FIDUCIAL MARKER DETECTION AND IDENTIFICATION FOR INDIVIDUAL HONEY BEE TRACKING USING CASCADED DEEP LEARNING MODELS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUSTAFA YAVUZ KARA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

DECEMBER 2024

Approval of the thesis:

FIDUCIAL MARKER DETECTION AND IDENTIFICATION FOR INDIVIDUAL HONEY BEE TRACKING USING CASCADED DEEP LEARNING MODELS

submitted by **MUSTAFA YAVUZ KARA** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Assoc. Prof. Dr. Hande Alemdar
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Sinan Kalkan
Computer Engineering, METU _____

Assoc. Prof. Dr. Hande Alemdar
Computer Engineering, METU _____

Prof. Dr. Pınar Duygulu Şahin
Computer Engineering, Hacettepe University _____

Date:03.12.2024

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Mustafa Yavuz Kara

Signature :

ABSTRACT

FIDUCIAL MARKER DETECTION AND IDENTIFICATION FOR INDIVIDUAL HONEY BEE TRACKING USING CASCADED DEEP LEARNING MODELS

Kara, Mustafa Yavuz
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Hande Alemdar

December 2024, 62 pages

Performant and efficient detection and decoding of visual fiducial markers is an important task for tracking unique objects over long periods of time. These markers are often utilised by biologists to enable long-term analysis of their honey bee experiment hives and setups. Having a highly precise and time-efficient method for this task is especially important in context of automated monitoring, data recording and analysis of scientific experiments. In this thesis we propose a performant, efficient and flexible method of detecting and decoding fiducial markers in images and video frames. This method is structured as a two-stage marker detection and decoding pipeline made up of independent deep learning models that can be trained separately. We also describe and analyse a method for generating highly effective training datasets of varying size using a small set of source object and background images. Finally we demonstrate the effectiveness of our methods for fiducial detection and decoding tasks using real biological experiment data on *Apis mellifera* tagged with visual fiducial markers.

Keywords: Visual recognition, Object detection, Data generation, Visual fiducial

marker, Bee tracking

ÖZ

ARDIŞIK DERİN ÖĞRENME MODELLERİ KULLANARAK BİREYSEL ARI TAKİBİ İÇİN GÜVENİLİR İŞARETÇİ TESPİTİ VE ÇÖZÜMLENMESİ

Kara, Mustafa Yavuz
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Hande Alemdar

Aralık 2024 , 62 sayfa

Görsel referans işaretçilerinin yüksek başarımlı ve verimli şekilde tespit edilip çözümlenmesi, nesnelerin kendi kimlikleri ile birlikte takip edilmesinde önemli bir gereksinimdir. Bu işaretçiler biyologlar tarafından deney kovanları ve düzeneklerinin uzun süreli otomatik analizinde kulllanılmaktadır. Bu konuda yüksek kesinlikte sonuçlar sağlayan ve zaman açısından verimli yöntemlerin geliştirilmesi, özellikle bilimsel deneylerde otomatik olarak gözlem, veri kaydı ve analiz yapılabilmesi açısından büyük önem arzetmektedir. Bu tezde görsel işaretçilerin tespiti ve çözümlenmesi için yüksek başarımlı, verimli ve esnek bir yöntem önerilmektedir. Bu model iki aşamalı bir işaretçi tespit ve çözümleme sistemi olarak yapılandırılmış olup, her aşamanın ana elemanı sistemin kalanından bağımsız olan ve ayrı olarak eğitlebilen bir derin öğrenme modeli barındırmaktadır. Aynı zamanda bu derin öğrenme modellerinin öğrenmelerini üzerinde etkin olarak gerçekleştirebilecekleri, insanlar tarafından işaretlenmeye gerek duymadan küçük bir arka plan görüntü setinden ve işaretçilerden farklı büyüklüklerde veri setlerinin oluşturulması için bir yöntem de belirtilmiştir. Son olarak bu tezde anlatılan işaretçi tespit ve çözümleme yöntemlerinin ve oluşturulan

eğitim veri setlerinin etkinliği, iki boyutlu görsel işaretleme ve kimlik takip etme işaretçilerinin kullanılmış olduğu *Apis mellifera* (bal arısı) gerçek biyoloji deneylerinden alınan veriler üzerinde gösterilmiştir.

Anahtar Kelimeler: Görsel tanıma, Nesne tespiti, Veri oluşturma, Görsel referans işaretçişi, Arı takibi

To my family and my friends, whose endless support has made this possible

ACKNOWLEDGMENTS

First, I would like to acknowledge Assoc. Prof. Dr. Hande Alemdar for her continual support and feedback throughout the master's program. Her guidance was vital both for my work and my development as a researcher.

I would also like to thank my colleagues in our research group, our insightful discussions have always been an aid in my research work.

Finally, I would like to thank my family and my friends for their support and encouragement throughout this process.

This work was partially supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) through the project titled "Kovan 4.0: Bal Arısı (Apis Mellifera) İzleminde Yapay Zeka Temelli Nesnel Yaklaşımlar" (project no. 122E014).

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiv

LIST OF FIGURES xv

LIST OF ABBREVIATIONS xvii

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation 1

1.2 Problem Definition 4

1.3 Proposed Methods and Models 5

1.4 Contributions 5

1.5 The Outline of the Thesis 6

2 BACKGROUND AND RELATED WORK 7

2.1 Visual Fiducial Markers 7

2.1.1 Visual Fiducial Markers in Biology 9

2.2 Deep Learning in Computer Vision and Visual Recognition 11

2.2.1 Image Classification 11

2.2.2 Object Detection 12

2.3 Data Augmentation, Generation and Efficiency in Computer Vision . 13

2.4 Performance Metrics 14

3 GENERATING ARTIFICIAL DATA FOR DETECTION AND IDENTIFICATION 17

3.1 Challenges of Manually Annotating Marker Data with IDs 17

3.2 Data Generation Pipelines 19

3.2.1 Generating Object Detection Scenes 20

3.2.2 Generating Identification Images 23

4 FIDUCIAL DETECTION 31

4.1 A First Approach to Using an Object Detection model for Marker Detection 31

4.2 Using Separate Models that are Cascaded for Detection and Identification 32

4.3 Data and Experiment Design 33

4.3.1 Test Data 33

4.3.2 Source Data for Image Generation 35

4.4 Experiments on Honey Bee Behavioural Experiment Data with BEE-Tags 35

4.4.1 Single-Class Object Detection Experiments 35

4.4.2 Conventional Tag Detection Experiments 41

4.4.3 Single-Class Detection with Conventional Method as Classifier 43

4.5 Detection Results Summary 44

5 FIDUCIAL MARKER IDENTIFICATION 45

5.1 Pre-Identification Processing 45
5.2 The Identification Stage 47
5.3 Results and Discussion 48
5.4 Overall Pipeline Results 51
6 CONCLUSION 53
6.1 Limitations and Future Work 54
REFERENCES 55
APPENDICES 61
A The Conventional Tag Detection Method 61

LIST OF TABLES

TABLES

Table 4.1 Test set statistics 35

Table 4.2 Dataset sizes for training and validation sets 37

Table 4.3 Comparison of detection performances 42

Table 4.4 Comparison of detection performances 43

Table 5.1 Comparison of identification runtimes (ms) 49

Table 5.2 Comparison of all-average identification runtimes differences (s) . . 51

LIST OF FIGURES

FIGURES

Figure 1.1 Close-up images of honey bees getting tagged with markers before an experiment 2

Figure 2.1 A sample of visual fiducial markers from different fields. Clockwise from the left: BEEtag (behavioural ecology), ArUco (augmented reality), Circulatrix (biology), AprilTag (robotics) 8

Figure 2.2 An example of honey bees tagged with visual fiducial markers on their thoraxes 9

Figure 3.1 The BEEtag representing ID 25 19

Figure 3.2 Data generation pipeline overview for object detection 26

Figure 3.3 A marker image as it goes through the stages of processing before being placed in a scene 27

