: the cost of being able to receive wake-up calls.

The data collected during this experiment covers 319 routines of the system. All routines are identical. They follow the steps presented at the beginning of this section. It is then possible to compute average costs for one routine. On average, the Raspberry Pi 3b+ is turned on, performs its tasks, and shuts down in 1 minute and 29 seconds, with an average power of 2.14 watts. This gives an average energy cost of 190.1 joules from the boot to the shutdown. The standard deviation for the lengths of routines is 3.5 seconds. This relatively high variability can be explained by the variance of the duration of the data transfer correlated to the unstable network throughput. The standard deviation for the average power of routines is 0.009 watts, which shows stability. The average consumed power for on/off cycles at different wake-up frequencies is presented in Figure 5.1.

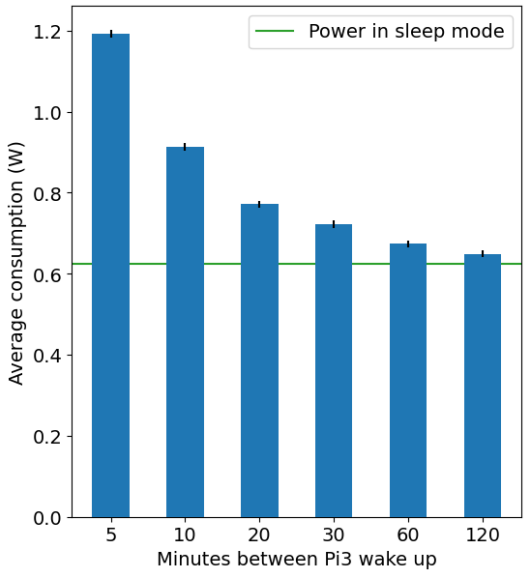


Figure 5.1: Average power consumed for several durations between two consecutive wake-ups of the smart beehive's system

At the highest frequency, when the system wakes up every 5 minutes, the average power values reach their maximum: 1.19 watts on average. When the duration between two consecutive wake-ups increases, the average power decreases and converges toward a value close to 0.62 watts, which is the consumption of the Raspberry Pi 3b+ in a sleep state. This shows that the sleep state is significant when considering long-term consumption.

The optimal wake-up frequency depends on the goal set by the user. Collecting a large amount of data implies a high frequency, whereas optimizing the energy costs to keep the system alive as long as possible implies a low frequency. The wake-up frequency also depends on the energy budget available which varies depending on the energy intake of the solar panel and the capacity of the battery. In this experiment, the energy intake of the battery from the solar panel is not a limiting factor. The

power capacity of the solar panel (30 W) shows that during the daytime, the system can operate at its maximum power (theoretically, with both Raspberry Pi always on and working). Using more limited energy production nodes would help to verify the presented measures and select the optimal wake-up frequency based on a lower energy intake.

5.3 Orchestration and energy optimization of one smart service

In this section, smart services are added on top of data collection and transfer systems. The main question is: which option is more energy efficient between performing the services' tasks at the edge or in the cloud? Two main scenarios are introduced:

- Edge scenario: the edge system wakes up, collects data, executes the service's tasks at the edge, optionally stores the data locally, sends results directly to the user (the beekeeper's phone, for example), and shuts down. This scenario does not require a cloud infrastructure.
- Edge+cloud scenario: the edge system wakes up, collects data, sends it to a remote cloud server, and shuts down. In parallel, a cloud server receives the data and performs the service's tasks. The unfolding of these steps can be found in Figure 5.2. In the previous section, we introduced a routine that collects and transfers the collected data to a remote cloud server, so the energy consumption data collected in that section will compute the costs of the edge device in this scenario.

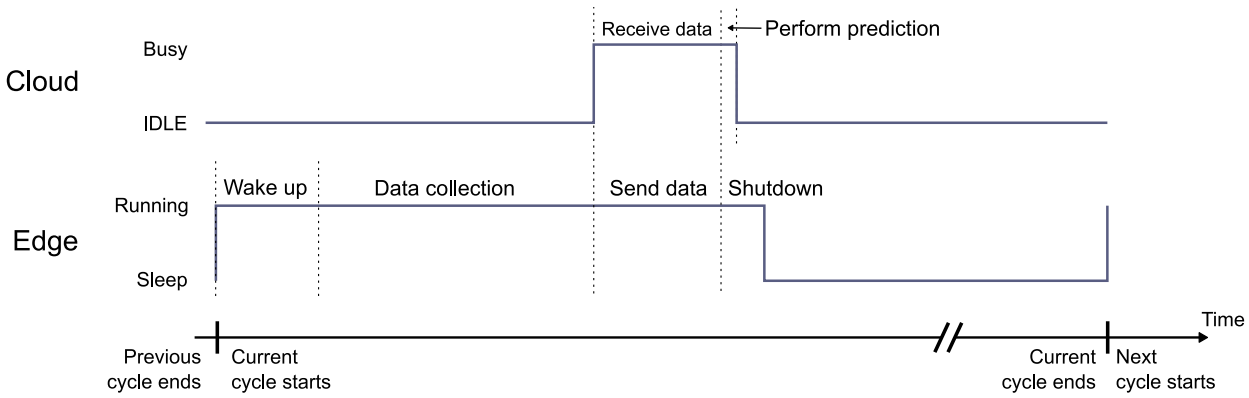


Figure 5.2: Task diagram for one cycle of data collection, processing, and transfer for edge+cloud scenario

Our selected cloud server has an Intel i7-8700K processor and an Nvidia RTX2070 GPU. This allows us to train the deep learning models considered in this chapter in a few minutes. Our selected edge device is the previously introduced Raspberry Pi 3b+, capable of performing numerous tasks: pollen detection, counting bees (image classification), and swarm prediction (audio classification or temperature analysis), among

others. Most of these tasks include AI models. In this section, we focus on the energy consumed by classification models for the presence of the queen inside a beehive, using sound data. In practice, in addition to the previously introduced data collection routine, an audio sample is used to predict the presence of the queen bee. For the *EAPBS*'s architecture, the prediction is performed with the sample from the microphone placed in the middle part of the queen excluder grid. The selected task of queen detection has already been studied with deep learning (convolutional neural network, CNN) and classical machine learning (support vector machine, SVM) methods [Nolasco and Benetos, 2018b]. The latter is a suited choice because it allows acceptable performances for reasonably small datasets (less than tens of thousands of samples). In this work, the same methodology is selected: selecting one classical machine learning option and one deep learning option.

For the training of both models, 1647 audio samples labeled with the presence of the queen are used. The selected training features are mel-scaled spectrogram features computed from 10-second audio recordings of bees sampled at 22 050 hertz, which allows capturing all the main frequencies emitted by bees, according to Section 3.2.1, all the while taking only 870 KB per sample. For the spectrogram function, the length of the fast-Fourier transform window is 2048, the number of audio samples between adjacent short-time Fourier transform columns is 512, and the number of generated Mel-bands is 128. Vector features are passed as it is for the training phase of the SVM model, whereas they are converted into images for the CNN model thanks to Librosa's `specshow` method[1]. The chosen architecture for the deep learning model is ResNet18 [He et al., 2015]. It is trained on the cloud server, through 4 epochs, with a learning rate of 0.001. Although pre-trained on regular real-world pictures, the model can achieve state-of-the-art performances when trained with spectrogram images. For the classical machine learning alternative, the SVM classifier is set with a radial basis function kernel, a regularization parameter of 20, and a kernel coefficient of 10^{-5}. The training phase of CNN models has a significant energy cost, but it is a less frequent task than the use of the trained models. Therefore, in this section, we only focus on the energy consumption of the usage of the trained models.

These models and parameters reproduced state-of-the-art performances for the queen detection task. Figure 5.3 shows the measured energy for the Raspberry Pi 3b+ to perform predictions and the accuracy of the trained models as functions of images' length. The 100 by 100 pixels resolution is the smallest that shows converged results, with a classification accuracy of 99%. Larger images as input of the CNN model show similar performances, but the energy cost when executing the model on the Raspberry Pi is greater than 100 by 100 pixels images. This cost increases as a quadratic function of the number of pixels in the images, which is the time complexity of the use of the trained deep learning model. Therefore, using 100 by 100 pixels images for the CNN model is the optimal choice when considering the accuracy and the energy cost of the model's execution at the edge.

Considering this focus on the energy consumption of one service, the upcoming paragraphs will generalize the results on the complete edge and edge+cloud scenarios

[1] https://librosa.org/doc/main/generated/librosa.display.specshow.html

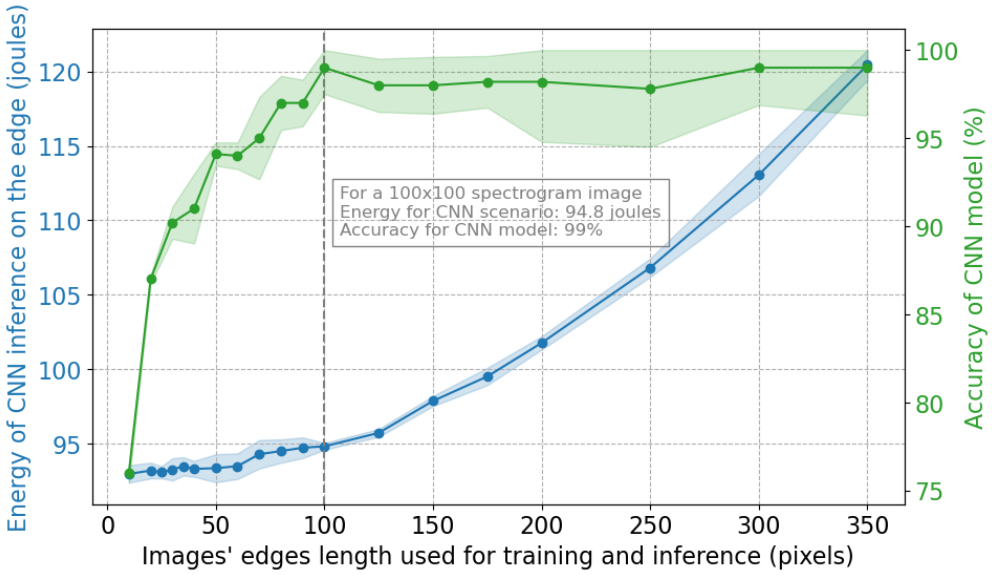
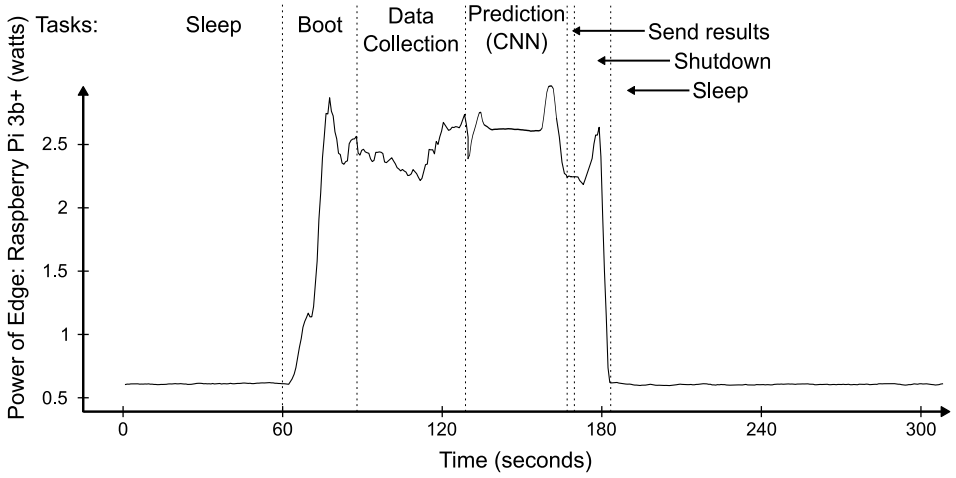


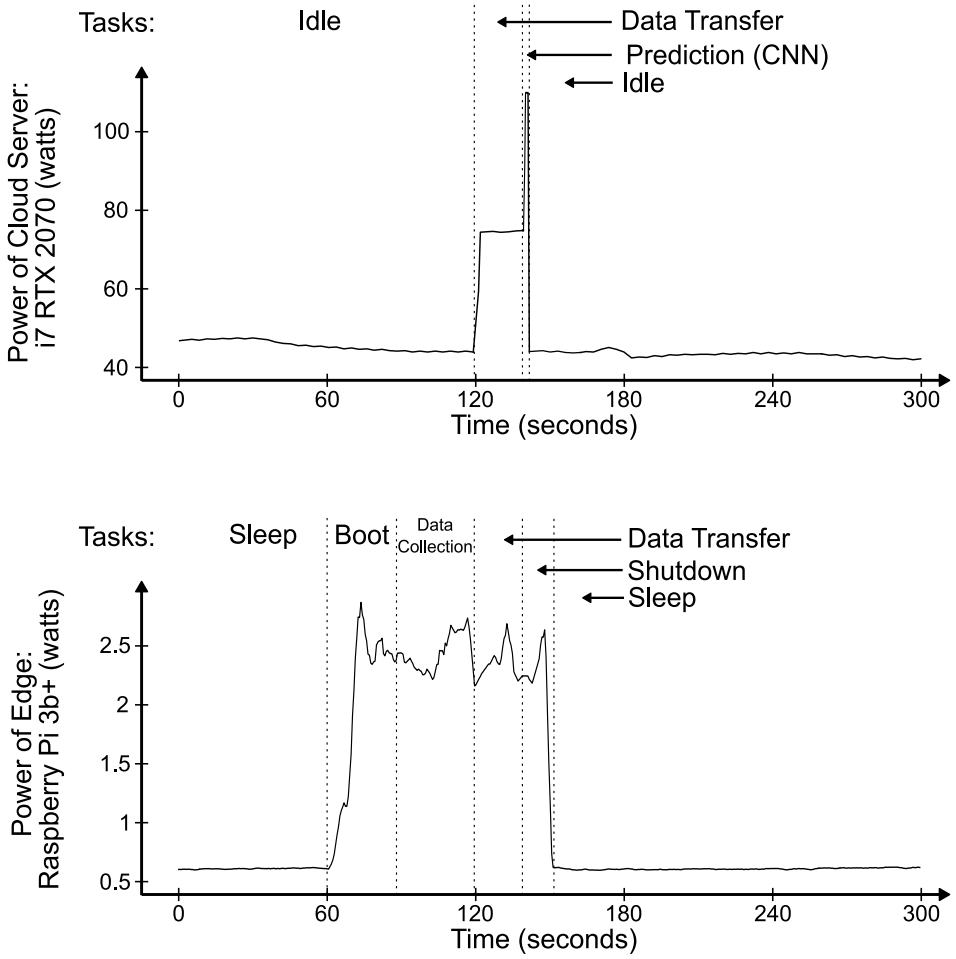
Figure 5.3: Energy and accuracy of queen classification CNN model inference on the test set as a function of training set's images' size.

that include this service. Figure 5.4 shows the unfolding of the edge+cloud scenario in real conditions. First, the Raspberry Pi 3b+ and the cloud server are in sleep and idle mode, respectively. The edge device is then turned on at the 60th second and starts collecting data. Then, the edge transfers the collected data to the cloud, and the edge starts shutting down as the server executes the service's tasks. Finally, the server switches to idle mode. The energy consumption graph of the Raspberry Pi during this routine shows a spike during the data transfer step. This step is more energy-intensive than the data collection because the network components have a larger energy cost than the sensors.

Each model (SVM and CNN) is then tested in the edge and edge+cloud scenarios. With the measures of the energy consumption of the edge and the cloud for the four options, we aim to find the best scenario for each service. Table 5.1 and Table 5.2 show the duration and the energy consumption for all cases when 5-minute cycles are considered. For each scenario, the collected time and energy consumption step by step, is shown in chronological order. Since the queen detection model's execution in the cloud takes less time than the shutdown of the Raspberry Pi 3b+, the shutdown is divided into two lines in the edge+cloud scenario, and their sum adds up to the 9.9 seconds shown in edge scenarios. The choice of the model does not generate a significant difference: only 1.2 joules of difference (0.3% of the edge's total cost) for the energy cost of the Raspberry Pi 3b+ in the edge scenarios and 61.7 joules (0.4% of the server's total cost) for the energy cost of the cloud server in the edge+cloud scenarios. In the edge+cloud scenarios, the main part of the service — the AI model — is not executed at the edge, so the energy cost of the edge is not influenced by the choice of the service. Then, the global energy cost of the IoT system is more significant for an edge+cloud scenario than an edge scenario because of the presence of a server that must be turned



(a) Edge scenario



(b) Edge+cloud scenario

Figure 5.4: Task and energy diagrams of both scenarios for one 5-minute routine of data collection, processing, and transfer

on and available at all times to receive data. However, this cost allows the edge to perform fewer tasks, hence a smaller energy cost for the edge+cloud scenario at the edge: there is a reduction of 12.1% and 12.4% of consumed energy for the SVM and CNN model, respectively. This can be useful in cases where the edge's energy must be saved. For example, when the energy available in the cloud is not a limiting factor and the energy produced by the solar panel at the edge is limited. A large-scale simulation with many smart beehives matched with one server could demonstrate the viability of the edge+cloud scenario.