Figure 3.4 A sample of identification images generated by our method, in both low and high frequency backgrounds 28

Figure 3.5 High frequency background samples. Text serves excellently as a high frequency background as it provides a large variety of edge lengths and thicknesses, and unique curves and points. 29

Figure 3.6 A low frequency background sample 29

Figure 4.1 Overview of the BeeDet detection and identification pipeline . . 33

Figure 4.2 A collection of BEEtag markers taken from experiment footage. Complicating factors such as deformations, motion blur and occlusions can all bee seen 34

Figure 4.3 A sample test frame and the locations of markers within it 36

Figure 4.4 Bees in an experiment setup, with their markers detected 38

Figure 4.5 Graphs of metrics for test results on 1000×750 images. The colours blue, green, red, yellow, cyan represent the results for 2048, 3072, 4096, 6144, and 8192 training samples, respectively 39

Figure 4.6 Graphs of metrics for test results on 2000×1500 images. The colours blue, green, red, yellow, cyan represent the results for 2048, 3072, 4096, 6144, and 8192 training samples, respectively 40

Figure 5.1 Two images of the same tag detection, taken from images of two different resolutions. 46

Figure 5.2 Identification accuracy as compared to the previous method . . . 48

Figure 5.3 Average and total runtimes over the same test set for all three compared methods. 50

Figure 5.4 Full pipeline results 52

Figure 5.5 Full pipeline results 52

LIST OF ABBREVIATIONS

VFM	Visual Fiducial Marker
VF	Visual Fiducial
VR	Virtual Reality
AR	Augmented Reality
BEETag	Behavioural Ecology Tag
IoU	Intersection over Union
mAP	Mean Average Precision

CHAPTER 1

INTRODUCTION

1.1 Motivation

Insect species play a crucial role in almost all ecosystems on Earth, due to the roles they play in recycling nutrients, maintaining soil health and pollinating various plant species [1]. As a result, studying and monitoring these insects, both in their natural habitats and in controlled experiment environments, is an area that attracts continuous interest [2]. Honey bees are considered especially important for pollination, given that they pollinate many common food crops in agriculture, as well as some wild flower species [3]. This leads various bee species, both social and solitary, to be the focal point of many studies concerning insects and their role in wider ecosystems of the world.

Experimental setups for honey bees mostly require a human supervisor to continuously monitor the experiment and record any notable occurrence, as defined by the parameters of the experiment. While experiments monitored in this way are a mainstay in life sciences, the requirement for a human to watch the experiment, whether during the experiment of through a recording, can be a bottleneck for performing and analysing experiments. There exists some systems for automated monitoring and analysis of certain biological experiments on bees, but they often need specialised hardware that is costly to acquire and narrowly task-specific.

More recent approaches based on applying computer vision methods on experiment recordings has seen some success, especially ones that utilise deep learning models for detecting honey bees in images and videos [4]. Some of these methods can detect the presence and location of bees accurately, however, they cannot identify individual



(a) Honey bee has been tagged with a BEEtag marker by a biologist for an experiment



(b) Honey bee in an experiment mechanism, just before the experiment begins

Figure 1.1: Close-up images of honey bees getting tagged with markers before an experiment

honey bees. This necessitates tracking bees across all frames of a video in order to be able to propagate their identities, and even then any non-trivial breaks in the tracking means the identity of the bee must be reassigned, which makes using tracking to assign identities to bees a fragile method.

The most prominent method for assigning persistent identities to bees in biology literature is the use of visual fiducial markers. These markers are attached to the bees via a biologically suitable adhesive and are then decoded by computer vision systems from captured images and videos in order to determine the location and identities of individual bees. Figure 1.1 shows bees that has been marked with BEETags [5], which is a visual fiducial marker designed for use in behavioural ecology studies. These markers are traditionally detected and decoded using image processing methods, which are very sensitive to minor changes in the environment conditions, requiring lots of manual tuning and re-tuning to keep working under different conditions.

After the success that deep learning models have achieved in the field of computer vision, there have been efforts to apply the methods of deep learning to the detection and identification of visual fiducial markers. However, these methods often require large amounts of manually annotated training data in order to achieve high performance. Manual annotation of training data is often the rate-limiting step in training deep learning models specialised to the task at hand.

The main motivation of this thesis is to propose and analyse new methods for building visual fiducial marker detection models that can be utilised to build models of equivalent or superior performance, while requiring less or no more annotated data than other comparable models. To achieve this, we propose a multi stage pipeline for detecting and identifying visual fiducial markers. We also address the training data need for the models constituting our pipeline by proposing a data generation method that can create marker detection scenes as well as classification image samples. We generate this training data using only the original marker images and a small number of background images.

1.2 Problem Definition

The main problem that we attempt to tackle with the methods proposed in this thesis is detecting and identifying visual fiducial markers that are attached to honey bees. Biology experiment setups and beehives, which have been fitted with cameras and suitable lighting for monitoring, are our primary target environments. We aim to enable automated monitoring and analysis of both commercial hives used for honey production and smaller experimental setups used for hypothesis testing by biologists.

We present the problem definition for our thesis in two parts. The first part of our problem is the generation of artificial datasets suitable for training object detection and classification models. The second part of our problem is simply the detection and identification of visual fiducial markers. More formally, we define our problem in the standard object detection in supervised learning setting, in which there is a labelled dataset $\{(x_1, y_1), (x_2, y_2)...(x_N, y_N)\}$ of N examples, where:

- $x_i \in X$ are sample images
- $y_i \in Y$ are sets of object detections with the following components:
  - A quadruple (x, y, w, h) describing the location and dimensions of a bounding box
  - A class label c
  - A confidence score p for the detection

and the goal is to find the parameters of a mapping function $f_\theta : X \to Y$ such that a predetermined loss function $\mathcal{L}(\hat{y_i}, y_i)$ is minimised, which can be formulated as:

$$\theta^* \leftarrow \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(f_\theta(x_i), y_i) \tag{1.1}$$

where the loss function $\mathcal{L}$ is minimised when the object prediction $f_\theta(x_i) = \hat{y_i}$ and the true object description y_i are equal.

1.3 Proposed Methods and Models

To address the first part of our problem statement, we propose a general method of generating training data for models working on visual fiducial markers. This method encompasses both parts of the detection and identification problem. Only a small handful of background images and the images of the tags themselves are required in order to generate data. The details of our method of data generation is explained in Chapter 3. The results obtained by using data generated using this method is presented in Chapter 3 and Chapter 4.

For the second part of our problem statement, we propose a detection and identification pipeline for visual fiducial markers. This pipeline consists of cascaded deep learning models, where the two stages of the pipeline address the problems of detection and identification separately. The models that make up the two stages of our pipeline can be trained, and indeed replaced, independently of each other when needed. We use data that has been generated using the methods proposed in this thesis to train the models in our pipeline. Detection results that have been obtained from these models are presented in Chapter 4. Results for the identification model, as well as the results of the overall pipeline, are given in Chapter 5.

1.4 Contributions

The contributions of this thesis are as follows:

- We propose a two-part fiducial marker detection and identification pipeline, whose parts are structured so that they are independently re-trainable, making a flexible and performant method.
- We describe methods for generating datasets to be used with fiducial marker detectors, as well as generating marker images for use with identifiers.
- We present an end-to-end detection pipeline structure built using the generated data that can be used for fiducial localisation and identification.