Edge Task	Energy of Edge (joules)	Time (seconds)
Scenario: Edge (SVM)		
Sleep	111.6	178.5
Wake up & Data collection	131.8	64.0
Queen detection model (SVM)	98.9	46.1
Send results	3.0	1.5
Shutdown	21.0	9.9
Total	**366.3 joules**	**300 seconds**
Scenario: Edge (CNN)		
Sleep	116.9	187.0
Wake up & Data collection	131.8	64.0
Queen detection model (CNN)	94.8	37.6
Send results	3.0	1.5
Shutdown	21.0	9.9
Total	**367.5 joules**	**300 seconds**

Table 5.1: Tasks of edge with time and energy consumed per 5-minute cycle for queen detection edge scenarios

5.4 Simulation at large scale

Currently, five smart beehives are deployed and equipped, but according to the recent beekeeping numbers in France from [ADA-France, 2022], the majority of beehives belong to professional beekeepers who own more than 150 beehives. These beekeepers would ideally want to monitor hundreds of hives and would need to embed such a system at a larger scale, in a whole apiary for example. Also, an organization of several beekeepers could put their hardware in one unique infrastructure of edge and cloud resources. This section aims at putting the currently deployed systems and their cloud servers on a large scale. A simulation method is proposed to determine whether a large infrastructure of smart beehives is energetically viable.

Edge Task	Energy of Edge (joules)	Cloud Server Task	Energy of Cloud Server (joules)	Time (seconds)
		Scenario: Edge+Cloud (SVM)		
Sleep	131.9	Idle	9415	211.1
Wake up & Data collection	131.8	Idle	2854	64.0
Send audio	37.3	Receive audio	1032	15.0
Shutdown	0.2	Queen detection model (SVM)	6.3	0.1
Shutdown	20.8	Idle	437	9.8
Total	**322.0 joules**		**13744.3 joules**	**300 seconds**
		Scenario: Edge+Cloud (CNN)		
Sleep	131.9	Idle	9415	211.1
Wake up & Data collection	131.8	Idle	2854	64.0
Send audio	37.3	Receive audio	1032	15.0
Shutdown	2.1	Queen detection model (CNN)	108	1.0
Shutdown	18.9	Idle	397	8.9
Total	**322.0 joules**		**13806 joules**	**300 seconds**

Table 5.2: Tasks of edge and cloud with time and energy consumed per 5-minute cycle for queen detection edge+cloud scenarios

5.4.1 Description of the simulation model

The simulation model has three main components: the client, the server, and the allocator. In the practical case of our *EAPBS*, one client refers to one smart beehive and one server refers to one cloud server. However, this method can encapsulate any AEHSS linked to any server.

- Client: its tasks are to acquire and optionally process and transfer data. It is initialized with the power consumption in the sleep state, a series of tasks (active state) and their respective time and power consumption, and the time between two consecutive wake-ups.
- Server: its tasks are to receive data from clients and process them. It also has a series of tasks, time, and power consumption. It supports a maximum number of clients allowed in parallel. Each server allows its clients to start communication at specific times so that every client within a group has to start their communication with the server at the same time as the other clients of the group, all synchronized in time thanks to specific hardware (GPS, for example). These specific time windows will be referred to as time slots. One server can allow multiple time slots. The shorter the time window for the server's tasks, the greater the number of time slots. For example, in a 5-minute cycle, given a data transfer and a model execution's duration of 1 minute, a server can allow 5-time slots.
- Allocator: its main task is to allocate several clients to servers. It takes a list of clients, creates servers based on their features and the features of the server type, allocates every client to one server, and links them to a wake-up time slot.

Currently, it has one filling policy: filling a server with clients by filling one slot up to its maximum after another.

The simulation will be performed in theoretically perfect situations as well as realistic cases involving losses. The results will be described in the following paragraphs.

5.4.2 Results in an ideal theoretical situation

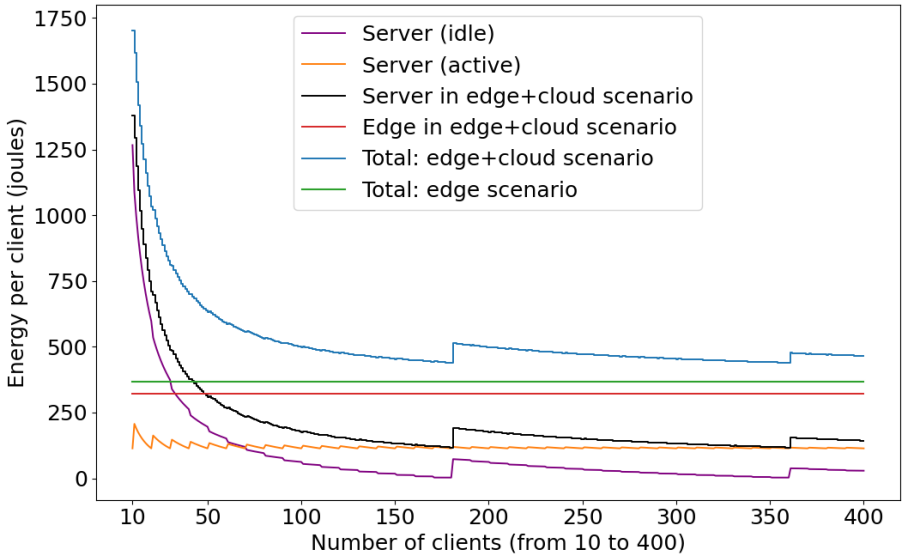
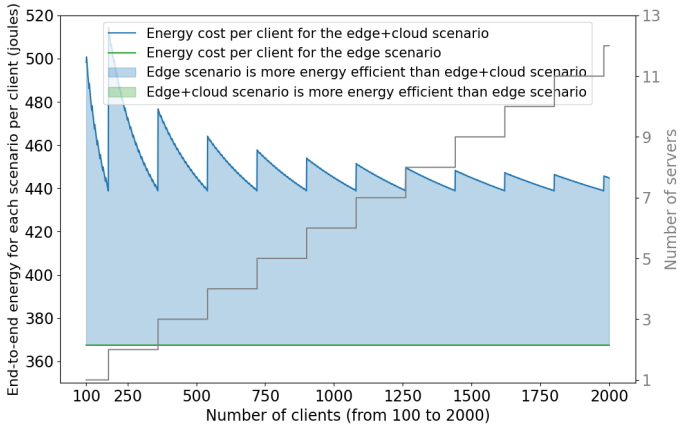


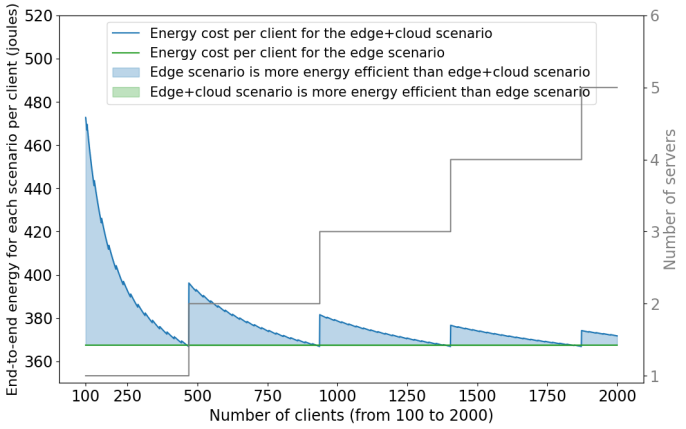
Figure 5.5: Simulation of energy consumption for all entities in a client-server scenario with no loss. Number of clients allowed in parallel in time slots: 10.

Figure 5.5 shows the simulation results for the number of servers required to handle clients, and the energy consumed per client and in total, respectively. This simulation recreates the previously introduced edge+cloud scenario and edge scenario for an infrastructure of clients and servers with clients ranging from 10 to 400. The tasks' duration and power are initialized with the measures described in Section 5.2 and Section 5.3, considering that one client is a smart beehive. The simulation also supposes a client's wake-up frequency of 5 minutes and a server that can allow a maximum of 10 clients per time slot. Since the cost per client increases exponentially when considering less than 10 clients, the lowest number of clients is set at this value for readability reasons. Figure 5.5 shows that the energy consumption per client of edge devices (shown in red) is equal to one client's energy consumption, 322 joules. It is because the tasks performed at one edge do not depend on the server's parameters or the number of edge devices in the infrastructure. The server's overall energy consumption per client (shown in black) converges toward 116 joules, the energy for which all the server's time slots are full. The smart beehive case brings the overall best cost per beehive (shown in blue) to $116 + 322 = 438$ joules. This can be described as the theoretical best cost per client. Also, it is 16% greater than the overall cost in the edge scenario. However, it is

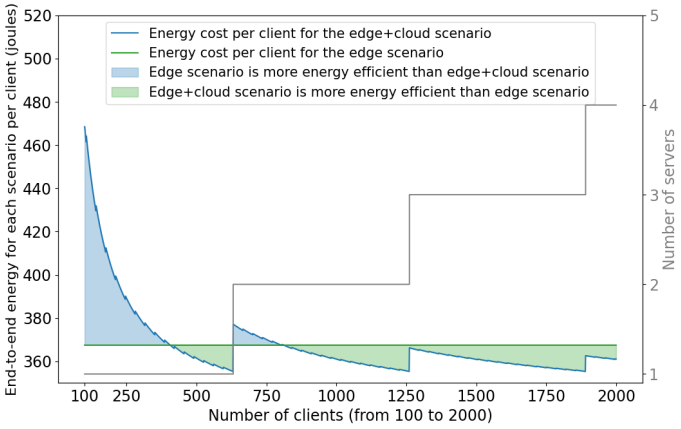
fair to compare the worth of the energy produced at the edge by the solar panel and the energy available in the cloud. It is more likely that cloud servers are powered by nationally produced electricity, so this part is not (or at least less) under pressure. On the other hand, the energy produced at the edge by the solar panel and stored inside the battery is less reliable, and the amount of energy available is much lower than in the cloud. Therefore, one joule of energy used at the edge is not equivalent to one joule of energy used on the cloud in terms of availability and resilience. One could argue that the 16% increase of energy in the edge+cloud scenario does not make this scenario less efficient than the edge scenario.



(a) Number of clients allowed in parallel in time slots: 10.



(b) Number of clients allowed in parallel in time slots: 26.



(c) Number of clients allowed in parallel in time slots: 35.

Figure 5.6: Comparison of end-to-end energy per client for the two scenarios with different server capacities

This difference between the two scenarios decreases as the number of clients allowed in parallel per time slot increases. Figure 5.6 focuses on the simulation data for the end-to-end total energy for both scenarios between 100 and 2000 clients and highlights the differences when the maximum clients allowed in parallel changes. This range for

the number of clients is different from the simulation results shown previously because increasing the number of clients allowed in parallel increases the total number of clients allowed per server. Figure 5.6a shows the same setting as Figure 5.5: 10 clients at maximum in parallel. Figure 5.6b and Figure 5.6c respectively show the simulation results when this parameter takes 26 and 35 as values. In those three figures, the blue area corresponds to settings where the average cost of one client is lower in the edge scenario than in the edge+cloud scenario. The green area corresponds to parameters making the edge+cloud scenario more energy-efficient. 26 clients are the tipping point when the edge+cloud scenario can become more energy efficient when used efficiently. Figure 5.6c shows that when 35 clients are allowed in parallel, 406 clients are needed to make the edge+cloud scenario more energy-efficient than the edge scenario. With these settings, the maximum difference in favor of the edge+cloud scenario is 12.5 joules at 630 clients, just before the need for an additional server when the first server is fully used. This graph also shows that from 803 clients, the edge+cloud scenario is more energy-efficient than the edge scenario and remains this way for any higher number of clients considered. This is explained by the filling policy introduced above. All numerical values for the results listed in this section correspond to our PB hardware and chosen parameters. These could be replicated and adapted to any client-server infrastructure if different parameters are given as prerequisites.

5.4.3 Results with losses for clients and cloud servers

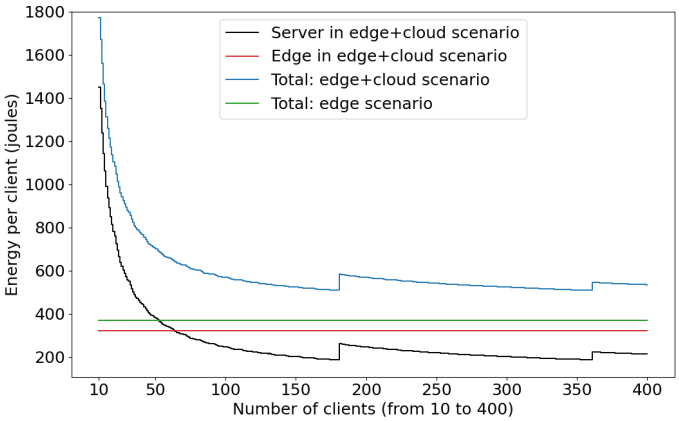
In this paragraph, the following parameters are added to depict real-life losses and costs:

- Loss scenario A: A penalty when a server's time slot starts saturating with its number of clients. The limit at which the penalty starts is set at 5 clients below the maximum allowed per slot. Each additional client penalizes the whole energy by 10%.

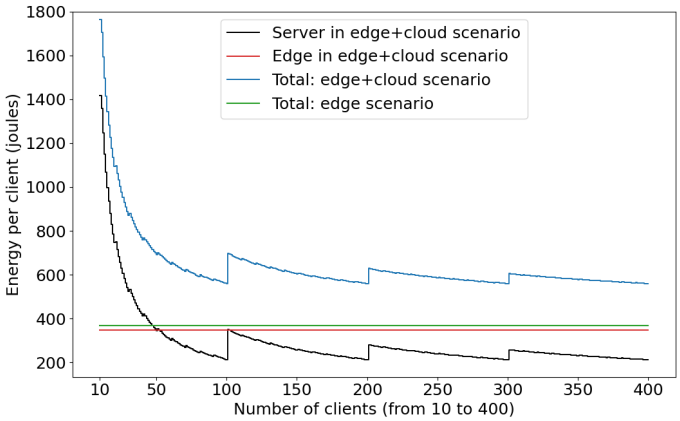
- Loss scenario B: A time penalty of 1.5 extra seconds per client for clients' data transfer time. Since the model considers that clients of the one-time slot are synchronized, they send their data simultaneously.

- Loss scenario C: A loss of clients at every wake-up time. A random Gaussian distribution (mean: 10% of the total number of clients; standard deviation: 2) is used to draw the number of lost clients.

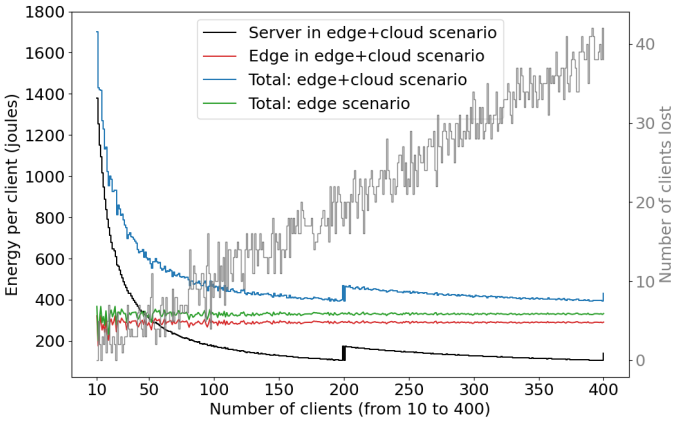
The above-mentioned values for the loss parameters are chosen thanks to the understanding gained during the data collection period. Figure 5.7 shows the simulation results of the energy consumed per client when losses are added to the models. Figure 5.7a shows that the first considered loss increases the energy consumed by the server. In this setting, the cost of the server converges toward 186 joules (to be compared with the 116 joules, which represent the lowest server cost for the no-loss model). Figure 5.7b highlights that extra transfer duration causes the number of time slots per 5 minutes to decrease, thus decreasing the maximum number of clients per server. More servers are needed for an equivalent number of clients compared to the no-loss scenario (for



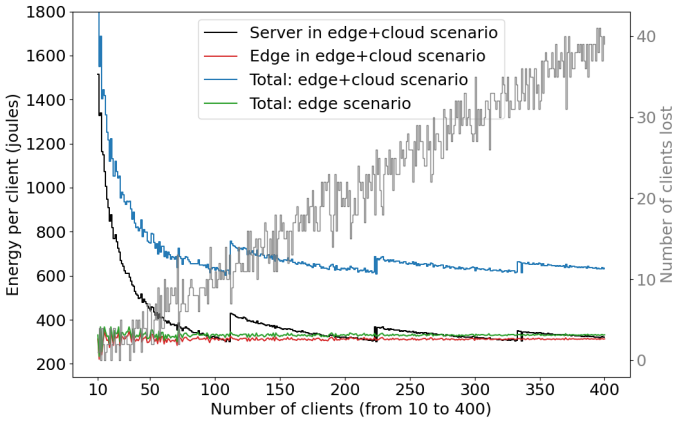
(a) Loss scenario A



(b) Loss scenario B



(c) Loss scenario C



(d) All three loss scenarios combined.

Figure 5.7: Simulation of energy consumption in a client-server scenario with several types of loss.
The number of clients allowed in parallel in time slots: 10.

350 clients: 4 servers when duration penalty is applied versus 2 servers in the no-loss case). Since there are fewer clients per server, the cost of servers per client increases. In this case, its minimal value is 212 joules. In Figure 5.7c, the random Gaussian number of clients lost alongside the energy consumption metrics are displayed. The x-axis displays the initial number of clients, so the energy consumed appears to be less than the default case without any loss. However, this is a case where lost clients cause other losses: all the services and the data would be missing. Figure 5.7d combines the three types of loss. On this figure, there are some abnormal rises around 225 clients and 340 clients. Those spikes are due to the varying number of randomly lost clients making the required number of servers decrease, although the initial number of clients increases.