1.5 The Outline of the Thesis

The thesis structure is as follows:

- In Chapter 2, we include background information on visual fiducial markers and their applications in biology; as well as methods for object detection, identification and data generation in computer vision.
- In Chapter 3 we discuss the reasons that make building a training data generation pipeline preferable to having the data manually annotated by humans, and introduce our methods for generating training data, for object detection as well as fiducial marker identification.
- In Chapter 4, we use the marker detector training data we have prepared in the previous chapter to train an object detection model for detecting visual fiducial markers on bees, and analyse the results that we obtain in terms of data suitability and model efficacy.
- In Chapter 5, we build an image classification model for use as a marker identifier, and measure its performance both in terms of task effectiveness and in terms of runtime. We also present the results that we have measured for our overall pipeline.
- In Chapter 6, we provide an overall summary of our work, its strengths, limitations and possible future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Visual Fiducial Markers

Visual Fiducial Markers (VFMs), sometimes simply called Visual Fiducials (VFs) are visually distinct markers that are embedded in the environment to aid automated digital systems to more easily calibrate to, perceive and/or interact with the real world around them. Adding a visual fiducial marker onto an object can be considered as adding an *artificial feature* to that object, in order to aid in its detection, classification, or tracking. They can often carry extra information embedded within them, which means they can also be used to encode additional information about the objects onto the visible surfaces of objects themselves. This information can include a unique ID for the object in question [6]. Though these markers can be of any size, shape and colour, black and white tags with a simple square or circular shape are often preferred, as not only are they much easier to build, they stand out better form their environment making them much easier to detect.

Visual fiducial markers are used across a wide range of fields and applications, and as such many different markers were designed and detection methods devised over time. Sample visual fiducial markers from various fields can be seen in Figure 2.1. One field where visual fiducial markers are often encountered are robotics, where the fiducial markers are used to simplify the process of recognising and interacting with the real world that surrounds a robot [7]. Visual fiducials is used to help robots localise themselves within their environments, which is a crucial step in planning how they should move to reach their end goals [8]. They have been applied extensively in warehouse management, where they would help robots identify inventory items, packages,

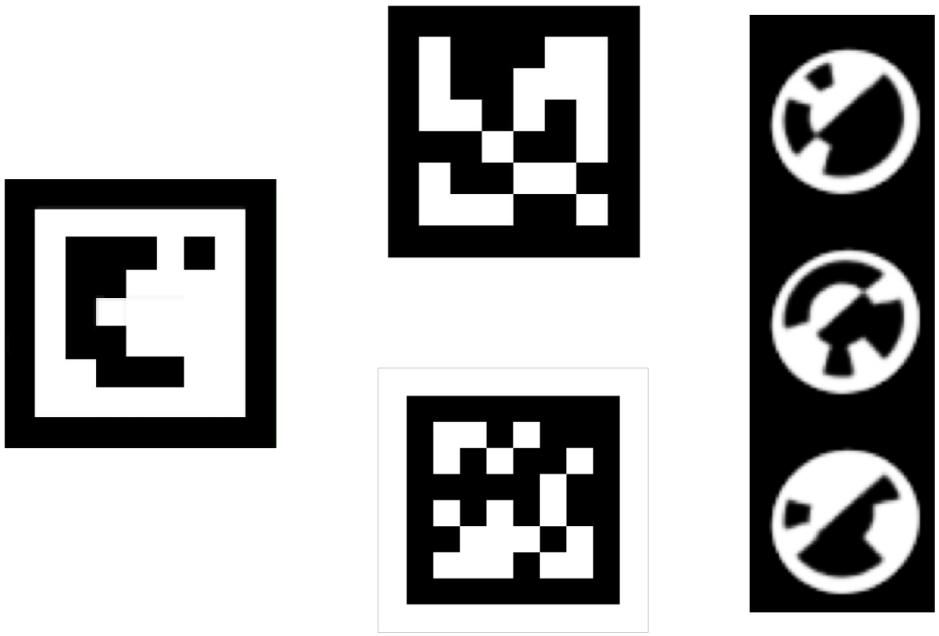


Figure 2.1: A sample of visual fiducial markers from different fields. Clockwise from the left: BEEtag (behavioural ecology), ArUco (augmented reality), Circulatrix (biology), AprilTag (robotics)

drop locations and even navigate the paths of the warehouse to efficiently reach their targets [9]. They are also used for improving the robots' ability to recognise certain objects in their environment and interact with them, follow moving objects through the environment, or help robots in multi-agent systems identify each other and follow, interact with or avoid one another as needed [10]. There have also been cases where newer detection methods with higher performance have been proposed for tags that had initially been presented with different detection methods [11].

Another area where visual fiducial markers are widely deployed are virtual reality (VR) and augmented reality (AR) setups [12]. Augmented reality systems that utilise markers for their operation are called marker-based augmented reality systems, as opposed to markerless augmented reality systems that attempt to detect relevant objects and positions without relying on markers to indicate them [13]. Marker-based augmented reality systems use these markers to indicate locations where virtual objects should be placed in the environment, and they may encode information about which object should be placed in their location [14]. Markers can also be used to indicate

surfaces in environments where additional details like writings and signs should be placed [15], or overlay some or all parts of objects with AR content [16].

2.1.1 Visual Fiducial Markers in Biology



Figure 2.2: An example of honey bees tagged with visual fiducial markers on their thoraxes

Visual fiducial markers has been used in the field of biology as well, owing to their ability to provide location, orientation and identity data on living specimens, whether under controlled experiment setups or marked and released wild individuals. They have also been widely applied to insect biology and behaviour for keeping track of behavioural changes on a per-individual basis. Some visual markers among those that had been discussed in Section 2.1 have been applied directly to individual insect tracking, despite having initially been designed with applications in robotics and VR/AR in mind. Mersch et al. have used the ARTags discussed previously to track ants within video recordings of experiments, and acquired precise information on

spatial and temporal distribution of the ants, from which they have obtained information on the rates of interactions between individuals and groups [17]. Such tags have also been used on honey bees, in order to assign unique identities to them that would persist even when the bees leave the hive to gather pollen and come back later [18]. This enabled the precise monitoring of different behaviours of honey bees, and track the differences that occur in these behaviours with as the bees move through different stages of their life cycles.

Even though it is possible to use markers created for robotics or VR/AR applications as-is for biological experiments on honey bees (or indeed any insect), there have been a number of markers that have been designed specifically for use on identifying animals, some for use on insects, and even some built with the express purpose of marking and detecting honey bees. The Circulatrix markers, which have been developed for the BeesBook project, is such a marker for detecting and identifying honey bees [6]. Similar to other visual fiducial markers, it contains special borders to help with the detection of the marker, and special regions for encoding the identity of the marker and determining its orientation. However, unlike other visual fiducial markers that have been discussed so far, special consideration has been given to how the markers would fit on a bee's thorax to ensure that they would not impede the movements, and especially flight, of the bees. They are also designed to be slightly bent while they are being punched out, so that they fit well onto the thorax of the bee, and has a low probability of falling off of the bees during the experiments. Another instance of such markers is the bCode marker, developed for monitoring social dynamics of honey bee colonies [19].

There also exist systems that are built by defining a new detection and identification procedure over and existing marker design. The circulatrix markers from BeesBook are one such system, for which a deep convolutional neural network approach has been proposed after their publication [20]. This new approach improves both localisation and identification performance of these tags. Other approaches have focused on integrating advancements from various newly developed models into the detectors for older tags, to improve performance while also extending the capabilities of the older tag system [21]. In this instance a pose estimation system is added to the model to get estimated poses of all bees, as well as their id's are known [22]. The overall

system that result from this is highly capable for pose estimation, marker detection and marker identification.

One other visual fiducial marker designed for and used on honey bees are the BEEtags [23], which are also the markers that has been used in the experiments of this thesis. BEEtag's are square tags containing a 5×5 coding region that can represent 25 bits overall and a 2-unit wide border region. Of the 25 bits of total usable capacity, 10 bits have been set aside for encoding error detection and correction codes. There is also a defined minimum Hamming distance requirements for the tags in a set to be considered valid, increasing the robustness. Note that this minimum distance is open to manual adjustment, depending on the required number of tags and the desired sensitivity of the system. An example scene of honey bees marked with BEEtags in an experimental setup is given in Figure 2.2.

2.2 Deep Learning in Computer Vision and Visual Recognition

The field of computer vision encompasses a wide variety of tasks ranging from image classification and semantic segmentation to scene reconstruction and motion analysis. While a large number of techniques have been developed for each computer vision task within the history of the field, almost all of the most successful results in recent years have been obtained by applications of machine learning, and more specifically deep learning techniques [24] [25] [26]. This trend is consistent with the increasing prevalence of machine learning methods in other sub-fields of computer science [27].