Finally, Figure 5.8 shows the comparison of the two scenarios with all three types of loss considered and displays more realistic simulated energy consumption values. This set of parameters shows that the limit of clients per time slot of 35 becomes worse than

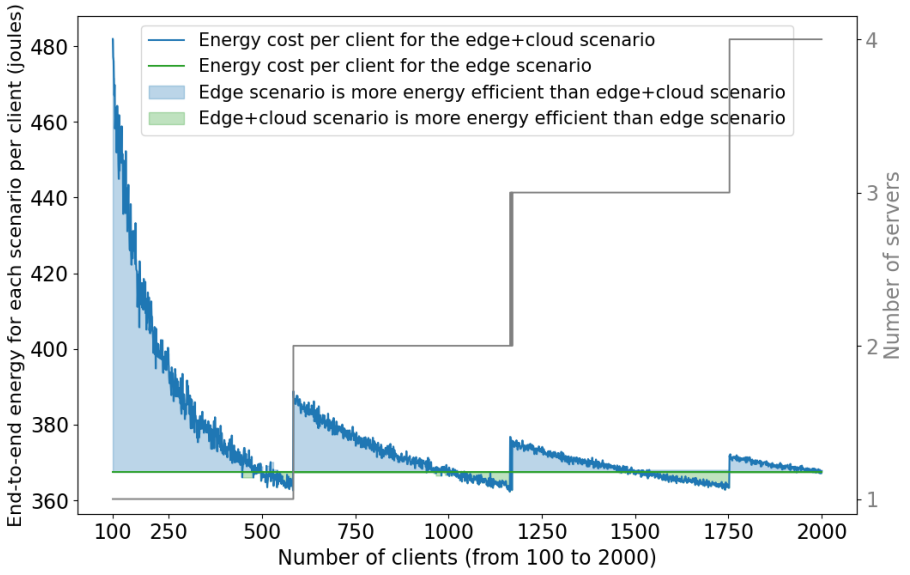


Figure 5.8: Comparison of end-to-end energy per client for the two scenarios with different server settings and with loss. Number of clients allowed in parallel in time slots: 35.

its equivalent without loss but still has some intervals where the edge+cloud scenario is more energy-efficient than the edge scenario. With this simulation, it is possible to scale a set of clients and servers, given the costs and penalties of energy consumption or time, which depend on the number of clients. For example, it is safe to assign three servers when the number of clients is between 1600 and 1750, and the edge+cloud scenario will be more energy-efficient than the edge scenario. On the other hand, for a number of clients corresponding to the blue areas, the best scenario becomes the edge scenario.

5.5 Conclusion of the orchestration of edge/cloud services

In this chapter, we presented a method for the orchestration of services in IoT systems at the edge and in the cloud with the use of direct continuous measures and large-scale simulation. We define a subcategory of IoT systems, which includes systems like connected beehives, as autonomous energy-harvesting stationary IoT systems (AEHSS).

We then measure the energy consumption of our energy-hardware precision beekeeping system's (EAPBS) data collection routine: 190.1 joules (2.14 watts on average for 1 minute and 29 seconds). The overall energy cost of the EAPBS is also affected by the energy consumed during the moments when the Raspberry Pi 3 is sleeping. The average overall consumption increases as the frequency of the system's wake-ups decreases: 1.19 watts are required for a wake-up frequency of 5 minutes, while frequencies of 60 and 120 minutes approach the sleep power of 0.62 watts.

The question of service orchestration is introduced with two scenarios: one only

keeping the EAPBS — the edge device — and one delegating a deep learning model's computation to a cloud server. The deep learning model is a convolutional neural network (CNN) that classifies bees' audio samples between two categories: whether the queen is present inside the hive. We find the tipping point at which the inference of the CNN model provides the best classification accuracy while consuming little energy: for 100×100 pixels images representation of spectrograms, one inference at the edge costs on average 94.8 joules. We also measure the costs of each task for the edge in the first scenario and for the edge and the cloud in the second scenario, and we also compare the CNN to an SVM-based alternative. Then, a client-server energy simulation model is introduced. The energy consumption parameters are instantiated with the energy consumption measures of our EAPBS. Simulations are run with a varying number of clients (edge devices) and different server configurations (their capacity). On top of this, we add losses inspired by the real-life deployment of the EAPBS: the saturation of the cloud server through a longer time for the computationally-heavy task, the saturation of the network through a decreased throughput, and the random loss of clients which do not wake up due to an unknown error. Adding cloud servers can become more energy-efficient than the edge-only counterpart when cloud servers are used close to their maximal capacity. In the case of our EAPBS, with our measured initial values, we find the sections where it is more energy-efficient to choose the scenario that uses a cloud server: for servers allowing 35 clients in parallel, one of these sections is between 1600 and 1750, and three servers would be needed. In practice, this is an achievable setup if some of the largest apiaries are equipped.

Future work will refine the simulation model, add new scenarios, introduce new allocation policies and refine the numerical estimations of losses. The next chapter is the realization of one of this chapter's future work: the use of machine learning and deep learning models to improve the simulation model and build connected beehives' intelligence to tune its parameters and choose between a set of scenarios.

Task Allocation in IoT Systems using Reinforcement Learning

Contents for chapter 6

In the previous chapter, the energy costs of a recurring routine performed in an autonomous energy-harvesting stationary IoT system (AEHSS) have been analyzed. It is key to understand the critical points to optimize a routine. However, AEHSS can have more scenarios in the shape of different tasks to select at every wake-up. For example, precision beekeeping (PB) systems can perform a wide range of tasks: queen detection, pollen detection, counting bees, swarm detection or prediction, pest detection and honey production estimation being the most common ones found in the literature (Chapter 3). Those services all use artificial intelligence (AI) models and are non-negligible in terms of energy consumption when compared to the other tasks of a PB system.

In this chapter, Section 6.1 first reintroduces reinforcement learning and the differences with its machine learning alternatives. We then propose a reinforcement learning-based solution capable of maximizing the long-term value of AEHSS services. The following assumption is made: an AEHSS has several tasks available and must choose a

set of them at every routine. To build an efficient strategy of task selection, we implement an environment that mimics real-life time and weather dynamics (Section 6.2) and use reinforcement learning AI models paired with photovoltaic energy prediction models. In Section 6.3, we describe the four reinforcement learning methods, how we implement their training phase, and we introduce two heuristics strategies that serve as a comparison for the performance tests that are analyzed in Section 6.4. Finally, Section 6.5 shared the summary and the concluding results of this chapter as well as its future works.

6.1 Reinforcement learning

In this section, reinforcement learning (RL) will be introduced alongside its machine learning counterparts.

Machine learning (ML) regroups all methods which teach machines how to solve problems thanks to examples and without any explicit logic. To build an ML algorithm, there are steps to follow. First, models have to be trained on data to learn patterns. Then, performances are tested whether a test set or new data is involved. There are three main types of ML algorithms: supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning Supervised learning gathers techniques that take labeled data as input. The label is the intended outcome, often called the target variable. For instance, in an image classification problem solved with supervised learning, the pair (image, label) can be a form of data for the learning phase. The trained algorithm should then be able to associate the most probable label to input images.

Algorithms like SVM, linear regression, logistic regression, decision trees and neural networks can be used in supervised learning problems. The advantage of supervised learning is its ease of understandability. However, if the selected method has a low convergence rate, the amount of data needed to achieve acceptable performances can sometimes be excessive. The time taken for the labeling phase, which sometimes has to be done manually, could be the limiting factor.

Unsupervised learning Unsupervised learning is suited for training models on data that is not labeled. Such algorithms take care of finding the right patterns on their own. It is close to the human way of learning: predicting the class of observation by finding similarities between observations of data. Such methods can be used on retail websites for example, to group items and build recommendation systems.

PCA, k-means or mixture models are some examples of unsupervised learning methods. The advantage of unsupervised learning is that it does not require manual intervention for labeling. However, as the algorithm finds patterns on its own, the results may be hard to explain.

Definition of reinforcement learning Reinforcement learning is a category of ML methods developed in the 1990s and popularized by [Sutton and Barto, 1998]. In

RL, an agent uses trial and error to explore and gain knowledge about the dynamics of an environment and maximize its reward. The environment with which the agent interacts needs one action as input and outputs an updated state and a reward to the agent (Figure 6.1). The agent chooses actions with its policy.

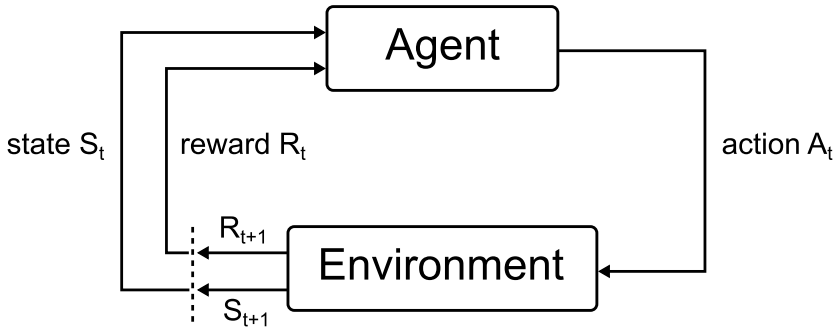


Figure 6.1: Base schema of reinforcement learning.

In contrast to supervised learning where the model is trained on labeled data to predict an output and to unsupervised learning where the model learns patterns and structures from unlabeled data with no explicit feedback signals, reinforcement learning works in an interactive setting where the agent receives rewards based on its actions. RL does not require labeled data and can learn from raw sensory inputs, making it suitable for tasks where it is hard to obtain labeled data, or the environment is continuously changing.

The training phase is structured as follows: at first, the agent does not know the optimal strategy, so it explores to find some good behaviors. It is called the exploration phase. The best way to explore is to behave randomly. ε is the exploration rate — the probability of choosing a random action, rather than the best possible one. During the exploration phase, ε tends to 1. After it is trained, the RL agent uses its knowledge and focuses on the best possible actions. It is called the exploitation phase, where ε tends to 0. RL is better than supervised learning at solving problems that generate a high number of states and actions (more than a human could handle).

RL training methods can be divided into three categories:

- Value-based methods, where the agent tries to optimize a value function. The value function maps an action or a pair (action, state) to an expectation of the future return.
- Policy-based methods, where the agent directly tries to optimize its policy, which maps states to actions.
- Actor-critic methods, where the agent learns both value function and policy.

6.2 Environment description

In the case of RL applied to resources placement in autonomous energy-harvesting stationary IoT systems (AEHSS), the agent is a task allocator which must maximize

the added value of tasks performed by the system (Figure 6.2).

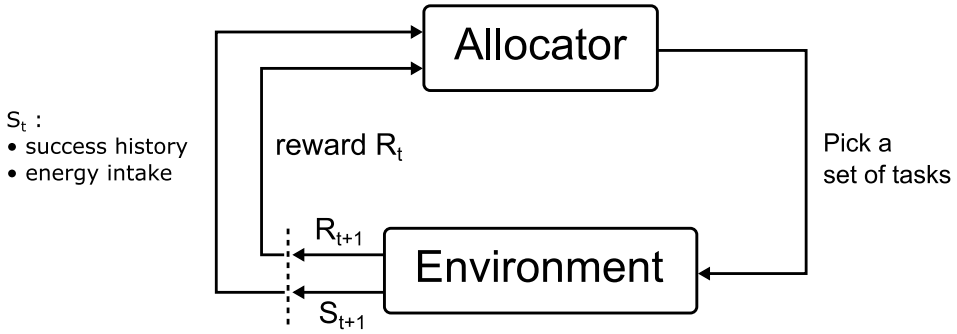


Figure 6.2: Schema of reinforcement learning applied to task optimization of AEHSS.

6.2.1 Actions

The agent must select a set of tasks to be performed within an initial list of tasks $\{T_1, T_2, ...\}$. Every task T_i has an initial value V_i (from the end user's perspective) and an energy consumption cost E_i. In our case, there are six tasks to choose from: queen detection, pollen detection, counting bees, swarm detection, pest detection and honey production estimation. These tasks are considered because they represent a balanced mix of diagnoses for a versatile beekeeping service (Chapter 3). The selected list by the RL agent can be empty, i.e., zero tasks can be performed. There are 64 possible subsets of tasks because the power set of a 6-item list, which represents all possible subsets, has a cardinality of 2^6, i.e. 64. Tasks are supposed to be independent and their order of execution does not impact their cost or value. Also, the cost of a set of tasks is equal to the sum of the costs of each task. For each iteration, the energy cost of the whole system routine is equal to the sum of the costs of each task added to a base cost E_{base} that represents the energy consumption of the on and off transitions, the acquisition of data and the transfer of results. These steps are mandatory for the whole execution of one routine. We share the values used in our implementation for the values and costs of the precision beekeeping tasks of our implementation in Table 6.1. The cost of the queen detection model and the base cost are taken from the measures shown in Table 5.1. The other costs are estimated thanks to the type of data that they handle and the methods that they use. For example, counting bees requires computer vision methods that are the most energy-consuming, as opposed to swarm detection which requires an audio analysis — usually less computationally heavy than computer vision methods — or honey production estimation which only requires single-point data like temperature and humidity (as seen in Figure 3.2).

6.2.2 Observations (States)

At the initialization of the environment and after every action, the environment generates an observable state made of two variables:

Task name	Cost (joules)	Value (relative terms)
Queen detection	94.8	60
Pollen detection	77	50
Counting bees	87	55
Swarm detection	60	40
Pest detection	70	45
Honey production estimation	20	10
Base cost (always mandatory)	272.7	-

Table 6.1: Costs and values of the six considered precision beekeeping tasks.

- The past outcomes $[S_{t-1}, S_{t-2}, ..., S_{t-n}]$: whether the system was correctly executed at the n last iterations, t referring to the current iteration;
- The energy output of the solar panel to the battery during the latest time interval: E_t. This value is estimated thanks to the model that will be described in Section 6.2.5.

The past outcomes are a list of size n. In our case, we set the value of n to 5. Together with our wake-up frequency of 5 minutes, it means that the agent can observe its history up to 25 minutes into the past. This middle-term memory is intended for the agent to estimate how well the system performed during the recent past iterations. A value of the list can take three different values that describe the state of the energy budget available. The first value (0) is used when the current action cannot be performed because its energy cost is larger than the current energy budget B. Here, the energy budget becomes exactly 0. The second (1) describes a critical low-battery mode when the currently available budget falls between a threshold value B_{critic} and strictly over 0. The third value (2) refers to a healthy energy budget: the resulting budget after the processing of the tasks is greater than B_{critic}.

In our case, the threshold values B_{critic} is set to 25% of the maximal battery level. This value is selected because a Li-Polymer battery, which is used in our system, has a flat discharge curve. Therefore, the voltage remains relatively constant until the battery's charge level drops to around 20%, from which we had 5 points of percentage for safety. Discharging a Li-Polymer battery below the recommended threshold results in decreased battery life and potential damage.

6.2.3 Rewards

There are four main categories of rewards given to the agent after every action performed:

- A negative maximal penalty P_{max} if tasks cannot be performed because the energy budget is empty: $B = 0$ after the execution of the desired action (routine fails: in Algorithm 1, which represents one update of the environment, it corresponds to line 2).

- 0 if tasks cannot be performed because of random fail (routine fails: Algorithm 1, lines 9 and 13).

- A negative penalty if the budget falls below B_{critic} but stays over 0 (routine succeeds: Algorithm 1, line 16). This penalty's absolute value is inversely proportional to the remaining budget and the multiplicative constant (0.05) is chosen so that this penalty linearly increases until the maximal penalty as the budget falls to 0.

- The sum of the positive values of all tasks if all tasks can be performed and the energy budget stays over B_{critic} (routine succeeds: Algorithm 1, lines 8 and 12). Each task's value is affected by the recentness of the last task's occurrence. The longer since the last occurrence, the larger the multiplier. This reflects the need to regularly be aware of all possible metrics and to want to not forget about any of them for too long. Every additional iteration since the last successful execution of a task multiplies the value of a task V_i by m. The increase is capped after it reaches c iterations so that the multiplier is capped at m^c. In our case, based on our actual data and experience, m and c are respectively set to 1.15 and 10. This also means that the maximal boosted value of a task is capped at $1.15^{10} = 4.05$ times the initial task's value. It is worth noting that the values of all the introduced parameters that we use are subject to change in a different implementation based on the needs of the end user. For example, if a beekeeper does not necessarily want the smart beehive to circle through all the available tasks, m could be lower than 1.15.

6.2.4 Environment rules

The task allocation process is repeated every time a system wakes up. The duration between which two consecutive iterations is not shown to the agent. This parameter is linked to the amount of energy gained by the system between iterations, thanks to the charge of the battery. In our case, the wake-up cycle is periodic, and its frequency is set to 5 minutes, which is a frequency that can make the energy budget tend toward 0 if too many energy-heavy tasks are selected. Therefore, a greedy strategy that selects all tasks at all times would pull the energy budget to 0 in less than a day and generate negative rewards.

At every step, the current weather category is given by the data from the OpenWeather API's current weather data endpoint[1]. This endpoint returns the qualitative type of weather among the following categories: clear, clouds, rain, mist, drizzle, fog, haze, snow and thunderstorm.

Algorithm 1 represents one update of the environment. It takes as a parameter the action (a list of tasks) selected by the agent and returns a reward. The energy budget lies between 0% and 100% of the battery capacity. In our case, it corresponds to 0 and 266 400 J — the energy that can be stored in a 20 000 mAh / 3.7 V power bank battery.