2.2.1 Image Classification

The task of image classification involves a given image to a class. The output is only the class assignments of images, indicating the exact location of the objects within the images is not part of this task, so no locations, bounding boxes are segmentation masks are included in model outputs. In general images are not assigned to multiple classes, so the only output per input image is a single class assignment.

After the slow progress of performance on classification tasks using classical im-

age processing methods, a breakthrough in performance came in Krizhevsky et al. [28], in which their deep convolutional neural network method (which would later be referred to as AlexNet) for classification demonstrated a significant improvement in performance over other contemporary state-of-the-art methods in ImageNet Large Scale Visual Recognition Challenge [29]. This paved the way for increasing use of deep neural networks for visual recognition tasks [30]. Some of the best performing models of their time tended to be increasingly wider and deeper networks [31].

2.2.2 Object Detection

The object detection task is differentiated from image classification by its requirement that location information (often in the form of a bounding box) on detected objects are also provided with object classes. Also, in most modern object detection tasks there is often more, sometimes significantly more, than one object to be detected per image.

A large variety of methods for object detection have been proposed over the years, starting with a plethora of different hand crafted features and transform for specific objects and tasks, most of which have fallen out of favour today. Then feature descriptors that can be used on a wide variety of images have been proposed, which include the famous Scale Invariant Feature Transform (SIFT) by Lowe (1999) [32] and Histogram of Oriented Gradients (HOG) by Dalal & Triggs (2005) [33]. While these methods are no longer in widespread use for object detection on their own, they are still used for various supporting tasks in the field of computer vision.

The most impactful change in the last decades in object detection, and in fact all of computer vision has doubtless been the adoption of deep learning techniques in computer vision. An early and particularly influential model for object detection has been R-CNN by Girshick et al. [34] This model has provided large improvements over conventional object detection methods. This model has what is often called a two-stage architecture. Another significant development was the advent of single-stage object detection models, like the so-called YOLO [35]. For some time these models, their immediate successors, and other models derived from these two were considered the strongest choices for the object detection tasks, and is in fact still very

widely used model for object detection tasks in all sorter of context. Finally, recently newer models and methods have been explored in the context of computer vision. More recent models like [36] incorporate newer techniques into themselves, leading to some even more promising results [37]. And some of the latest models have been inspired by the success of models that are in other sub-fields of the deep learning fields, such as NLP. A famous example of a transfer of a method from a another sub-field is the Vision Transformers [38]. There have also been efforts to improve the usefulness of visual fiducials in other fields [39], which if successful would lead to even faster advancements in the future.

2.3 Data Augmentation, Generation and Efficiency in Computer Vision

Due to the cost, in terms of human labour, of acquiring and labelling a large dataset of images, there have almost always been great interest about data efficiency in computer vision. There are various approaches to finding solutions that can improve the data efficiency of these models.

A very common approach is to use **transfer learning**, that is, utilising a model pre-trained on a large, general dataset and only doing fine-tuning for the task at hand with a smaller task-specific dataset. A model trained in this way can achieve high performance thanks to the extensive internal representations acquired during the pre-training, however, if there is a significant enough distance between the pretraining data and the task for which fine-tuning should occur, the model may fail to learn anything substantial.

A second, somewhat related approach is **semi-supervised learning**. This approach combines some labelled data with large amounts of unlabelled data and tries to utilise the labelled data to be able to take advantage of the unlabelled data. This is often done by attempting to generate labels for the unlabelled data using the limited amount of labelled data at hand. Another approach called active learning works on an already working (even if terribly) model and attempts to pick the most informative examples based on the model output for humans to annotate.

A very common approach to address the limited availability of human annotated data

is to use various **data augmentation** techniques to make the most of the available data. These augmentations are often in the form of scaling, rotating, cropping and partially occluding or other similar transformations. These transformation can boost the amount of available data while also increasing model robustness to rotations etc.

Another approach is to rather than augment available data, create data from scratch using various techniques such as overlaying images, physics simulations, object models of 3D rendering, which is frequently called **synthetic data generation** . This greatly alleviates the need for human annotated data, but also creates the risk that whatever environment is available in the generation inputs may not accurately represent their target domain.

2.4 Performance Metrics

It is important to have a good performance metric in order to be able to accurately evaluate various models against each other. There exist various metrics for comparing the results of computer vision models in general and object detection in particular. The most prominent among these are:

- **Intersection over Union (IoU):** IoU is considered a good measure of how well an object is detected in an image. The IoU between two bounding boxes in an image is simply calculated by first calculating the ares of the intersecting region of the two images, and then calculating the union (which can be calculated extremely simply from the areas of the two bounding boxes and the area of the intersection). IoU ranges between 0 and 1, where 0 means completely disjoint bounding boxes and 1 means a perfectly located, perfectly sized and perfectly aligned bounding box.
- **Precision:** Precision is a common metric throughout all of machine learning, and is simply defined as the ratio of true positive predictions of the model to the all predictions made by the model. It is a good measure of the task-specificness of a model, for example, we can be fairly sure of the information we can get from a model with a high precision value. It can be used fairly straightforwardly for object detection as well, the only point that needs consideration is the cri-

teria for a detection being true or false. To be able calculate the precision from the results of a system, we first need to consider what we to be a successful/true detection. Most often, we define the criteria for success as surpassing some IoU threshold in the predictions, and any predictions that have lower overlap with their underlying true object are considered failed detections. This is of course accompanied by the condition that the predicted class must be the same as that of the object's. In object detection tasks, mean average precision **(mAP)** is often preferred to raw precision values.

- **Recall:** The definition of recall is similar to that of the precision, it is defined as the ration of the true positive predictions to the total number of predictions made. This measures whether our method can comprehensively predict all the objects in the image. Note that precision and recall are often at odds with each other, and attempts to improve one run the risk of inadvertently diminishing the other.

- **F1 Score:** F1 score was developed as a way to combine the information provided by precision and recall into a single metric that can be indicative of the model's overall performance. It is defined as:

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{2.1}$$

  where P denotes precision and R denotes recall, as defined previously.

- **Mean Average Precision (mAP):** Mean Average Precision is a metric commonly used to evaluate the performance of object detection and image retrieval models in computer vision. It provides a way to measure how well a model is able to correctly identify objects in an image and rank them in the correct order.

  Average Precision (AP): For each class, the Average Precision is calculated by considering the precision and recall at different levels of confidence (thresholds). It is given by the formula:

$$AP_i = \frac{1}{N} \sum_{n=1}^{N} P_n \tag{2.2}$$

  where N is the number of different recall thresholds considered, and P_n is the precision for a given recall threshold.

Having calculated the Average Precision scores, mAP is the mean of the Average Precision scores across all classes. mAP provides an overall measure of how well the model performs across all classes. It is defined as:

$$mAP = \frac{1}{C} \sum_{i=1}^{C} AP_i \tag{2.3}$$

where C denotes the number of classes.

CHAPTER 3

GENERATING ARTIFICIAL DATA FOR DETECTION AND IDENTIFICATION

In this chapter we explain the data generation methods that were utilised for the artificial datasets that we have used for the training of our BEEtag detection and identification models. In Section 3.1 we start with the reasons we consider artificial generation (as a replacement for or in supplementation of human-labelling of relevant data) necessary, as well as why the case of detecting and identifying BEEtags is uniquely amenable to using artificially generated training data. Section 3.2 details the procedures for generating the training data. We start with the data for our object detection model, the output of which also serves as the starting point of the data for our BEEtag identification model. We then explain the additional processing that we perform in order to generate the training data of our identification model. Finally in Section 3.3 we analyse the results of our data generation pipelines, and compare the properties of the resulting images. We also propose some post-processing steps to the generated data based on the results of these analyses.

3.1 Challenges of Manually Annotating Marker Data with IDs

Getting high quality human labelled visual data is an important part of any computer vision task that is approached through machine learning methods. Acquiring this data and getting it reliably labelled is therefore crucial to the successfully developing a computer vision model, but is also costly and time-consuming for model builders. These problems are compounded by the nature and unique visual properties of BEEtag markers (and of most fiducial markers in general, and especially of those that is

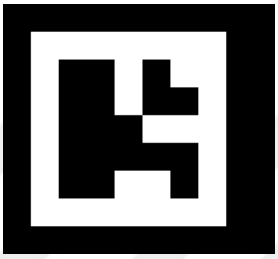
able to represent a large set of distinct IDs) as target objects for detection and identification. This is caused by the fact that representing a significant number of bits of information on a physical object, to be recovered later from images and videos of the object, must often rely on a black-and-white region encoding of the data into the tag in order to be robust to lighting changes, physical deformations and camera artifacts such as motion blurring and lens distortions. The most notable, and the most problematic, problem arising from these is the difficulty of distinguishing markers by sight (this is also a crucial factor for developing a model to detect them, as will be discussed in detail in later chapters). This makes it so that it takes a great deal longer, and cost quite a bit more, to have the data labelled by human labellers, as well as increasing the error rate in human-labelled datasets, which can have significant adverse effects on the training outcomes of models built using such data.

Various techniques for getting the most out of limited amounts of high quality human labelled data exists. These include transfer learning techniques like pre-training, few-shot and semi-supervised learning, some ensemble methods, active learning, and data augmentation and synthetic generation methods. Synthetic generation models often utilise generative model such as generative adversarial networks (GANs) or rely on simulated environments, like used for many applications in robotics. While these methods can deliver great results under the right circumstances, they often have significant data and computational requirements of their own. We use a method of direct image combination that is much simpler and faster, while still retaining the ability to produce visual data that is useful for the task of training models for the tasks of detection and identification of BEEtags. This is made possible by taking advantage of unique properties of visual markers, such as BEEtags, as objects present in scenes.

One such property is the fact that each tag is an artificially generated object with an exact visual specification. This means that the original images of these tags can be considered a 'canonical visual representation' or a 'master copy' for all possible visual appearances of these tags, under all possible conditions. This implies that in principle, any possible real-world appearance of these tags, no matter the camera angle, lighting, noise, deformations or distortions present, can be arrived at by applying a suitable set of image transformations to the original image of that tag. We can also see that these transformations can be constructed from a small set of standard and

simple image transforms. To illustrate how this is different from the general case for object detection, consider an everyday object that may be a target class for an object detection model, such as chair. This single class encompasses objects of a wide variety of possible distinct forms, textures and colours. It is clear that there is no single unique image can be used as a canonical representation of all chairs, that can be used to arrive at various possible visual appearances of all chairs in different environments. Contrast that with a visual marker, for example the BEEtag that represents the ID 25.

Figure 3.1: The BEEtag representing ID 25



Another property that makes generating useful samples of images containing tags by simple combination of images possible is the fact that the tags themselves are designed to stand out from the general environment in which they appear. While an image of any other object that is removed from its original background and moved onto a new one is likely to be a poor representation of how that object would appear if it was originally in the second scene, this distinction is less problematic for visual markers. As such, we can build a less data- and compute-intensive pipeline that can still generate images of BEEtags in various backgrounds representative enough for use as training data in building detection and identification models for these tags.

3.2 Data Generation Pipelines

As we use separate detection and classification models to build our overall model, we generate two sets of training data: the first one is for the initial detection task, with object images placed in backgrounds; the second one is for the later classification of the detected objects, consisting solely of object images under various transforms and modifications.

Here we present the data generation methods that we have used for generating training data for the tag localisation and identification steps of our pipeline. Each of the two steps requires different training data to be built. For the first step of tag localisation, we generate a set of images that consist of BEEtags placed on various different backgrounds. For the second stage of identification, we generate a set of tag images that resemble images of tags cropped using the detections of the localisation step. The outputs of these methods are then used as training data for each step of our BEEtag detection and identification pipeline.

3.2.1 Generating Object Detection Scenes

We use a simple strategy for building scenes for the object detection, which can be split into four steps: selection of object and background images, processing of selected images, overlaying of the object images onto the background image, and processing the final image again before the image is ready to use. An overview of the process for generating object detection scenes is laid out in Figure 3.2. The details of this process, as well as the various parameters used for generating the scenes, is described below.

Our algorithm takes the following parameters:

- N, total number of images to be generated
- (w_b, h_b), width and height of output images
- (w_o, h_o), width and height of overlaid objects
- o_{min}, minimum number of objects per image
- o_{max}, maximum number of objects per image

To facilitate blending overlaid object images with the backgrounds, we also separate our background images into two groups, one high-frequency group and one low-frequency group, depending on the level of sharp detail they contain. We take advantage of this distinction during the image generation.

The procedure for generating the images are as follows:

Given a set of n background images and m object images:

1. Pick source images
   (a) Pick one background image randomly from among n, with uniform probabilities.
   (b) If the background image is smaller than the desired output size:
      i. If the background image is low frequency, resize the image to output dimensions (w_b, h_b).
      ii. If the background image is high frequency, repeat the image horizontally and vertically to create a tiled image at least as large as the output size. Take a random (w_b, h_b) crop of the selected image, to be used as the background.
   (c) Otherwise, take a random (w_b, h_b) crop of the selected image, to be used as the background.
   (d) Pick the number of objects $o_{min} \leq o_p \leq o_{max}$ to be placed in the background image, with uniform probabilities.
   (e) Pick o_p ***unique*** object images from among m, with uniform probabilities.
2. Process selected source images
   (a) Scale each object image to the dimensions (sw_o, sh_o), where $s \in [0.8, 1.2]$ is a scale factor determined randomly for each selected object image.
   (b) Apply Gaussian blur with a 3×3 kernel and $\sigma = 0.85$ to each object image.
   (c) Add Gaussian noise with $\mu = 0$ and $\sigma^2 = 1000$ to each object image.
   (d) Normalise all pixel values back to the range $[0, 255]$.
   (e) Apply a random perspective transform of scale $s_p \in [0.1, 0.5]$ on each object with probability $p_p = 0.5$, where the scale s_p denotes the perspective shift the image will experience relative to its width and height.
   (f) Rotate each object image by α degrees on the image plane, where α is discrete uniform with range $0 \leq \alpha < 360$, and is resampled for each object image (note that this may result in the object image size being larger than (w_o, h_o)).

3. Overlay objects onto backgrounds

   (a) Take $x \in [\frac{w_f}{2}, w_b - \frac{w_f}{2}]$ and $y \in [\frac{h_f}{2}, h_b - \frac{h_f}{2}]$ for each object, where w_f and h_f are the final width and height of the object image after rotation.

   (b) Overlay the object image centred on (x, y) with 100% opacity over the background image.

4. Process the final image

   (a) Apply Gaussian blur to the resulting image, with a 7×7 kernel and $\sigma = 1.44$.

Note that the steps 2.b, 2.c and 2.d are performed only when the chosen background image is from the low-frequency background images group. For background images in the high-frequency group, step 2.a is followed directly by step 2.e, skipping over 2.b, 2.c and 2.d.

The procedure described above is applied N times to generate N sample images. All random numbers and selections are regenerated for each sample.

Along with the image files, we also generate text files containing the annotations for the objects placed within the images. For use in these annotations, we randomly assign integer ID's to each object, from the set of sequential integers 1 to m inclusive, where m is again the total number of objects. Using these ID's we then create 3 sets of files that describe the makeup of the generated images:

- A single identity mappings file, containing pairs of IDs and their assigned objects
- N annotation files, one per image, that contain the identities and locations of objects in the corresponding images. These files contain one object annotation per line, and the annotation lines are of the form $i\ x\ y\ w\ h$, where
  - i is the ID of the object
  - x, y describe the location of the *centre* of the object
  - w, h are the width and the height of the object in the image

x, y, w, h are all normalised to be in the range $[0, 1]$ using the dimensions of the image.