[1] https://openweathermap.org/current

If the sum of the tasks' energies is greater than the energy budget, the routine fails, the latest outcome (0 in this case) is pushed into the history and the maximal penalty is given as a reward (line 2). If the sum of the tasks' energies is lower than the energy budget (line 6), the routine succeeds 95% of the time if the weather category is "clear" (line 8) and the latest outcome pushed into the history is 2. For other weather categories (line 11), the completion rate is 90%. For the other 5% and 10%, the routine fails because of unknown reasons: the reward and the latest outcome are both equal to 0. This behavior mimics the deployed precision beekeeping system introduced in Chapter 4. When the routine succeeds, the energy budget decreases by the sum of the base cost (wake-up, data collection, transfer and shutdown) and the sum of the energies of the selected tasks (line 1). If the new energy budget B falls below B_{critic} (line 15), a negative penalty, smaller in absolute value than P_{max}, is returned and the value of 1 is pushed into the history of outcomes.

Algorithm 1: Environment algorithm for processing one action

Input : Action $\{T_a, T_b, ...\}$
Output : Reward r
Variables: Set of tasks: $\{T_1, T_2, ...\}$, their values V_i, their cost (energy) E_i
Number of steps since their last successful occurrence O_i
Task's value multiplier m, Multiplication of iterations' limit c
Energy budget B, Critical budget B_{critic}, Weather W
Past outcomes $[S_{t-1}, S_{t-2}, ..., S_{t-n}]$, Current outcome S_t
The base cost of the system E_{base}, Maximal penalty P_{max}

1 $B = max(0, B - (E_{base} + \sum_i E_i))$
2 **if** $B = 0$ **then**
3 $r = -P_{max}$
4 $S_t = 0$
5 **end**
6 **else**
7 **if** W *is "clear"* **then**
8 95% of the time: $S_t = 2$; $r = \sum_i V_i \times m^{min(c, O_i - 1)}$
9 5% of the time: $S_t = 0$; $r = 0$
10 **end**
11 **else**
12 90% of the time: $S_t = 2$; $r = \sum_i V_i \times m^{min(c, O_i - 1)}$
13 10% of the time: $S_t = 0$; $r = 0$
14 **end**
15 **if** $B < B_{critic}$ **then**
16 $r = r \times (B - B_{critic}) \times 0.05$
17 $S_t = 1$
18 **end**
19 **end**
20 return r

The environment uses the weather from the year 2022 to simulate its weather component. Between iterations (i.e., between two actions chosen by the agent and processed by the environment), the environment updates its date and time, the past outcomes and the energy budget. The first two are straightforward updates as it only depends on the routine frequency and the current outcome, respectively.

6.2.5 Prediction of solar power intake

For the update of the energy budget, the case of photovoltaic energy is considered, and a prediction model inspired by [Kraemer et al., 2020] is built. This cited article studies the different machine learning approaches for the prediction of solar power intake. The objective is to predict the solar power intake at one time interval using meteorological data and the position of the sun.

Sources of the predictor variables

The weather data used as part of the predictor variables are from the Météo-France[2] station of Lyon-Saint Exupéry airport (latitude: 45.7265; longitude: 5.077833), which is close to our deployed energy-hardware precision beekeeping system (EAPBS) deployed in Lyon. Initially, the database contains 82 parameters gathered every 3 hours. For 71 of them, the data either do not correspond to weather descriptors (for instance: location constants, time data) or contain a majority of unknown values. Only 11 of them are selected to be part of the models' predictor variables: temperature, humidity, air pressure, precipitations of the last 3 hours, horizontal visibility, nebulosity (global) and the nebulosities of 5 different cloud layers. In addition to these 11 variables, the zenith, the azimuth and the number of seconds elapsed in the current day are also part of the predictors. Zenith and azimuth angles are obtained thanks to the PyEphem Python package[3].

Experimental system for the collection of the target variable

The energy intake of the solar panel is the target variable — the metric that the models must predict.

The system used to collect photovoltaic energy is slightly different from the connected beehive's energy node introduced in Section 4.2 because, for the latter, the solar panel outputs its energy into a battery, which can become full. In that case, the battery prevents any more current to flow through the electrical circuit, and therefore, the collected energy data does not correspond to the potential energy created by the solar panel. Using data from the system deployed in the 2022 season would introduce inconsistencies. Inconsistencies can be removed if the current flowing in the circuit always corresponds to the photovoltaic production of the solar panel. To do so, an appropriate resistor is directly wired to the solar panel, in place of the battery. This solar production

[2] https://donneespubliques.meteofrance.fr/?fond=produit&id_produit=90&id_rubrique=32

[3] https://rhodesmill.org/pyephem/

collection system is deployed and collects data continuously since early March 2023. It is oriented toward the east. The data is recorded with the current sensor introduced in Section 4.1.2. Figure 6.3 shows the data collected over the 12th and the 13th of April 2023 days. Data points are sampled at 3.6 Hz and each represents the average current of 250 values measured. The two represented days have different weather conditions: the first is a fully rainy day and the second is a partially cloudy day, hence the higher energy production on the 13th.

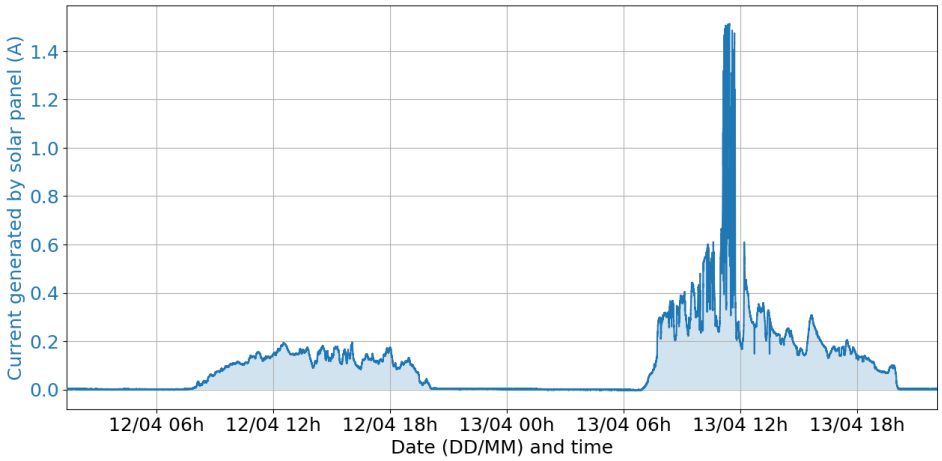


Figure 6.3: Photovoltaic energy production of east-oriented solar panel

Although we collect energy production data at a rate of 3.6 Hz, the predictor variables — the weather data — are sampled in chunks of 3 hours. Therefore, for compatibility purposes during the training phase between the two sets of variables, we resample the energy production data in chunks of 3 hours. For each date and time from the weather data, the corresponding value of the photovoltaic production is equal to the average over 3 hours putting this date and time at the middle of the 3-hour window. Our prediction model will then aim at predicting this 3-hour average value using the weather features. Once in production, the weather data for a given date and time is assumed to approach best the weather of the 3-hour interval that puts the given moment in the middle. In other words, for a given time, the goal is to predict the total amount of energy created from the previous hour and a half to the next hour and a half.

Prediction models

Before training, the data is first centered and scaled by removing the mean and dividing it by the standard deviation. Then, the train-test split is set at 80%-20%, keeping both splits continuous — i.e., the test dataset is either all before or all after the training dataset. Several architectures of machine learning prediction models are selected to fit the data. For the accuracy metric, [Kraemer et al., 2020] chooses to scale the existing mean absolute percentage error (MAPE) metric to eliminate scale dependency, variations of seasonality, and prevent this metric from becoming very large when the produced energy is close to zero. We select MAPE as our main accuracy metric because

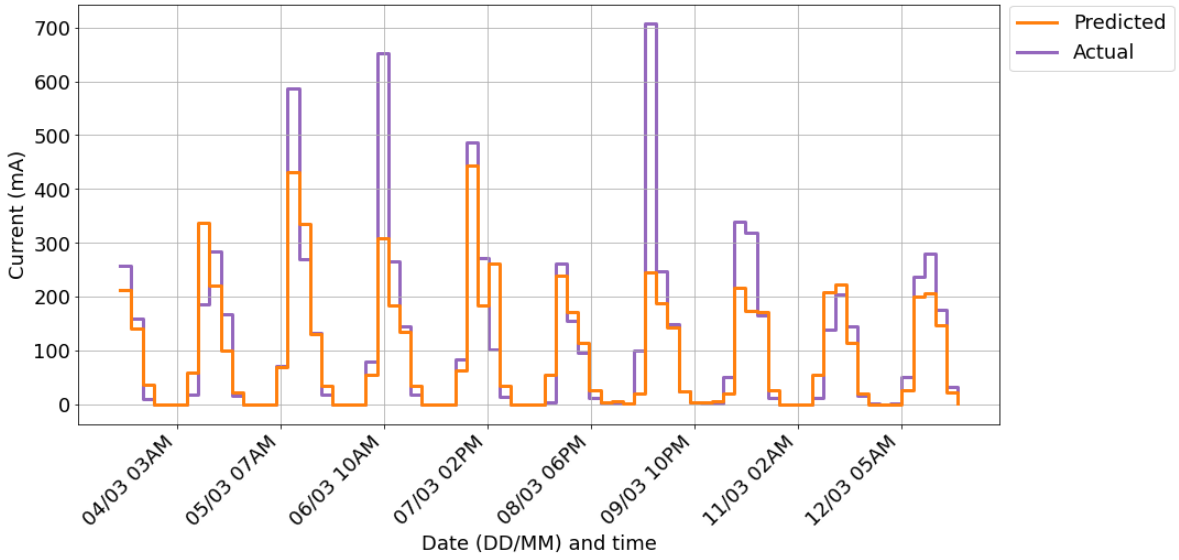
it still eliminates seasonality, and we choose to take into account its large values in our models for small amounts of energy intake. Indeed, we do not want our solar energy intake estimator to give non-null values during the night. Table 6.2 shows the performances of models ordered from best MAPE to worse. The other performance metrics are mean squared error (MSE), R-Squared (R^2) and the maximal error among the test set. The base unit for the target variable is the milliampere. The results confirm that random forest regressors (RFR) are quality models for solar intake prediction (as shown in [Kraemer et al., 2020]). In our case, 5 or 10 estimators, a hyperparameter of RFR, achieve the best results. This is lower than the 30 estimators which are the best settings for the cited article. This difference in value can be explained by the difference between datasets as predictor variables differ between the two study cases. Also, differences can be explained by weather data observations being spaced by 3 hours in our case as opposed to 1 hour in the cited article.

Model Name	**MSE**	R^2	**Max Error**	**MAPE**
RFR (5 estimators)	4843	0.80	392.5	4.0e+14
RFR (10 estimators)	3981	0.84	360.6	4.3e+14
RFR (20 estimators)	6693	0.72	426.3	4.7e+14
RFR (30 estimators)	7156	0.70	432.3	5.2e+14
RFR (40 estimators)	7465	0.69	435.0	5.5e+14
RFR (50 estimators)	6953	0.71	414.2	5.7e+14
RFR (60 estimators)	6889	0.72	411.6	5.7e+14
RFR (80 estimators)	7322	0.70	422.2	6.0+14
RFR (90 estimators)	7054	0.71	423.5	6.0e+14
RFR (70 estimators)	7079	0.71	415.5	6.1e+14
RFR (200 estimators)	7248	0.70	435.2	8.7e+14
RFR (150 estimators)	7224	0.70	437.5	9.4e+14
RFR (100 estimators)	7087	0.71	423.0	1.1e+15
Gradient Boosting Regressor	7374	0.69	471.1	8.7e+15
ElasticNet	12414	0.49	528.7	2.8e+16
SVR	27655	-0.14	656.0	3.3e+16
SGD Regressor (L1 norm penalty, alpha = 1e-3)	10184	0.58	464.9	5.2e+16
SGD Regressor (Elastic Net penalty, alpha = 1e-4)	10192	0.58	465.1	5.2e+16
SGD Regressor (L2 norm penalty, alpha = 1e-4)	10193	0.58	465.1	5.2e+16
Ridge Regression	9238	0.62	417.7	5.8e+16
Linear Regression	9304	0.62	407.5	6.4e+16
Kernel Ridge Regression	31379	-0.30	582.5	1.8e+17

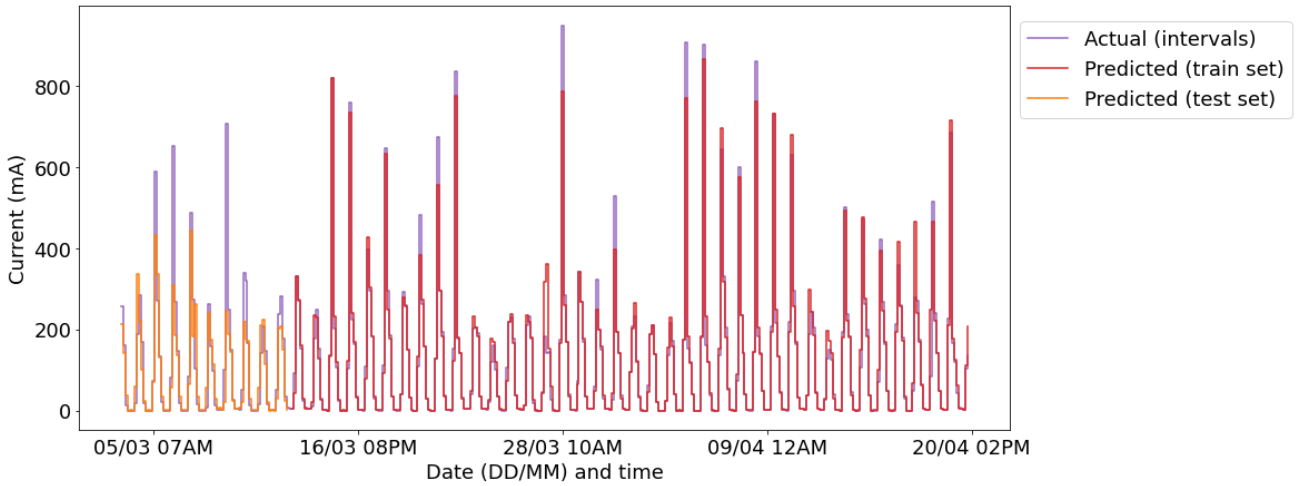
Table 6.2: Summary of machine learning models' performances for the prediction of solar energy production. Models are ordered from lowest MAPE to highest.

Figure 6.4a shows the actual and predicted energy data in chunks of 3 hours. In

this graph, the absolute average error is 37.8 mA per chunk of 3 hours. Also, the non-absolute average error (actuals subtracted from predicted) is -17.5 mA, showing that the model's predictions underestimate the actual values We also share the daily absolute average error for the nine full days displayed in Figure 6.4a from the 4th to the 12th of March 2022: 25.7 mA. The non-absolute error (actuals subtracted from predicted) of daily averages is -18.0mA, which also shows the tendency of our model to underestimate photovoltaic production. It is crucial to obtain a modest model which rather underestimates the actual value. Otherwise, when this model will be embedded into the reinforcement learning environment, there will be a risk of non-null estimations of the energy budget when the battery is empty.



(a) Predicted and actual values of solar intake for the test set



(b) Predicted and actual values of solar intake for the test set and the training set. The more realistic (more continuous) value of the solar intake is also shown.

Figure 6.4: Graphs of the actual and predicted solar intake values

Figure 6.4b shows the full picture of the data. It includes the data already shown

in Figure 6.4a on the left and the equivalent for the training set on the right. In other words, the orange part of the two graphs is the exact same data: the predicted values of the test. For the right part (colored in red), the model is fed the training set as a test for this visualization. It is noticeable that the model performs better on the training data than the test data, which is normal behavior for a machine learning model. The 3-hour chunks shown on these two graph show peaks that are smaller than the energy production peak of Figure 6.3, which are around 1.4 A. It indicates that the peaks of the current production sampled at 3.6 Hz are considerably shorter than 3 hours, and quickly decrease to less than 800 mA, which is close to the value of the maximal 3-hour averaged production.

For the following section, the RFR with 5 estimators is selected to be incorporated into the reinforcement learning environment in order to produce estimates of the solar energy intake and therefore update the energy budget at every step.

6.3 Methods

For the training algorithm, four RL methods are selected for comparison: Proximal Policy Optimization (PPO), Recurrent PPO (RPPO), Trust Region Policy Optimization (TRPO) and Advantage Actor Critic (A2C). We use the implementation introduced by [Raffin et al., 2021] for these methods.

6.3.1 Proximal Policy Optimization

PPO was first introduced by OpenAI[4] in 2017. It is a policy-based RL method that has proven to obtain state-of-the-art results for different benchmarks [Schulman et al., 2017]. It relies on an exploration strategy that allows the agent to collect data during a certain number of iterations before processing this data by updating its policy. This exploration strategy reduces the variance of updates, compared to methods that update the policy at every step in the environment. PPO is designed to be a simple, scalable and effective algorithm for learning policies that can operate in high-dimensional continuous action spaces. It uses a clipped surrogate objective function to update the policy parameters, which ensures that the new policy is not too far away from the previous policy to maintain stability during training. PPO's advantages include its simplicity and ease of implementation, fast training time and good sample efficiency.