- N secondary annotation files of the same form as the first ones, where all objects are assigned the same object ID (essentially a single-class annotation).

3.2.2 Generating Identification Images

In order to generate images that are similar enough to what the output of the detector would be on real-world samples, we start the identification sample generation process in a very similar way to the detection data generation methods, by applying tag images on background images, although with some minor changes. We then crop the tag images from the backgrounds that they've been placed onto. This ensures a realistic background on the fringes and corners of the tag images we generate for the tag identifier. The full process for generating tag identifier training images is given below.

The algorithm takes the following parameters:

- N, total number of images to be generated per class
- s_o, the scaling constant for the objects

Similar to the previous method, we separate our image into high- and low-frequency groups, and whether steps 2.b, 2.c and 2.d are performed is dependent on the group of the chosen background image in the same way as the object detection method.

Given a set of n background images and m object images:

1. Pick source images
   (a) Pick one background image randomly from among n, with uniform probabilities.
   (b) Take all m object images to create one sample of each class
   (c) Scale all object images by s_o, so that their dimensions are now $(s_o w_o, s_o h_o)$, where the previous dimensions were (w_o, h_o).

2. Process selected source images

   (a) Scale each object image by s so that their new dimensions are $(ss_o w_o, ss_o h_o)$, where $s \in [0.8, 1.2]$ is a scale factor determined randomly for each selected object image.

   (b) Apply Gaussian blur with a 3×3 kernel and $\sigma = 0.85$ to each object image.

   (c) Add Gaussian noise with $\mu = 0$ and $\sigma^2 = 1000$ to each object image.

   (d) Normalise all pixel values back to the range $[0, 255]$.

   (e) Apply a random perspective transform of scale $s_p \in [0, 0.6]$ on each object with probability $p_p = 0.6$, where the scale s_p denotes the perspective shift the image will experience relative to its width and height.

   (f) Rotate each object image by α degrees on the image plane, where α is discrete uniform with range $0 \leq \alpha < 360$, and is resampled for each object image (note that this may result in the object image size being larger than $(ss_o w_o, ss_o h_o)$).

3. Overlay objects onto backgrounds

   (a) Take $x \in [\frac{w_f}{2}, w_b - \frac{w_f}{2}]$ and $y \in [\frac{h_f}{2}, h_b - \frac{h_f}{2}]$ for each object, where w_f and h_f are the final width and height of the object image after rotation and w_b, h_b are the width and height of the background image, respectively.

   (b) Overlay the object image centred on (x, y) with 100% opacity over the background image.

4. Process the final image

   (a) Apply Gaussian blur to the resulting image, with a 7×7 kernel and $\sigma = 1.44$.

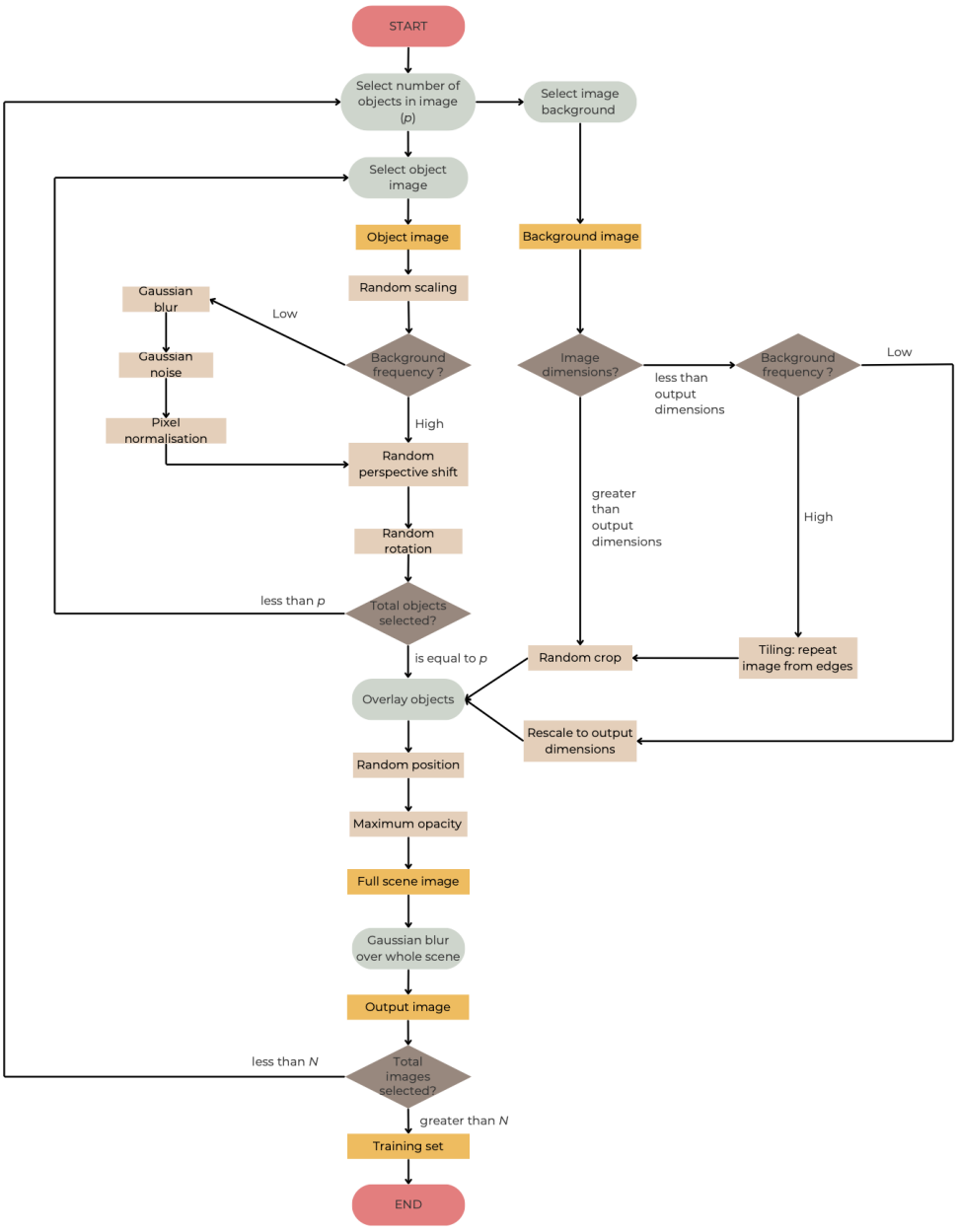
5. Recover the object images from the final image

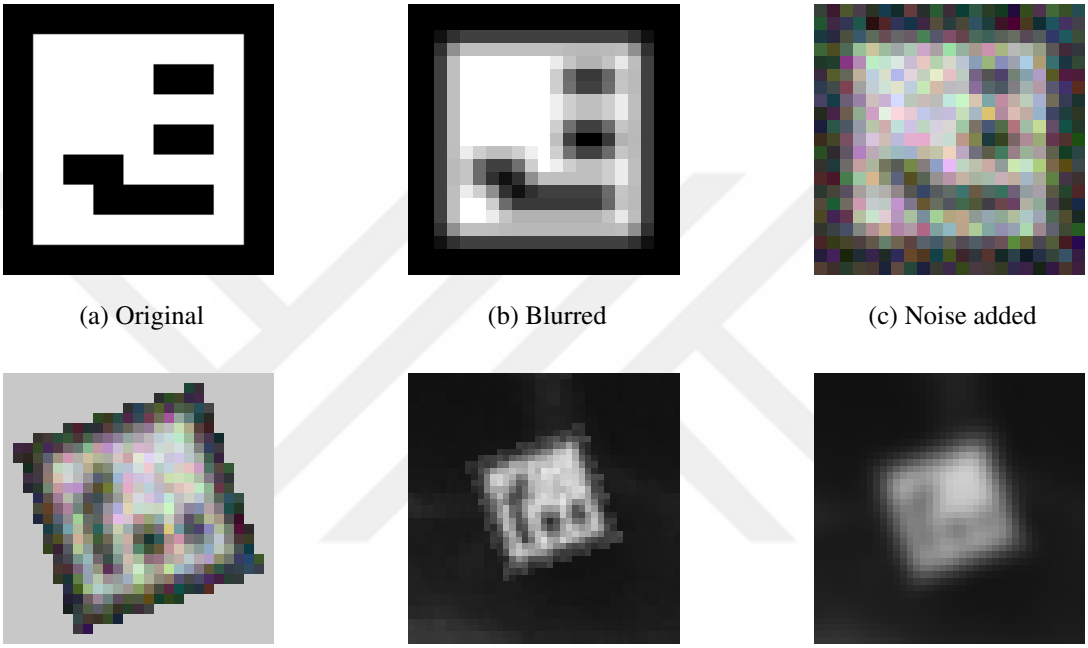
   (a) Take a crop of the final image centred on $(x + r_x, y + r_y)$ of width $w_f + r_w$ and height $h_f + r_h$, where $r_x \in [-0.1, 0.1]$, $r_y \in [-0.1, 0.1]$, $r_w \in [-0.05, 0.15]$ and $r_h \in [-0.05, 0.15]$ are random adjustments to the location and dimensions of the final crop.

The procedure described above is applied N times to generate a dataset with N samples per class. All random numbers and selections are regenerated for each sample.

The output is organised into a folder hierarchy that reflects the classes of the generated objects. One folder is created for each different class (that is to say, BEEtag ID), where all samples of this class is written. No additional annotation files are output at the end of the algorithm.

Figure 3.2: Data generation pipeline overview for object detection





(a) Original (b) Blurred (c) Noise added

(d) Rotated (e) Placed in background (f) After final blur

Figure 3.3: A marker image as it goes through the stages of processing before being placed in a scene

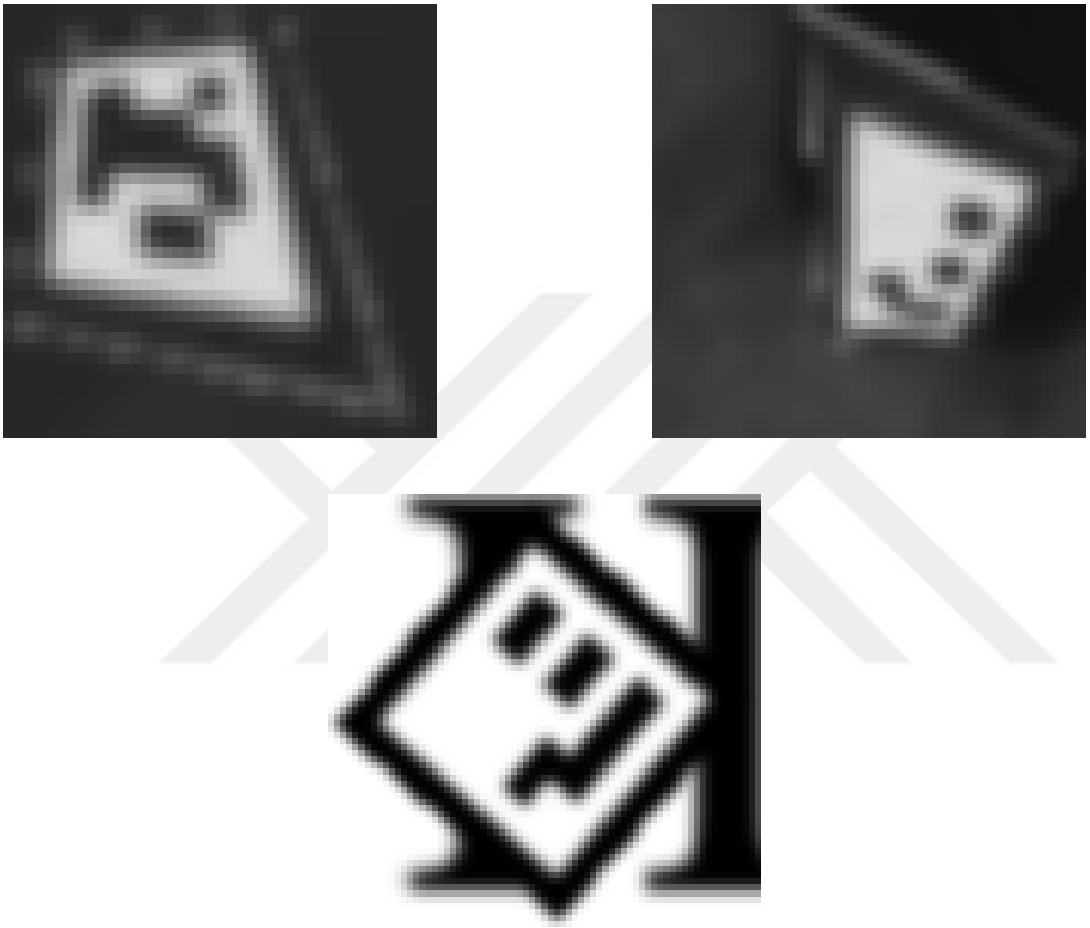


Figure 3.4: A sample of identification images generated by our method, in both low and high frequency backgrounds

rocess in $S_x(R)$. They define the following inhomogeneous process $\{N(t,x), 0 \leq$
$\leq R\}$,

$$N(t,x) = \sum_{i=1}^{n} 1\{X_i \in S_x(t)\}, \tag{4.1}$$

/hich counts observations within distance t from fixed point x. MLE approximates
is binomial process by a Poisson process and suppresses the dependence on x, where
e rate $\lambda(t)$ of the process $N(t)$ is then

$$\lambda(t) = f(x)V(m)mt^{t-1}. \tag{4.2}$$

etting $\theta = \log f(x)$, they obtain the log-likelihood of the observed process $N(t)$:

$$L(m,\theta) = \int_0^R \log \lambda(t) dN(t) - \int_0^R \lambda(t)dt. \tag{4.3}$$

n particular, the MLEs must satisfy the following likelihood equations below:

$$\frac{\partial L}{\partial \theta} = \int_0^R dN(t) - \int_0^R \lambda(t)dt = N(R) - e^{\theta}V(m)R^m = 0, \tag{4.4}$$

$$\begin{aligned}\frac{\partial L}{\partial m} &= \left(\frac{1}{m} + \frac{V'(m)}{V(m)}\right) N(R) + \int_0^R \log(t) dN(t) \\ &\quad - e^{\theta}V(m)R^m \left(\log(R) + \frac{V'(m)}{V(m)}\right) \\ &= 0.\end{aligned} \tag{4.5}$$

Figure 4.1 Point-wise ID estimations of MLE [4] on mixtures of Gaussian distributions. (a) Equal mixture of two Gaussian distributions. (b) Mixture for two Gaussian distributions with changing ratio of sample contribution. Here, a dataset with a ratio of 0.0 only includes samples from the first component, 1.0 only includes samples from the second component, and 0.5 means equal contribution of samples from both components. 33