6.3.2 Recurrent Proximal Policy Optimization

Recurrent PPO's ideas were introduced in [Wu et al., 2017]. RPPO is a policy-based RL method that uses recurrent neural networks in order to model temporal dependencies for time-series decision-making tasks. It uses the Kronecker-factored approximation (introduced in [Martens and Grosse, 2015]) to efficiently compute the inverse Hessian matrix of the policy, which enables it to handle long-term dependencies. RPPO's advantage is its ability to handle temporal dependencies and long-term credit assignment, as well as its good performance on tasks with high-dimensional inputs, such as video

games and robotic control tasks. One of the drawbacks of RPPO is its computational complexity, which can limit its scalability. It has achieved state-of-the-art performance on benchmarks like the OpenAI Gym robotics control suite.

6.3.3 Trust Region Policy Optimization

TRPO was introduced in [Schulman et al., 2015]. The distinctive characteristic of TRPO is close to PPO's, as it uses a trust region constraint to ensure that the policy update is not too aggressive, which can prevent the policy from diverging during training. It can handle stochastic policies well and has good convergence properties, but can sometimes face the computational complexity challenge for large-scale problems.

6.3.4 Advantage Actor Critic

A2C (introduced in [Mnih et al., 2016]) is an actor-critic reinforcement learning method that uses parallel agents to collect experience and a central critic element to estimate the value function. A2C combines the advantages of policy gradient methods (such as PPO) with value-based methods (such as Deep Q-Network) by using both an actor and a critic network to estimate the policy and the value function, respectively. The use of parallel agents to improve sample efficiency and speed up training. A2C can suffer from high variance in the gradient estimates, which can lead to slow convergence or instability.

6.3.5 Training phase implementation and optimization

The performance of a model is defined as the cumulative reward (defined in Section 6.2.3) at the end of the simulation.

We first focus on the optimization of the discount factor parameter through a sensitivity analysis. This parameter depicts how much future potential rewards are taken into account when selecting each task and can take values between 0 and 1. Low discount factor values create models which care about immediate reward. Higher discount factors create models which put some importance on future rewards. Figure 6.5 shows that high values of the discount factor parameter for training produce better results for the considered goal. In other words, a model which updates its weights while considering more future potential rewards is better than a model which focuses more on immediate reward. This is an expected result since the rules of the environment make the reward depend on past outcomes (boost of tasks' values if some tasks are not used for several iterations), so anticipating a few steps ahead should be beneficial behavior. From this point, models use a discount factor of 0.99.

Then, 108 models are trained per RL method in order to be able to obtain representative average values when grouping their performance scores. They are all trained through 1 000 000 iterations of the RL environment.

During the training phase, we aim at avoiding overfitting. Overfitting occurs when a model becomes too complex and is too closely fit to the training data, resulting in poor generalization to new data. As an RL model is being trained, it updates its policy

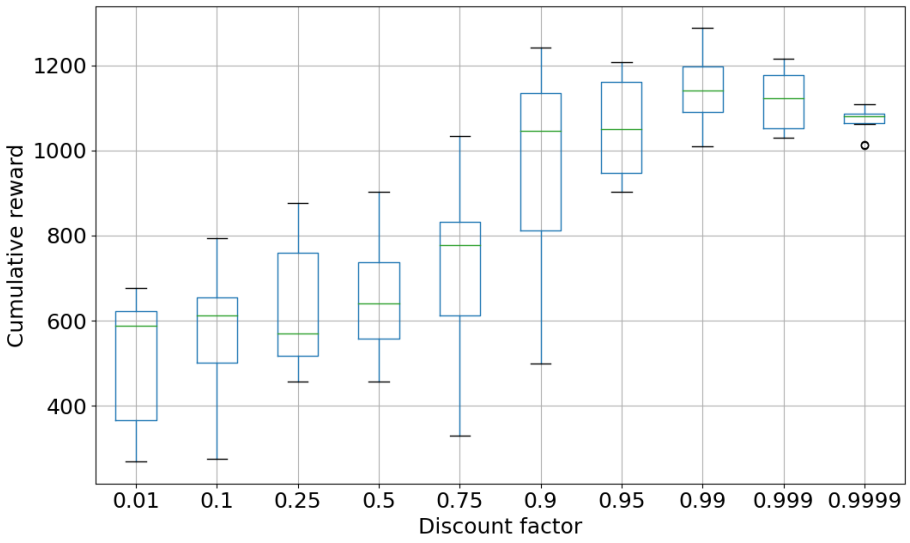


Figure 6.5: Cumulative reward for 30-day simulations of different models, grouped by the discount factor parameter at which models are trained

by optimizing its parameters to maximize the expected reward. However, if the model is trained for too long, it may become too specialized to the specific scenarios and situations it encountered during training and may not generalize well to new scenarios. To prevent this phenomenon, the performances of models are tested at regular intervals of iterations during the training phase. The performance check relies on the performance of a given model at a one-month best-effort strategy simulation, starting at a random time. More precisely, models are tested every 16 384 iterations during their training phase, allowing to keep high granularity along the training phase and to save a model during the middle of the training phase if the cumulative reward is positive and better than at any previous check. The specific value of 16 384 is chosen because it is a power of 2 and the implementation of RL methods by [Raffin et al., 2021] uses smaller powers of 2 for the hyperparameter *n_steps*, which is the number of iterations to run before updating the weights of a model. Therefore, our implementation of the training phase always performs the performance check exactly after an update. If not, it would ignore the iterations after the previous update of weights and before the performance check.

6.3.6 Implementation of heuristics for task allocation

We provide a point of comparison for the RL models with two heuristics implemented manually: a simple one and an advanced one. Both heuristics are fed the same observations of the environment as the RL agents: the past outcomes and the energy output of the solar panel.

The first heuristic, introduced in Algorithm 2 performs a random selection of two tasks if the success history is encouraging, and selects zero tasks otherwise. To do so, we introduce a new variable $confidence$ which depicts how well the system performed

during the recent past iterations (line 1): it is the ratio of the sum of the values of the past outcomes (0, 1 or 2) over the theoretical maximum of this sum — when all n previous outcomes are successful. Therefore, this metric is used to split unhealthy operations (line 2) where no tasks should be selected from healthy operations (line 5) of the system where two tasks are randomly selected as the action.

In the second heuristic presented in Algorithm 3, the variable $confidence$ is also used and the variable E_{max} is introduced. It represents the maximal amount of energy gained by the system between two iterations. In our work, this maximal amount of energy E_{max} is observed to be 1422 J (948 mA at 5 V during 5 minutes). E_{max} helps to assess the quantity of the current energy intake through the R_{energy} ratio introduced at line 1. Together with the $confidence$ variable, they are the two parameters that define the limit of energy below which the heuristic should stay when it selects tasks. Line 9 fixes the condition for adding more tasks to the list. On the left side of the inequality, there is the ratio of the sum of energy costs of currently selected tasks over the sum of all possible tasks' energy costs. This ratio is compared with a multiplication between a term that includes $confidence$ and a term that includes R_{energy}. The first term acts as a limiter for the selection of a task, and the lower the value of $confidence$, the more restrictive limit. Because $(confidence - 0.2)$ is always between 0 and 1, the 1.2 exponent's purpose is to penalize the smallest values even more. The second term has the same limiting effect but depends on the R_{energy} variable: the lower R_{energy}, the more restrictive. We tested several sets of the three numerical values shown in line 9 and optimized their values through the repetition of successive simulations of the environment while this heuristic was in action. We manually converged to 0.2, the 1.2 exponent and 1.3 for those three values as they induced the best performances by the heuristic all the while maintaining safe energy budget management. In the second heuristic, tasks are also selected in a random order, one task by one task and stopping before the energy limit is crossed (line 9).

Algorithm 2: Simple heuristic for selecting a subset of tasks at iteration t

Variables: Set of tasks: $T = \{T_1, T_2, ...\}$
Input : List of past outcomes of size n: $[S_{t-1}, S_{t-2}, ..., S_{t-n}]$
Output : Set of tasks $T_{selected}$

1 $confidence = \frac{\sum_{i=0}^{n-1} S_{t-i}}{2 * n}$
2 **if** $confidence < 0.5$ **then**
3 | return [ ]
4 **end**
5 **else**
6 | $T_{selected} = randomSample(T, 2)$
7 | return $T_{selected}$
8 **end**

Algorithm 3: Advanced heuristic for selecting a subset of tasks at iteration t

Variables: Set of tasks: $T = \{T_1, T_2, ...\}$
Their cost (energy) $E = \{E_1, E_2, ...\}$
The best-case solar-produced energy over a time interval: E_{max}
Input : Energy produced at time interval t: E_t
List of past outcomes of size n: $[S_{t-1}, S_{t-2}, ..., S_{t-n}]$
Output : Set of tasks $T_{selected}$

$R_{energy} = \frac{E_t}{E_{max}}$
$confidence = \frac{\sum_{i=0}^{n-1} S_{t-i}}{2 * n}$
shuffle(T)
$E_{selected} = 0$; $T_{selected} = [\,]$
if $confidence < 0.5$ **then**
  return $T_{selected}$
end
for T_i *in* T **do**
  if $\frac{E_{selected}}{\sum E_i} > (confidence - 0.2)^{1.2} * \min(1, (1.3 - R_{energy}))$ **then**
    break
  end
  $E_{selected} = E_{selected} + E_i$
  $T_{selected}.append(T_i)$
endFor
return $T_{selected}$

6.4 Results

In this section, the benchmark chosen to evaluate the performances of models is a 30-day simulation taken during the summer of 2022, which corresponds to our EAPBS's deployment period. The performances of models are compared with the ones of two heuristic methods, which are defined in Algorithms 2 and 3.

Figure 6.6 shows the performances of all RL methods averaged over each category's 108 models. We observe that TRPO models produce on average the best performances and the ones with the smallest variances across different models. Compared to the TRPO models, A2C, RPPO and PPO suffers from more inconsistency. The standard deviation over the different models (displayed in brighter colors on Figure 6.6) is higher than TRPO by a factor of 8.5, 5.5 and 3.7 for A2C, RPPO and PPO respectively. However, the best of each method still achieves performances that are in the same order as TRPO's best performance – our best reinforcement learning model. Table 6.3 summarizes the best model's performance for each method over a 30-day simulation during July 2022. For every model, the discount factor is set at 0.99, but the number of iterations used to collect experience before updating the policy during the training phase, n_steps, can vary. The best models show a peak of performance for values of

512, 1024 or 2048. In our case, the successive routines performed in the environment as spaced by five minutes. In practice, it means that the best-performing agents collect exploration data between 1 day and 19 hours ($n\_steps = 512$), and 7 days and 3 hours ($n\_steps = 2048$) before updating their policy. This behavior is wanted, because we prefer to feed several days of data rather than just a few hours, which would be very biased because of the instantaneous weather and the day-night cycle.

Another observation of Table 6.3 is the number of training steps at which the best models' performances are achieved. It is at first counterintuitive that the RPPO's best version is met after 622 592 rather than a value closer to 1 000 000 iterations, where the models' final form is met. However, the technique described in Section 6.3 used to prevent overfitting allows us to pinpoint best-performing models before the end of the training phase.

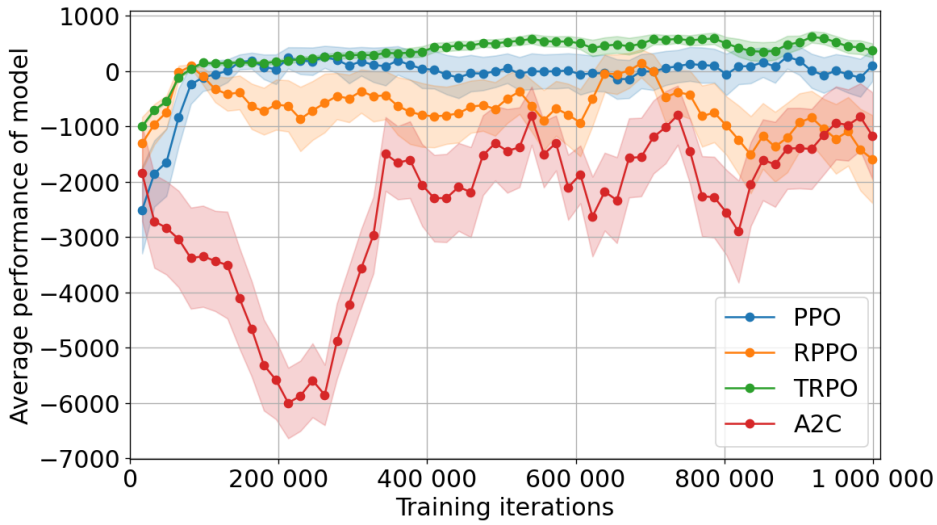


Figure 6.6: Performance of each method over the number of training iterations

Algorithm	**Training steps**	**n_steps**	**Best model's performance**
TRPO	999 424	2048	1291
PPO	835 584	512	1290
RPPO	622 592	1024	1267
A2C	901 120	2048	1231
Advanced heuristic	-	-	1226
Simple heuristic	-	-	1014

Table 6.3: Best model's performance (cumulative reward) for every method tested on a 30-day simulation. Models are ordered from the highest performance to the lowest.

Figure 6.7 highlights the best obtained TRPO-based model with one 30-day simulation (Figure 6.7a) and one 45-day simulation (Figure 6.7b). It corresponds to the TRPO model whose metrics are displayed in Table 6.3. We display the data of the

best TRPO model (darker color), our advanced heuristic (middle color) and our simple heuristic (clearer color). The cumulative performance over time and the energy budget are shown in shades of blue and green, respectively. At the bottom of the graph, we display the weather category (introduced in Section 6.2.4) over time. Yellow, gray and blue respectively correspond to clear, cloudy and rainy weather categories.

The 30-day simulation (Figure 6.7a) is chosen to be during July 2022, which is an average summer month in terms of weather. In these conditions, we want to show a comparison between the three strategies when the weather conditions are at their best for several weeks in a row. TRPO outperforms the two heuristic strategies after 30 days. However, the difference between TRPO and the advanced heuristic is substantial only after the middle of the 30 days is passed, and increases significantly towards the end. In parallel, we note a difference between the two managements of the battery levels: the heuristic is not capable of switching gears, whereas TRPO uses the battery to make it reach values close to the first defined limit.

The 45-day simulation (Figure 6.7b) starts at the beginning of May 2022 and ends around the end of June. According to the lower strip that represents the weather category, the weather during the 45-day window is more cloudy and rainy than the summer's 30-day window. This difference explains the faster decrease of the battery percentage over time in Figure 6.7b for all strategies: the energy intake of the solar panel into the battery becomes lower under cloudy or rainy conditions than sunny conditions, and no strategy accounts for this change. This graph highlights the superiority of the TRPO-based agent's strategy for long and unfriendly weather conditions: although the two heuristics reach energy budgets close to B_{critic} slower than the TRPO strategy, they do not handle this critical state well because their cumulative rewards are pulled to values below 0. On the other hand, the RL approach shows adaptation and even though the agent only sees the history of the five last iterations together with the instantaneous energy intake, the energy budget close to the critical value does not affect the cumulative reward as much as for the heuristics: it stays at worst stable at the scale of a week (from the 1^{th} to the 8^{th} of June for example), and quickly recover when better weather conditions are back (after mid-June). Although close to practically damaging values, the energy budget is kept within the acceptable range as it never flirts with 0%. More importantly, the RL approach is more consistent over 45 days in terms of reward variations.

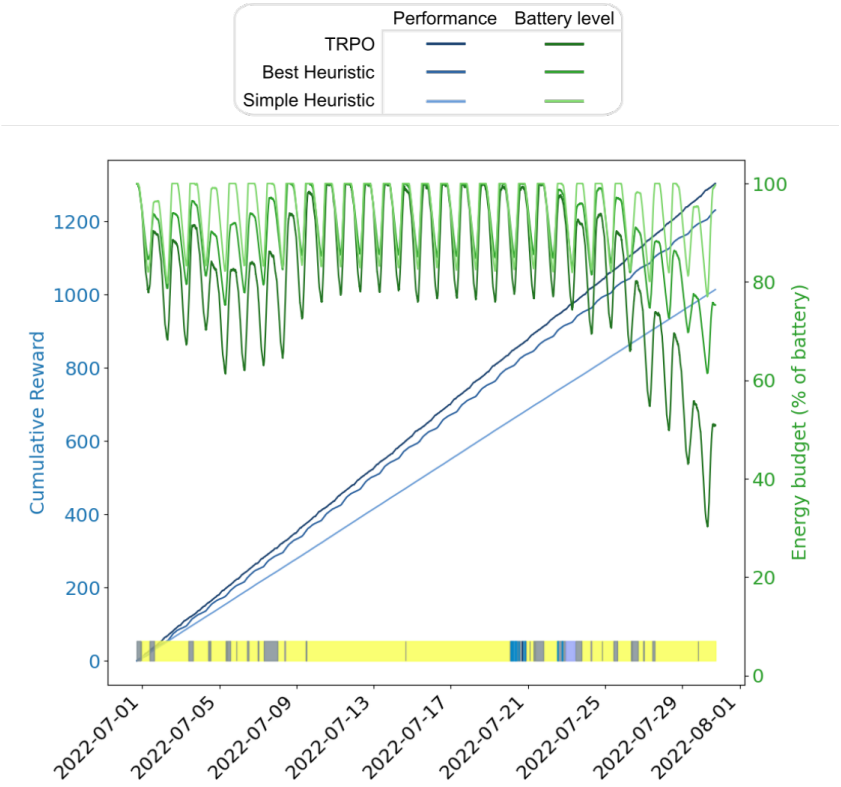
6.5 Conclusion of service placement models

In this chapter, we explored the different reinforcement learning (RL) approaches to optimize the problem of service placement in autonomous energy-harvesting stationary IoT systems (AEHSS). In parallel, we implemented our own solar power intake prediction model using our collected data. This implementation helped to validate the work presented in [Kraemer et al., 2020] because it identifies random forest regressors as the best predictors of photovoltaic energy intake based on meteorological data, even if our weather data differs from the one in the initial work in terms of composition and acquisition frequency. This prediction model was integrated into the environment of the reinforcement learning methods to simulate real-life data.

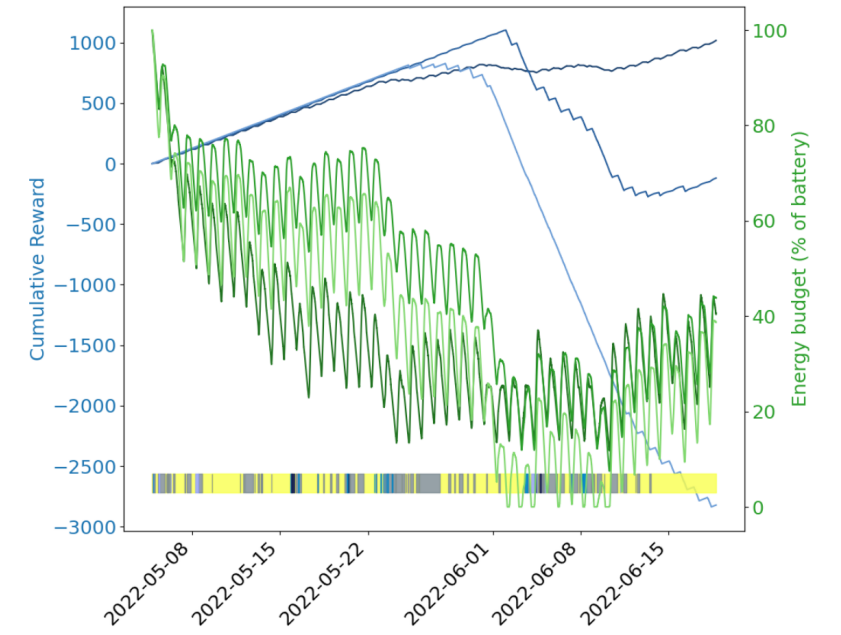
We then introduced a reinforcement learning environment applicable to the case of energy-efficient task allocation in AEHSS. The RL agent must pick a set of tasks at each iteration — later in practice, at each beginning of the data acquisition routine — while observing a set of two parameters: its own success history and the current energy intake. The agent receives positive rewards if the action can effectively be performed while the energy budget remains out of the critical zone. On the other hand, negative rewards are given to the agent when the selected set of tasks cannot be performed with the remaining energy available, or can be performed in a critical energy budget state.

We repeatedly trained four types of RL methods: PPO, TRPO, RPPO and A2C. During the training, the overfitting phenomenon is reduced by iteratively measuring the performance of a model. The best-proposed models outperform heuristics which are specifically designed to perform well at this task and in this environment. TRPO-based models produce the best results in terms of maximal performance: 1291 cumulative reward over a one-month simulation, which is an improvement of 5% compared to the advanced heuristic. Our RL-based solutions are also better at managing energy levels close to values considered critical by the environment as they can hover over those values only thanks to the observed history and the estimation of energy production. TRPO models are also the best in terms of stability over the different instances. However, the variance of the results of A2C, RPPO and PPO models remains high, which highlights the remains of overfitting, but the best individual models efficiently select tasks while staying within an acceptable energy budget range. It is worth noting that, whenever such a model would need to be retrained in a different environment (for example, because of a change of location), the corresponding heuristic would need to be manually tuned for this specific case.

In future works, we aim to make the rules of the environment more complex to better model weather and energy harvesting dynamics. Also, it would be beneficial to create a dedicated testing benchmark to obtain more robust results and optimize the consistency of results obtained by the reinforcement learning methods. The explicability of the strategies of the best models also remains to be analyzed: so far, we suppose that the energy intake allows the agent to keep an eye on the instantaneous variations, while the success history allows it to have an idea of past events. Another important point is to implement a multi-client/multi-server environment where the agent will not only select tasks but also place them at the edge and in the cloud. Finally, the validation of our models will be performed through actual deployment in the field.



(a) Cumulative reward and battery level for one 30-day simulation of the best TRPO-based trained RL agent



(b) Cumulative reward and battery level for one 45-day simulation of the best TRPO-based trained RL agent

Figure 6.7: Comparison of performances and energy budget's evolutions for the two heuristics and our best model

Discussion on the scientific context of this thesis

Contents for chapter 7

7.1 Beekeeping as an applied field for informatics research

The research in sustainable, autonomous, smart and energy-efficient Internet of things (IoT) systems applied to precision beekeeping is an interdisciplinary scientific topic. The main fields of biological research are apidology and precision beekeeping. The latter also belongs to the computer science field together with several other specialized topics: the hardware design of IoT systems, the software development for IoT services, energy conservation in IoT systems with embedded services, the artificial intelligence methods applied to precision beekeeping and energy consumption data, the orchestration of tasks between the edge and the cloud, and the selection of tasks within the edge systems.

The main application domain, beekeeping, is not typical for a computer science thesis, and it is easy to move away from informatics to focus on other equally interesting topics that do not coincide with the primary theme. One challenge during these three years has been to reconcile computer science with the cross-disciplines that constitute this thesis. Beekeeping is fertile ground to perform field experiments and has been key to the improvement of the proposed computer solution.

On the one hand, part of the collected apiary data was used to calibrate our simulations. There are no such databases in the literature, so without them, it would have been more challenging to obtain realistic simulations that reflect field events, and thus the quality of service potentially given to beekeepers by our results would have been strongly affected. Losses of data because of hardware failure were detected in the field during the collection period, which is usually lacking in classic computer lab models. The models trained thanks to this imperfect data take into account the real conditions observed in beekeeping, and more globally for IoT systems placed outdoors and subjected to the weather. By using these losses during training, our models underestimate

rather than overestimate: for example, they underestimate the number of clients to allocate per server for cloud computing. This characteristic is key when one wants to deploy outdoor infrastructures subject to environmental variations.

Then what does this connected hive look like if it is deployed on a large scale, say, the scale of France? The large-scale simulation of our system as it is, is the most direct way to this answer. But in an approach of contribution in computer science, the objective has been refocused by generalizing to an IoT system similar to ours. The variables are initialized for our connected hive and its services, but the scientific contribution allows a flexible parameterization that can be applied to any autonomous energy-harvesting stationary IoT systems (AEHSS).

7.2 Unite several computer science subfields

The subtopics of computer science present in this thesis are numerous. In my initial representation of a thesis, one of them (e.g., reinforcement learning for energy consumption optimization, or the development of computer vision algorithms to detect bee pathologies) is the focus of the scientific contribution made by the thesis. Here, the work is more interdisciplinary compared to other theses, and it has been another important challenge to make these sub-themes enter a virtuous circle to produce a work that has consistency. The four main parts of the thesis are complementary: the literature review allows, in addition to introducing beekeeping and existing solutions in precision beekeeping, to set a course for the thesis that is focused on a better way to collect data as well as focus on the aspects that make a connected hive frugal. This work coupled with the energy benchmark of the existing precision beekeeping system, allowed for the development of the material contribution of the thesis: the energy-aware and autonomous connected hive.

7.3 Toward an "auto-tuning" smart beehive

One of the underlying goals at the start of the development of a hive with integrated services was to get closer to an "auto-tuning" hive: a hive able to create its own routines' policy according to the observed data, both apidology data and energy consumption data. This axis is explored in Chapter 6 through a reinforcement learning model. The flexibility comes with the training data that is defined by the environment. For instance, the solar energy intake prediction model can be trained thanks to data collected in France or in Norway. Both training datasets will be different because of the more intense sun exposure in France. The France-based reinforcement learning environment will generate energy budget intake values larger than Norway's. The resulting agents will therefore have different strategies: more optimistic for France, and more conservative for Norway.

Also, one can imagine a solution that estimates the stability of the health of a bee colony. Such stability could be predicted with previous diagnoses, the history of the colony's health, the climate, the weather, and the dates of the beekeeper's visits. With

this stability estimation, the task scheduling model can regulate the frequency of the routines as well as their intensity in terms of energy consumed.

Conclusion

Contents for chapter 8

The main objective of this thesis is to contribute to a more energy-efficient usage of IoT services, applied to precision beekeeping. In the following sections, the contributions toward this objective are summarized, and future works are listed.

8.1 Contributions of this thesis

The scientific object presented in this thesis is multifaceted but can be summarized as an expertise on precision beekeeping mixed with methods to optimize the energy consumption of autonomous energy-harvesting stationary IoT systems (AEHSS). The four main contributions can be described as follows:

1. Literature review of the current state of the art in smart beekeeping
This review introduces the challenges met by beekeepers and the colonies of bees and then describes how connected beehives are designed across the scientific literature. Many solutions exist and the placement of the sensors, the choice of the hardware, the network and the energy source are all diverse: there are as many connected beehives as precision beekeeping initiatives. However, this kind of hardware raises challenges involving price, energy and reproducibility. Cheap, energy-efficient and open-source solutions are highlighted in the review and are encouraged. Smart beekeeping provides a wide range of services to beekeepers thanks to the processing of data. The possible AI-based solutions are listed, categorized and compared. This review lays the foundation for the future of the field of precision beekeeping which involved more energy-efficient solutions, better reproducibility, and a wider range of collected data.

One of the trends emerging from the precision beekeeping (PB) literature is the inconsistency over different articles regarding the durations of data collection and the costs (price, energy) analysis of systems and services. Future PB research should focus more on those topics, and the energy data used to provide insights should tend to be collected in the field rather than in the lab.

2. Autonomous energy-aware smart beekeeping system After an energy consumption analysis of an existing system, an energy-aware PB system is proposed. It is based on a Raspberry Pi and integrates the use of environmental sensors, microphones and cameras, as well as sensors that are placed directly in the electrical wires to collect energy consumption data. This system runs through photovoltaic energy, collects bee-related data at regular intervals, and brings a novel implementation amongst the PB systems of the literature: the ability to constantly monitor its energy consumption as well as the energy intake of the battery from the solar panel. The data collected during the 2022 high season is shared through an open-source dataset and shows potential for future AI-based PB services and serves as a base for the values of the variables passed into the simulation and optimization models.

3. Analysis and large-scale simulation of autonomous energy-harvesting stationary IoT systems The energy consumption data from the operation of the introduced system is analyzed and the energy cost of the considered AI-based service involving a computer vision model is optimized. Then, two scenarios are proposed and considered: one scenario that only involves the edge device and the other that involves the edge and a cloud server which relieves the edge for resource-heavy computation. Those scenarios are simulated at a large scale with some modeling of losses, to identify the tipping points at which one scenario becomes better than the other.

A simulation of AEHSS systems is performed. It shows that the use of a cloud server can be beneficial even with some external deterioration. Such a scenario involving the edge infrastructures and cloud servers becomes better than a scenario only using the edge when the heavy computation at the edge can easily be delegated to the cloud and when the number of edge devices (beehives in our case) increases to at least 500.

4. Reinforcement learning service placement model for task selection A reinforcement learning model that iteratively picks a set of tasks in an energy-efficient manner is proposed. To do so, an environment that mimics real-life dynamics is modeled, with the help of a photovoltaic energy intake prediction model which confirms the knowledge gained in [Kraemer et al., 2020]. Reinforcement learning models are suited for such tasks as they operate in dynamic environments which return new observations (the next iteration's weather and the estimation of the photovoltaic energy intake) and some rewards (depending on the success or failure of the selected tasks). Four reinforcement learning methods are selected to train the agent toward the best picking strategy under a constrained and varying energy budget. We manage to train models of all four architectures which achieve better performance than our best heuristic, but the challenge of consistency of reinforcement learning models' training remains. Our best model is based on the Trust Region Policy Optimization method.

8.2 Future Works

The improvements of this thesis are interdisciplinary and revolve around several questions.

First, the deployment of the PB services alongside the reinforcement learning model in our energy-aware precision beekeeping system (EAPBS) is the next part of the field research. This will allow the collection of more bee-related data and most importantly more energy consumption data of the running system. The initial parameters of the reinforcement learning environment will be adjusted to better fit real conditions.

Other questions emerged during and after the development of the EAPBS system: over 1 400 000 hives in France, for example, how many should be equipped to maximize the added value of PB services compared to the invested resources? In other words, the question of redundancy of the data of neighboring beehives should be explored. In parallel, another important aspect is the added value of connected hives working collaboratively, which could involve using available connected hives within their range (Wi-Fi or Bluetooth) to perform tasks that would overload the other.

This also brings the question of the rest of the life cycle of EAPBS objects: how to optimize all the steps of the life cycle assessment of a smart beehive? For example, if the costs of the beekeeper's activity are incorporated in our reinforcement learning model, especially the transportation which is non-negligible because of the distance that professional beekeepers usually need to cover between their home and their apiaries, is there any other service placement strategy that is close to the optimum? In a reinforcement learning-based implementation of this solution, an approach would be to mix a list of tasks different from the ones of PB, such as the car ride of a beekeeper from home to an apiary, with the beekeeping services, rather than exclusively starting from a list of beekeeping services, to optimize more steps of the life cycle of a smart beekeeping solution.

Future work should investigate the profitability timeline of deploying the EAPBS in terms of energy saved, accounting for the collection of raw materials, production, and transport of the hardware. One potential approach for large-scale deployment confirmation could involve emulating the network of connected hives using large-scale computer equipment, rather than simulating it. Additionally, data storage is a crucial point to address when scaling up the system, as this thesis' current implementation with five connected hives generated only a few hundred gigabytes of data in 2022.

Regarding the deep learning queen classification model, the manuscript analyzed the inference costs, but future work should also investigate training costs, including data storage and computational resources. It is also important to understand how often retraining is necessary to maintain maximum accuracy and whether colonies "exchange footprints" during the few years of a queen's life. Finally, while some losses of the system have been identified, future work should explore losses due to extreme temperatures or humidity and deterioration of the material in both the hermetic box and inside the hive.

To sum up, this thesis has provided answers to problems about PB and the energy in AI-based IoT systems, and has opened up others that deserve to be taken into account and analyzed. We encourage all research initiatives aimed at deepening the topics

covered in this manuscript or dealing with cross-cutting topics. We make ourselves available for dialogue allowing the initialization of such work.

Publications

International Conference

[1] H. Hadjur, D. Ammar, and L. Lefèvre, "Analysis of energy consumption in a precision beekeeping system" In Proceedings of the 10th International Conference on the Internet of Things (IoT '20). Association for Computing Machinery, New York, NY, USA, Article 20, pp. 1–8, 2020. https://doi.org/10.1145/3410992.3411010

International Journal

[2] H. Hadjur, D. Ammar, and L. Lefèvre, "Toward an intelligent and efficient beehive: A survey of precision beekeeping systems and services" Computers and Electronics in Agriculture, vol. 192, pp. 1-16, 2022. https://doi.org/10.1016/j.compag.2021.106604

International Workshop

[3] H. Hadjur, D. Ammar and L. Lefèvre, "Services Orchestration at the Edge and in the Cloud for Energy-Aware Precision Beekeeping Systems" 2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), St. Petersburg, FL, USA, pp. 769-776, 2023. https://doi.org/10.1109/IPDPSW59300.2023.00129

Presentation

[4] Hugo Hadjur, Doreid Ammar, Laurent Lefèvre, "Conception de systèmes connectés durables, autonomes et à basse consommation énergétique pour un réseau de ruches connectées distribuées" Journées du GDR Réseaux et Systèmes Distribués, 27-28 avr. 2022 Rennes (France) sciencesconf.org:gdr-rsd2022:381481

[5] Hugo Hadjur, "Optimiser la sélection de tâches sous contrainte énergétique dans des systèmes IoT grâce à l'apprentissage par renforcement" GreenDays2023, 27-28 mars 2023 Lyon (France).

Open-source Dataset

[6] H. Hadjur, D. Ammar, and L. Lefèvre, "EAPBSdata: Energy consumption and bee dataset of five connected beehives of one high season in France" https://doi.org/10.5281/zenodo.7880085

Bibliography

[ITU, 2012] (2012). Overview of the internet of things. https://handle.itu.int/11.1002/1000/11559. Accessed: 2023-05-05.

[FAO, 2016] (2016). Pollinators vital to our food supply under threat. http://www.fao.org/news/story/en/item/384726/icode/. Accessed: 2023-05-05.

[Abdollahi et al., 2023] Abdollahi, M., Henry, E., Giovenazzo, P., and Falk, T. H. (2023). The importance of context awareness in acoustics-based automated beehive monitoring. *Applied Sciences*, 13(1).

[ADA-France, 2022] ADA-France (2022). L'apiculture professionnelle en chiffres. https://www.adafrance.org/le-marche-du-miel-en-france/. Accessed: 2023-05-05.

[Amirtharaj et al., 2018] Amirtharaj, I., Groot, T., and Dezfouli, B. (2018). Profiling and improving the duty-cycling performance of linux-based iot devices. *CoRR*, abs/1808.10097.

[Ammar et al., 2019] Ammar, D., Savinien, J., and Radisson, L. (2019). The makers' beehives: Smart beehives for monitoring honey-bees' activities. In *Proceedings of the 9th International Conference on the Internet of Things, IoT 2019, Bilbao, Spain, October 22-25, 2019*, pages 16:1–16:4.

[Anand et al., 2018] Anand, N., Raj, V. B., Ullas, M. S., and Srivastava, A. (2018). Swarm detection and beehive monitoring system using auditory and microclimatic analysis. In *2018 3rd International Conference on Circuits, Control, Communication and Computing (I4C)*, pages 1–4.

[Andrijević et al., 2022] Andrijević, N., Urošević, V., Arsić, B., Herceg, D., and Savić, B. (2022). Iot monitoring and prediction modeling of honeybee activity with alarm. *Electronics*, 11(5).

[Anuar et al., 2019] Anuar, N., Md Yunus, M. A., Baharuddin, M., Sahlan, S., Abid, A., Ramli, M., Amin, M., and Lotpi, Z. (2019). Iot platform for precision stingless bee farming. In *2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, pages 225–229.

[Aumann et al., 2017] Aumann, H., Payal, B., Emanetoglu, N., and Drummond, F. (2017). An index for assessing the foraging activities of honeybees with a doppler sensor. In *2017 IEEE Sensors Applications Symposium (SAS)*, pages 1–5.

[Aumann and Emanetoglu, 2016] Aumann, H. M. and Emanetoglu, N. W. (2016). The radar microphone: A new way of monitoring honey bee sounds. In *2016 IEEE SENSORS*, pages 1–2.

[Babic et al., 2016] Babic, Z., Pilipovic, R., Risojevic, V., and Mirjanic, G. (2016). 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*, III-7:51–57.

[Bjerge et al., 2019] Bjerge, K., Frigaard, C. E., Mikkelsen, P. H., Nielsen, T. H., Misbih, M., and Kryger, P. (2019). A computer vision system to monitor the infestation level of varroa destructor in a honeybee colony. *Computers and Electronics in Agriculture*, 164:104898.

[Borlinghaus et al., 2022] Borlinghaus, P., Odemer, R., Tausch, F., Schmidt, K., and Grothe, O. (2022). Honey bee counter evaluation – introducing a novel protocol for measuring daily loss accuracy. *Computers and Electronics in Agriculture*, 197:106957.

[Braga et al., 2021] Braga, A. R., Freitas, B. M., Gomes, D. G., Bezerra, A. D., and Cazier, J. A. (2021). Forecasting sudden drops of temperature in pre-overwintering honeybee colonies. *Biosystems Engineering*, 209:315–321.

[Bumanis, 2020] Bumanis, N. (2020). Data fusion challenges in precision beekeeping: a review. *Research for Rural Development 2020*.

[Catania and Vallone, 2019] Catania, P. and Vallone, M. (2019). Design of an innovative system for precision beekeeping. In *2019 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, pages 323–327.

[Catania and Vallone, 2020] Catania, P. and Vallone, M. (2020). Application of a precision apiculture system to monitor honey daily production. *Sensors*, 20:2012.

[Cecchi et al., 2019] Cecchi, S., Terenzi, A., Orcioni, S., and Piazza, F. (2019). Analysis of the sound emitted by honey bees in a beehive. *Journal of The Audio Engineering Society*.

[Cecchi et al., 2018] Cecchi, S., Terenzi, A., Orcioni, S., Riolo, P., Ruschioni, S., and Isidoro, N. (2018). A preliminary study of sounds emitted by honey bees in a beehive. In *Audio Engineering Society Convention 144*.

[Cejrowski et al., 2020] Cejrowski, T., Szymański, J., and Logofătu, D. (2020). Buzz-based recognition of the honeybee colony circadian rhythm. *Computers and Electronics in Agriculture*, 175:105586.

[Cejrowski et al., 2018] Cejrowski, T., Szymański, J., Mora, H., and Gil, D. (2018). Detection of the bee queen presence using sound analysis. In Nguyen, N. T., Hoang, D. H., Hong, T.-P., Pham, H., and Trawiński, B., editors, *Intelligent Information and Database Systems*, pages 297–306, Cham. Springer International Publishing.

[Cejrowski and Szymański, 2022] Cejrowski, T. and Szymański, J. (2022). Detection of anomalies in bee colony using transitioning state and contrastive autoencoders. *Computers and Electronics in Agriculture*, 200:107207.

[Chazette et al., 2016] Chazette, L., Becker, M., and Szczerbicka, H. (2016). Basic algorithms for bee hive monitoring and laser-based mite control. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8.

[Chen et al., 2012] Chen, C., Yang, E.-C., Jiang, J.-A., and Lin, T.-T. (2012). An imaging system for monitoring the in-and-out activity of honey bees. *Computers and Electronics in Agriculture*, 89:100–109.

[Chen et al., 2015] Chen, W., Wang, C., Jiang, J., and Yang, E. (2015). Development of a monitoring system for honeybee activities. In *2015 9th International Conference on Sensing Technology (ICST)*, pages 745–750.

[Chiron et al., 2013a] Chiron, G., Gomez-Krämer, P., and Ménard, M. (2013a). Outdoor 3D Acquisition System for Small and Fast Targets. Application to honeybee monitoring at the beehive entrance. In *GEODIFF 2013*, pages 10–19, Barcelona, France.

[Chiron et al., 2013b] Chiron, G., Gomez-Krämer, P., and Michel, M. (2013b). Detecting and tracking honeybees in 3D at the beehive entrance using stereo vision. *EURASIP Journal on Image and Video Processing*, 2013(1):59.

[Cook et al., 2022] Cook, D., Tarlinton, B., McGree, J. M., Blackler, A., and Hauxwell, C. (2022). Temperature Sensing and Honey Bee Colony Strength. *Journal of Economic Entomology*, 115(3):715–723.

[Corbari and Mancini, 2022] Corbari, C. and Mancini, M. (2022). Irrigation efficiency optimization at multiple stakeholders' levels based on remote sensing data and energy water balance modelling. *Irrigation Science*, 41.

[Cousin et al., 2019] Cousin, P., Căuia, E., Siceanu, A., and de Cledat, J. (2019). The development of an efficient system to monitor the honeybee colonies depopulations. In *2019 Global IoT Summit (GIoTS)*, pages 1–5.

[Crane, 1999] Crane, E. (1999). *The World History of Beekeeping and Honey Hunting*. Routledge.

[Cunha et al., 2020] Cunha, A., Rose, J., Prior, J., Aumann, H., Emanetoglu, N., and Drummond, F. (2020). A novel non-invasive radar to monitor honey bee colony health. *Computers and Electronics in Agriculture*, 170:105241.

[Davidson et al., 2020] Davidson, P., Steininger, M., Lautenschlager, F., Kobs, K., Krause, A., and Hotho, A. (2020). Anomaly detection in beehives using deep recurrent autoencoders. *SENSORNETS 2020*.

[Dimitrijevic and Zogovic, 2022] Dimitrijevic, S. and Zogovic, N. (2022). Machine learning advances in beekeeping. *Conference: 12th International Conference on Information Society and Technology - ICIST 2022*.

[Dimitrios et al., 2022] Dimitrios, K. I., Bellos, C. V., Stefanou, K. A., Stergios, G. S., Andrikos, I., Katsantas, T., and Kontogiannis, S. (2022). Performance evaluation of classification algorithms to detect bee swarming events using sound. *Signals*, 3(4):807–822.

[Dugan et al.,] Dugan, J., Elliott, S., Mah, B. A., Poskanzer, J., and Prabhu, K. iperf website. https://iperf.fr/. Accessed: 2023-05-05.

[Edwards-Murphy et al., 2015a] Edwards-Murphy, F., Magno, M., O'Leary, L., Troy, K., Whelan, P., and Popovici, E. (2015a). Big brother for bees (3b) - energy neutral platform for remote monitoring of beehive imagery and sound. In *2015 6th International Workshop on Advances in Sensors and Interfaces (IWASI)*, pages 106–111.

[Edwards-Murphy et al., 2015b] Edwards-Murphy, F., Magno, M., Whelan, P., and Vici, E. (2015b). B+wsn: Smart beehive for agriculture, environmental, and honey bee health monitoring - preliminary results and analysis. *SAS 2015 - 2015 IEEE Sensors Applications Symposium, Proceedings*.

[Edwards-Murphy et al., 2015c] Edwards-Murphy, F., Popovici, E., Whelan, P., and Magno, M. (2015c). Development of an heterogeneous wireless sensor network for instrumentation and analysis of beehives. *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, 2015:346–351.

[Edwards-Murphy et al., 2015d] Edwards-Murphy, F., Srbinovski, B., Magno, M., Popovici, E., and Whelan, P. (2015d). An automatic, wireless audio recording node for analysis of beehives. In *2015 26th Irish Signals and Systems Conference (ISSC)*.

[Eskov and Toboev, 2011] Eskov, E. and Toboev, V. (2011). Changes in the structure of sounds generated by bee colonies during sociotomy. *Entomological Review*, 91:347–353.

[Etxegarai-Legarreta and Sanchez-Famoso, 2022] Etxegarai-Legarreta, O. and Sanchez-Famoso, V. (2022). The role of beekeeping in the generation of goods and services: The interrelation between environmental, socioeconomic, and sociocultural utilities. *Agriculture*, 12(4).

[Ferrari et al., 2008] Ferrari, S., Silva, M., Guarino, M., and Berckmans, D. (2008). Monitoring of swarming sounds in bee hives for early detection of the swarming period. *Computers and Electronics in Agriculture*, 64:72–77.

[Fiedler et al., 2020] Fiedler, S., Zacepins, A., Kviesis, A., Komasilovs, V., Wakjira, K., Nawawi, M., Hensel, O., and Purnomo, D. (2020). Implementation of the precision beekeeping system for bee colony monitoring in indonesia and ethiopia. In *2020 21th International Carpathian Control Conference (ICCC)*, pages 1–6.

[Fitzgerald et al., 2015] Fitzgerald, D., Edwards-Murphy, F., Wright, W., Whelan, P., and Popovici, E. (2015). Design and development of a smart weighing scale for beehive monitoring. In *2015 26th Irish Signals and Systems Conference (ISSC)*, pages 1–6.

[FranceAgriMer, 2022] FranceAgriMer (2022). Observatoire de la production de miel et de gelée royale. https://www.franceagrimer.fr/Actualite/Filieres/Apiculture/2022/Retrouvez-la-synthese-de-l-etude-observatoire-de-la-production-de-miel-et-de-gelee-royale-donnees-2021. Accessed: 2023-05-05.

[Frings and Little, 1957] Frings, H. and Little, F. (1957). Reactions of honey bees in the hive to simple sounds. *Science*, 125(3238):122–122.

[Gabitov et al., 2022] Gabitov, I., Linenko, A., Yumaguzhin, F., Akchurin, S., and Valishin, D. (2022). The system of remote monitoring of microclimate parameters of bee colonies. *Journal of Ecological Engineering*, 23(1):264–273.

[Giammarini et al., 2015] Giammarini, M., Concettoni, E., Zazzarini, C., Orlandini, N., Albanesi, M., and Cristalli, C. (2015). Beehive lab project - sensorized hive for bee colonies life study. In *2015 12th International Workshop on Intelligent Solutions in Embedded Systems (WISES)*.

[Gil-Lebrero et al., 2017] Gil-Lebrero, S., Quiles-Latorre, F., Ortiz-López, M., Sánchez-Ruiz, V., Gàmiz-López, V., and Luna-Rodr'iguez, J. (2017). Honey bee colonies remote monitoring system. *Sensors*, 17(1).

[Giurfa et al., 2001] Giurfa, M., Zhang, S., Jenett, A., Menzel, R., and Srinivasan, M. (2001). The concepts of 'sameness' and 'difference' in an insect. *Nature*, 410:930–933.

[Gupta et al., 2021] Gupta, U., Kim, Y., Lee, S., Tse, J., Lee, H. S., Wei, G., Brooks, D., and Wu, C. (2021). Chasing carbon: The elusive environmental footprint of computing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 854–867, Los Alamitos, CA, USA. IEEE Computer Society.

[Hadjur et al., 2020] Hadjur, H., Ammar, D., and Lefèvre, L. (2020). Analysis of energy consumption in a precision beekeeping system. In *Proceedings of the 10th International Conference on the Internet of Things*, IoT '20, New York, NY, USA. Association for Computing Machinery.

[Hadjur et al., 2023] Hadjur, H., Ammar, D., and Lefèvre, L. (2023). EAPBSdata: Energy consumption and bee dataset of five connected beehives of one high season in France. https://doi.org/10.5281/zenodo.7880085. Available from mid-May 2023.

[Hansson, 1945] Hansson, Å. (1945). *Lauterzeugung und Lautauffassungsvermögen der Bienen*, volume 6. Entomologiska sällskapet i Lund.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

[Henry et al., 2019] Henry, E., Adamchuk, V., Stanhope, T., Buddle, C., and Rindlaub, N. (2019). Precision apiculture: Development of a wireless sensor network for honeybee hives. *Computers and Electronics in Agriculture*, 156:138 – 144.

[Hong et al., 2020] Hong, W., Xu, B., Chi, X., Cui, X., Yan, Y., and Li, T. (2020). Long-term and extensive monitoring for bee colonies based on internet of things. *IEEE Internet of Things Journal*, 7(8):7148–7155.

[Horvath et al., 2022] Horvath, T., Dániel, V., and Alexy, M. (2022). Beyond sensor data analysis: Unexpected challenges in a honeybee monitoring project. In *ITAT 2022*.

[Howard et al., 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861.

[Howard et al., 2018] Howard, D., Duran, O., and Hunter, G. (2018). A low-cost multimodal sensor network for the monitoring of honeybee colonies/hives. *Intelligent Environments*, pages 69–78.

[Howard et al., 2013] Howard, D., Duran, O., Hunter, G., and Stebel, K. (2013). Signal processing the acoustics of honeybees (apis mellifera) to identify the "queenless" state in hives. *Proceedings of the Institute of Acoustics*, 35:290–297.

[Hunter et al., 2019] Hunter, G., Howard, D., Gauvreau, S., Duran, O., and Busquets, R. (2019). Processing of multi-modal environmental signals recorded from a "smart" beehive. *Proceedings of the Institute of Acoustics*, 41:337–348.

[Kirchner, 1993] Kirchner, W. (1993). Acoustical communication in honeybees. *Apidologie*, 24:297–307.

[Komasilova et al., 2020] Komasilova, O., Komasilovs, V., Kviesis, A., Bumanis, N., Mellmann, H., and Zacepins, A. (2020). Model for the bee apiary location evaluation. *Agronomy Research*, 18:1350–1358.

[Kotovs and Zacepins, 2023] Kotovs, D. and Zacepins, A. (2023). Gis-based interactive map to improve scheduling beekeeping activities. *Agriculture*, 13(3).

[Kraemer et al., 2020] Kraemer, F. A., Palma, D., Braten, A. E., and Ammar, D. (2020). Operationalizing solar energy predictions for sustainable, autonomous iot device management. *IEEE Internet of Things Journal*, 7(12):11803–11814.

[Kridi et al., 2016] Kridi, D., Carvalho, C., and Gomes, D. (2016). Application of wireless sensor networks for beehive monitoring and in-hive thermal patterns detection. *Computers and Electronics in Agriculture*, 127:221–235.

[Kulyukin and Mukherjee, 2019] Kulyukin, V. and Mukherjee, S. (2019). On video analysis of omnidirectional bee traffic: Counting bee motions with motion detection and image classification. *Applied Sciences*, 9(18).

[Kulyukin et al., 2018] Kulyukin, V., Mukherjee, S., and Amlathe, P. (2018). Toward audio beehive monitoring: Deep learning vs. standard machine learning in classifying beehive audio samples. *Applied Sciences*, 8:1573.

[Kulyukin and Reka, 2016] Kulyukin, V. and Reka, S. (2016). Toward sustainable electronic beehive monitoring: Algorithms for omnidirectional bee counting from images and harmonic analysis of buzzing signals. *Engineering Letters*, 24:317–327.

[Kulyukin et al., 2023] Kulyukin, V. A., Coster, D., Tkachenko, A., Hornberger, D., and Kulyukin, A. V. (2023). Ambient electromagnetic radiation as a predictor of honey bee (apis mellifera) traffic in linear and non-linear regression: Numerical stability, physical time and energy efficiency. *Sensors*, 23(5).

[Kviesis et al., 2020] Kviesis, A., Komasilovs, V., Komasilova, O., and Zacepins, A. (2020). Application of fuzzy logic for honey bee colony state detection based on temperature data. *Biosystems Engineering*, 193:90–100.

[Kviesis et al., 2015] Kviesis, A., Zacepins, A., Durgun, M., and Tekin, S. (2015). Application of wireless sensor networks in precision apiculture. In *14th International Scientific Conference Engineering for Rural Development*.

[Langstroth, 2004] Langstroth, L. (2004). *Langstroth's Hive and the Honey-Bee: The Classic Beekeeper's Manual*. Dover Publications, Incorporated.

[Lettmann and Chauzat, 2018] Lettmann, M. and Chauzat, M.-P. (2018). Les outils connectés en apiculture : Evaluation de leurs application auprès des apiculteurs français. https://be.anses.fr/sites/default/files/O-028_2018-12-28_Outils-abeilles_Lettmann_VF.pdf. Accessed: 2023-05-05.

[Li et al., 2022] Li, L., Lu, C., Hong, W., Zhu, Y., Lu, Y., Wang, Y., Xu, B., and Liu, S. (2022). Analysis of temperature characteristics for overwintering bee colonies based on long-term monitoring data. *Computers and Electronics in Agriculture*, 198:107104.

[Magnier et al., 2018] Magnier, B., Ekszterowicz, G., Laurent, J., Rival, M., and Pfister, F. (2018). Bee hive traffic monitoring by tracking bee flight paths. In *13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, January 27-29, 2018, in Funchal, Madeira, Portugal*, pages 563–571.

[Marchal et al., 2020] Marchal, P., Buatois, A., and Kraus, S. e. a. (2020). Automated monitoring of bee behaviour using connected hives: Towards a computational apidology. *Apidologie*.

[Markovic et al., 2016] Markovic, D., Pesović, U., Djurasevic, S., and Randic, S. (2016). Decision support system for temperature monitoring in beehives. *Acta agriculturae Serbica*, 21:135–144.

[Martens and Grosse, 2015] Martens, J. and Grosse, R. (2015). Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2408–2417. JMLR.org.

[Meikle and Holst, 2015] Meikle, W. G. and Holst, N. (2015). Application of continuous monitoring of honeybee colonies. *Apidologie*, 46(1):10–22.

[Meikle et al., 2006] Meikle, W. G., Holst, N., Mercadier, G., Derouané, F., and James, R. R. (2006). Using balances linked to dataloggers to monitor honey bee colonies. *Journal of Apicultural Research*, 45(1):39–41.

[Meikle et al., 2016] Meikle, W. G., Weiss, M., and Stilwell, A. R. (2016). Monitoring colony phenology using within-day variability in continuous weight and temperature of honey bee hives. *Apidologie*, 47(1):1–14.

[Midtiby and Pastucha, 2022] Midtiby, H. S. and Pastucha, E. (2022). Pumpkin yield estimation using images from a uav. *Agronomy*, 12(4).

[Mnih et al., 2016] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937.

[Mukherjee and Kulyukin, 2020] Mukherjee, S. and Kulyukin, V. (2020). Application of digital particle image velocimetry to insect motion: Measurement of incoming, outgoing, and lateral honeybee traffic. *Applied Sciences*, 10(6).

[Murphy et al., 2016] Murphy, F. E., Magno, M., Whelan, P. M., O'Halloran, J., and Popovici, E. M. (2016). b+wsn: Smart beehive with preliminary decision tree analysis for agriculture and honey bee health monitoring. *Computers and Electronics in Agriculture*, 124:211–219.

[Murphy and Whelan, 2017] Murphy, F. E. and Whelan, P. M. (2017). Apisprotect. https://apisprotect.com/. Accessed: 2023-05-05.

[Ngo et al., 2021] Ngo, T. N., Rustia, D. J. A., Yang, E.-C., and Lin, T.-T. (2021). Automated monitoring and analyses of honey bee pollen foraging behavior using a deep learning-based imaging system. *Computers and Electronics in Agriculture*, 187:106239.

[Nolasco and Benetos, 2018a] Nolasco, I. and Benetos, E. (2018a). To bee or not to bee: An annotated dataset for beehive sound recognition. https://doi.org/10.5281/zenodo.1321278. Accessed: 2023-05-05.

[Nolasco and Benetos, 2018b] Nolasco, I. and Benetos, E. (2018b). To bee or not to bee: Investigating machine learning approaches for beehive sound recognition. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, pages 133–137.

[Nolasco et al., 2019] Nolasco, I., Terenzi, A., Cecchi, S., Orcioni, S., Bear, H. L., and Benetos, E. (2019). Audio-based identification of beehive states. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8256–8260.

[Ntawuzumunsi and Kumaran, 2019] Ntawuzumunsi, E. and Kumaran, S. (2019). Design and implementation of smart bees hiving & monitoring system. In *2019 IST-Africa Week Conference (IST-Africa)*, pages 1–9.

[Odemer, 2021] Odemer, R. (2021). Approaches, challenges and recent advances in automated bee counting devices: A review. *Annals of Applied Biology*, 180.

[Papachristoforou et al., 2008] Papachristoforou, A., Sueur, J., Rortais, A., Angelopoulos, S., Thrasyvoulou, A., and Arnold, G. (2008). High frequency sounds produced by cyprian honeybees apis mellifera cypria when confronting their predator, the oriental hornet vespa orientalis. *Apidologie*, 39(4):468–474.

[Pérez et al., 2016] Pérez, N., Jesús, F., Pérez, C., Niell, S., Draper, A., Obrusnik, N., Zinemanas, P., Spina, Y. M., Letelier, L. C., and Monzón, P. (2016). Continuous monitoring of beehives sound for environmental pollution control. *Ecological Engineering*, 90:326 – 330.

[Pignagnoli et al., 2023] Pignagnoli, A., Pignedoli, S., Carpana, E., Costa, C., and Dal Prà, A. (2023). Greenhouse gas (ghg) emissions from honey production: Two-year survey in italian beekeeping farms. *Animals*, 13(4).

[Prescott et al., 2023] Prescott, K., Kortman, S., Duque, J., Muramoto, J., Shennan, C., Greenstein, G., and Haffa, A. L. M. (2023). Analysis of trace volatile compounds emitted from flat ground and formed bed anaerobic soil disinfestation in strawberry field trials on california central coast. *Agronomy*, 13(5).

[Qandour et al., 2014] Qandour, A., Ahmad, I., Habibi, D., and Leppard, M. (2014). Remote beehive monitoring using acoustic signals. *Acoustics Australia / Australian Acoustical Society*, 42:204–209.

[Raffin et al., 2021] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8.

[Ramìrez et al., 2012] Ramìrez, M., Prendas, J. P., Travieso, C. M., Calderón, R., and Salas, O. (2012). Detection of the mite varroa destructor in honey bee cells by video sequence processing. In *2012 IEEE 16th International Conference on Intelligent Engineering Systems (INES)*, pages 103–108.

[Ramsey et al., 2017] Ramsey, M., Bencsik, M., and Newton, M. I. (2017). Long-term trends in the honeybee "whooping signal" revealed by automated detection. *PLOS ONE*, 12(2):e0171162.

[Ramsey et al., 2020] Ramsey, M.-T., Bencsik, M., Newton, M., Reyes, M., Pioz, M., Crauser, D., Simon-Delso, N., and Le Conte, Y. (2020). The prediction of swarming in honeybee colonies using vibrational spectra. *Scientific Reports*, 10.

[Reyes et al., 2012] Reyes, O. A. M., Àvila, A. A. M., Sebastian Eslava, G., and Rozo, G. B. (2012). Beekeeping monitoring module. In *2012 IEEE 4th Colombian Workshop on Circuits and Systems (CWCAS)*, pages 1–6.

[Rigakis et al., 2023] Rigakis, I., Potamitis, I., Tatlas, N.-A., Psirofonia, G., Tzagaraki, E., and Alissandrakis, E. (2023). A low-cost, low-power, multisensory device and multivariable time series prediction for beehive health monitoring. *Sensors*, 23(3).

[Robles-Guerrero et al., 2017] Robles-Guerrero, A., Saucedo-Anaya, T., González-Ramérez, E., and Galván-Tejada, C. E. (2017). Frequency analysis of honey bee buzz for automatic recognition of health status: A preliminary study. *Res. Comput. Sci.*, 142:89–98.

[Robles-Guerrero et al., 2023] Robles-Guerrero, A., Saucedo-Anaya, T., Guerrero-Mendez, C. A., Gómez-Jiménez, S., and Navarro-Solís, D. J. (2023). Comparative study of machine learning models for bee colony acoustic pattern classification on low computational resources. *Sensors*, 23(1).

[Robustillo et al., 2022] Robustillo, M. C., Pérez, C. J., and Parra, M. I. (2022). Predicting internal conditions of beehives using precision beekeeping. *Biosystems Engineering*, 221:19–29.

[Rybin et al., 2017] Rybin, V., Butusov, D., Karimov, T., Belkin, D., and Kozak, M. (2017). Embedded data acquisition system for beehive monitoring. In *2017 IEEE II International Conference on Control in Technical Systems (CTS)*, pages 387–390.

[Sakanovic and Kevric, 2020] Sakanovic, S. and Kevric, J. (2020). Habeetat: A novel monitoring platform for more efficient honey production. In Badnjevic, A., Škrbić, R., and Gurbeta Pokvić, L., editors, *CMBEBIH 2019*, pages 193–200, Cham. Springer International Publishing.

[Schulman et al., 2015] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France. PMLR.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

[Schurischuster et al., 2018] Schurischuster, S., Remeseiro, B., Radeva, P., and Kampel, M. (2018). A preliminary study of image analysis for parasite detection on honey bees. In Campilho, A., Karray, F., and ter Haar Romeny, B., editors, *Image Analysis and Recognition*, pages 465–473, Cham. Springer International Publishing.

[Schwartz et al., 2020] Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. (2020). Green ai. *Commun. ACM*, 63(12):54–63.

[Seritan et al., 2018] Seritan, G., Enache, B.-A., Florin, A., Adochiei, F., and Toader, S. (2018). Low cost platform for monitoring honey production and bees health. In *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pages 1–4.

[Shepherd et al., 2019] Shepherd, S., Hollands, G., Godley, V. C., Sharkh, S. M., Jackson, C. W., and Newland, P. L. (2019). Increased aggression and reduced aversive learning in honey bees exposed to extremely low frequency electromagnetic fields. *PLOS ONE*, 14(10):1–13.

[Shimasaki et al., 2018] Shimasaki, K., Jiang, M., Takaki, T., Ishii, I., and Yamamoto, K. (2018). Hfr-video-based honeybee activity sensing using pixel-level short-time fourier transform. *2018 IEEE SENSORS*, pages 1–4.

[Shimasaki et al., 2020] Shimasaki, K., Jiang, M., Takaki, T., Ishii, I., and Yamamoto, K. (2020). Hfr-video-based honeybee activity sensing. *IEEE Sensors Journal*, 20(10):5575–5587.

[Sledevic, 2018] Sledevic, T. (2018). The application of convolutional neural network for pollen bearing bee classification. *2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, pages 1–4.

[Sledevič et al., 2022] Sledevič, T., Serackis, A., and Plonis, D. (2022). Fpga implementation of a convolutional neural network and its application for pollen detection upon entrance to the beehive. *Agriculture*, 12(11).

[Soares et al., 2022] Soares, B. S., Luz, J. S., de Macêdo, V. F., e Silva, R. R. V., de Araújo, F. H. D., and Magalhães, D. M. V. (2022). Mfcc-based descriptor for bee queen presence detection. *Expert Systems with Applications*, 201:117104.

[Sparavigna, 2016] Sparavigna, A. C. (2016). Analysis of a natural honeycomb by means of an image segmentation. *Philica*, 2016(897).

[Stalidzans and Berzonis, 2013] Stalidzans, E. and Berzonis, A. (2013). Temperature changes above the upper hive body reveal the annual development periods of honey bee colonies. *Computers and Electronics in Agriculture*, 90:1 – 6.

[Stockwell et al., 1996] Stockwell, R. G., Mansinha, L., and Lowe, R. P. (1996). Localization of the complex spectrum: the s transform. *IEEE Transactions on Signal Processing*, 44(4):998–1001.

[Stojnic et al., 2018] Stojnic, V., Risojević, V., and Pilipovic, R. (2018). Detection of pollen bearing honey bees in hive entrance images. *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–4.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

[Tashakkori et al., 2021] Tashakkori, R., Hamza, A. S., and Crawford, M. B. (2021). Beemon: An iot-based beehive monitoring system. *Computers and Electronics in Agriculture*, 190:106427.

[Tashakkori et al., 2015] Tashakkori, R., Kae, D., and Parry, R. (2015). Automated beehive surveillance using computer vision. *Conference Proceedings - IEEE SOUTHEASTCON*, 2015.

[Terenzi et al., 2020] Terenzi, A., Cecchi, S., and Spinsante, S. (2020). On the importance of the sound emitted by honey bee hives. *Veterinary Sciences*, 7(4).

[Tolstov, 2012] Tolstov, G. (2012). *Fourier Series*. Dover Books on Mathematics. Dover Publications.

[Ulfa et al., 2022] Ulfa, F., Orton, T. G., Dang, Y. P., and Menzies, N. W. (2022). Developing and testing remote-sensing indices to represent within-field variation of wheat yields: Assessment of the variation explained by simple models. *Agronomy*, 12(2).

[Uthoff et al., 2023] Uthoff, C., Homsi, M. N., and von Bergen, M. (2023). Acoustic and vibration monitoring of honeybee colonies for beekeeping-relevant aspects of presence of queen bee and swarming. *Computers and Electronics in Agriculture*, 205:107589.

[vanEngelsdorp et al., 2009] vanEngelsdorp, D., Evans, J. D., Saegerman, C., Mullin, C., Haubruge, E., Nguyen, B. K., Frazier, M., Frazier, J., Cox-Foster, D., Chen, Y., Underwood, R., Tarpy, D. R., and Pettis, J. S. (2009). Colony collapse disorder: A descriptive study. *PLOS ONE*, 4(8):1–17.

[von Frisch et al., 1967] von Frisch, K., Chadwick, L., and Seeley, T. (1967). *The Dance Language and Orientation of Bees*. Belknap Press of Harvard University Press.

[Voudiotis et al., 2022] Voudiotis, G., Moraiti, A., and Kontogiannis, S. (2022). Deep learning beehive monitoring system for early detection of the varroa mite. *Signals*, 3(3):506–523.

[Wachowicz et al., 2022] Wachowicz, A., Pytlik, J., Małysiak-Mrozek, B., Tokarz, K., and Mrozek, D. (2022). Edge computing in iot–enabled honeybee monitoring for the detection of varroa destructor. *Int. J. Appl. Math. Comput. Sci.*, 32(3):355–369.

[Weber, 2013] Weber, E. (2013). Apis mellifera: The domestication and spread of european honey bees for agriculture in north america. *University of Michigan Undergraduate Research Journal*, 9:20–23.

[Wenner, 1964] Wenner, A. M. (1964). Sound communication in honeybees. *Scientific American*, 210(4):116–125.

[Wu et al., 2017] Wu, Y., Mansimov, E., Liao, S., Grosse, R., and Ba, J. (2017). Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5285–5294, Red Hook, NY, USA. Curran Associates Inc.

[Yang and Collins, 2018] Yang, C. and Collins, J. (2018). Improvement of honey bee tracking on 2d video with hough transform and kalman filter. *Journal of Signal Processing Systems*, 90.

[Yang and Collins, 2019] Yang, C. and Collins, J. (2019). Deep learning for pollen sac detection and measurement on honeybee monitoring video. In *2019 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6.

[Zacepins et al., 2015] Zacepins, A., Brusbardis, V., Meitalovs, J., and Stalidzans, E. (2015). Challenges in the development of precision beekeeping. *Biosystems Engineering*, 130.

[Zacepins et al., 2018] Zacepins, A., Jelinskis, J., Kviesis, A., Dzenis, M., Komasilovs, V., and Komasilova, O. (2018). Application of lorawan technology in precision beekeeping. *IX International Agricultural Symposium "Agrosym 2018"*.

[Zacepins and Karasha, 2013] Zacepins, A. and Karasha, T. (2013). Application of temperature measurements for bee colony monitoring: A review. *Engineering for Rural Development*.

[Zacepins et al., 2016a] Zacepins, A., Kviesis, A., Ahrendt, P., Richter, U., Tekin, S., and Durgun, M. (2016a). Beekeeping in the future — smart apiary management. In *2016 17th International Carpathian Control Conference (ICCC)*, pages 808–812.

[Zacepins et al., 2020] Zacepins, A., Kviesis, A., Komasilovs, V., and Muhammad, F. (2020). Monitoring system for remote bee colony state detection. *Baltic Journal of Modern Computing*, 8.

[Zacepins et al., 2016b] Zacepins, A., Kviesis, A., Stalidzans, E., Liepniece, M., and Meitalovs, J. (2016b). Remote detection of the swarming of honey bee colonies by single-point temperature monitoring. *Biosystems Engineering*, 148:76–80.

[Zacepins et al., 2022] Zacepins, A., Ozols, N., Kviesis, A.and Gailis, J., Komasilovs, V., Komasilova, O., and Zagorska, V. (2022). Evaluation of the honey bee colonies weight gain during the intensive foraging period. *Agronomy Research*, 20(2):457–548.

[Zacepins et al., 2012] Zacepins, A., Stalidzans, E., and Meitalovs, J. (2012). Application of information technologies in precision apiculture. In *Proceedings of the 13th International Conference on Precision Agriculture (ICPA 2012)*, Indianapolis, IN, USA.

[Zgank, 2018] Zgank, A. (2018). Acoustic monitoring and classification of bee swarm activity using mfcc feature extraction and hmm acoustic modeling. In *2018 ELEKTRO*, pages 1–4.

[Zhang et al., 2021] Zhang, T., Zmyslony, S., Nozdrenkov, S., Smith, M., and Hopkins, B. (2021). Semi-supervised audio representation learning for modeling beehive strengths. *CoRR*, abs/2105.10536.

[Zhu et al., 2019] Zhu, X., Wen, X., Zhou, S., Xu, X., Zhou, L., and Zhou, B. (2019). The temperature increase at one position in the colony can predict honey bee swarming (apis cerana). *Journal of Apicultural Research*, 58(4):489–491.

[Ziegler et al., 2022] Ziegler, C., Ueda, R. M., Sinigaglia, T., Kreimeier, F., and Souza, A. M. (2022). Correlation of climatic factors with the weight of an apis mellifera beehive. *Sustainability*, 14(9).