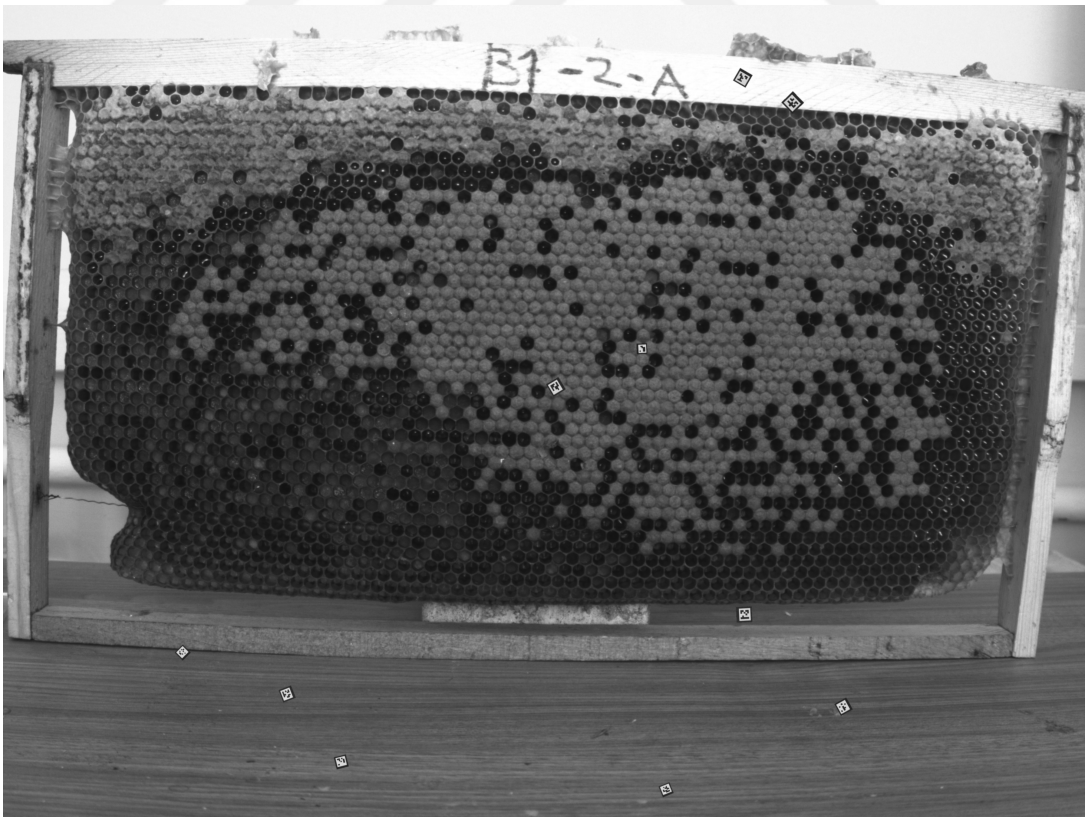
Figure 4.2 Estimating the ID of CIFAR-10 class categories with MLE [4] (a) Using the isolated manifold of each class category. (b) Using the manifold created by the entire dataset. 34

Figure 4.3 ID estimates of MLE [4] on a simple Gaussian distribution with an ID of 5, in a space with extrinsic dimensionality of 10. We use the unbiased MLE estimator of [5], highlighted in blue. 35

Figure 4.4 ID estimates of MLE [4] for synthetic Gaussian data and their expected values. 36

Figure 3.5: High frequency background samples. Text serves excellently as a high frequency background as it provides a large variety of edge lengths and thicknesses, and unique curves and points.

Figure 3.6: A low frequency background sample



CHAPTER 4

FIDUCIAL DETECTION

In this chapter, we focus the methods and models we have considered for the detection of BEEtag objects, with or without identification. In Section 4.1 we describe the initial method of treating each different tag identity as a different class of object, and the difficulties arising from such an approach due to the visual features of the BEEtags. Section 4.2 details our approach to using object detection methods for locating BEEtags in a given image, which includes deferring BEEtag identification to later stages in the pipeline. In Section 4.3 we explain the experimental setup and data acquisition methodology, as well as the datasets that will be used for testing our models. Section 4.4 presents the test results we have acquired from all approaches considered so far and provide comparisons of performance and runtime characteristics of each method. Finally, Section 4.5 concludes this chapter with an overall summary of our results so far.

4.1 A First Approach to Using an Object Detection model for Marker Detection

One method that can be used in training object detection models for detecting the BEEtag markers is to consider every possible BEEtag (that has a distinct ID number) as a unique object class. Using this method, we would have a single object detector that would have as many output classes as the number of different tags that we would like to detect and identify. This would then require a dataset of BEEtag containing images labelled with their locations and IDs (which we can generate). Finally, having generated this dataset, it would be used to train the desired model to detect and identify BEEtags that appear in images. While such a method, if it worked well,

could offer a simple solution that handles both detection and identification of BEE-tags at once, it has several drawbacks non-obvious drawbacks and when attempted in practice it runs into several problems, which we describe in the rest of this section.

The first limitation of this approach is that it requires a new model to be trained every time a new set of tags is required to be detected. As explained in the previous chapters, the BEEtag specifications allows different sets of tags with different characteristics (most notably different minimum hamming distances between all pairs in the set) to be produced and used. Each of these sets contain a different group of tags, and in the general case these sets cannot be assumed to contain one another, or even share any number of elements with each other. This implies that a new detection model would need to be trained every time a larger or smaller set of tags, or a set of tags with different hamming distance requirements, are needed. This creates a need to train a new instance of the model each time a changed set of BEEtags is used, which can be a lengthy process that ties up computational resources. Attempting to address this by training a detector for a very large set of tags reveals leads to rapidly decreasing rates of object identification, which is a second limitation of this method. And even with a fairly limited number of distinct BEEtags, both detection and identification performances are severely reduced, which precludes the use of this method for distinguishing the identities of any number of tags.

4.2 Using Separate Models that are Cascaded for Detection and Identification

Given the problems of the methods we discussed in the previous section, we propose a different approach for localising and identifying BEEtags in images and videos. Our method's structure is a two-model pipeline that consist of different models to handles the localisation and identification of the marker tags. We build the structure of our pipeline in such a way that the two models that make up this pipeline can operate independently, and if needed in parallel. The models can be trained independently of each other, we can swap out the identification head independently without needing a full retrain of the detector model. The overall structure of the BeeDet-ID detection and identification pipeline is illustrated in Figure 4.1.

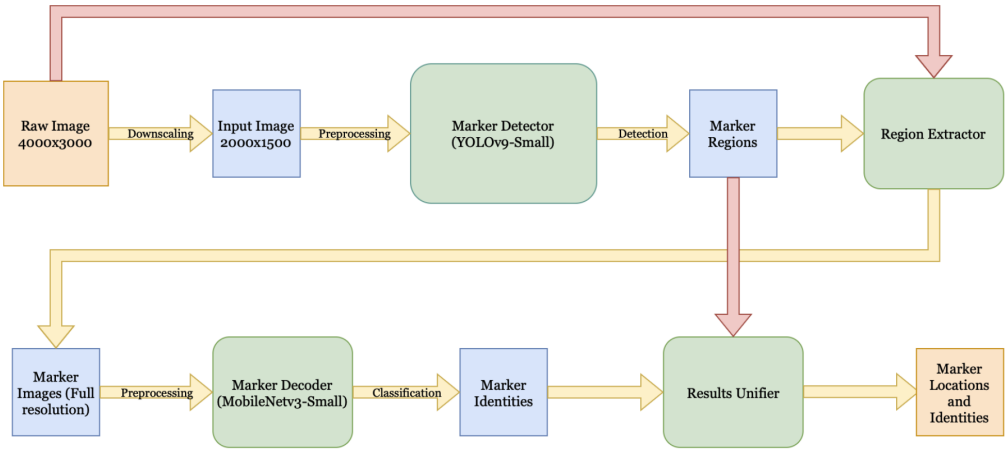


Figure 4.1: Overview of the BeeDet detection and identification pipeline

In the remainder of this chapter we describe the methods we have developed and the results we have got for the first stage of our pipeline, which is the detection/localisation of tag objects in any given images. The following sections provide detailed descriptions of our experiment setups and our results.

4.3 Data and Experiment Design

4.3.1 Test Data

To test the effectiveness of models trained on data generated by our methods, we build a simple dataset of visual fiducials to train and test object detection models on. To this end, we build a set of background and object images for training data generation, as well as human annotated test data to measure and compare the effectiveness of the resulting models.

The test images come from multiple behavioural experiments on bees marked with BEEtags [40]. The BEEtags are attached to each bee's thorax, and are 3mm×3mm in size. The images are acquired using 4000×3000 cameras through a glass panel in the experimental setup, in a room illuminated by white LED lighting. The bees are at a constant distance from the camera for the entire duration of the experiments, and on average each 3mm×3mm tag occupies a 50 pixel × 50 pixel square when parallel to the camera plane and axis-aligned. Each test image may contain a different number

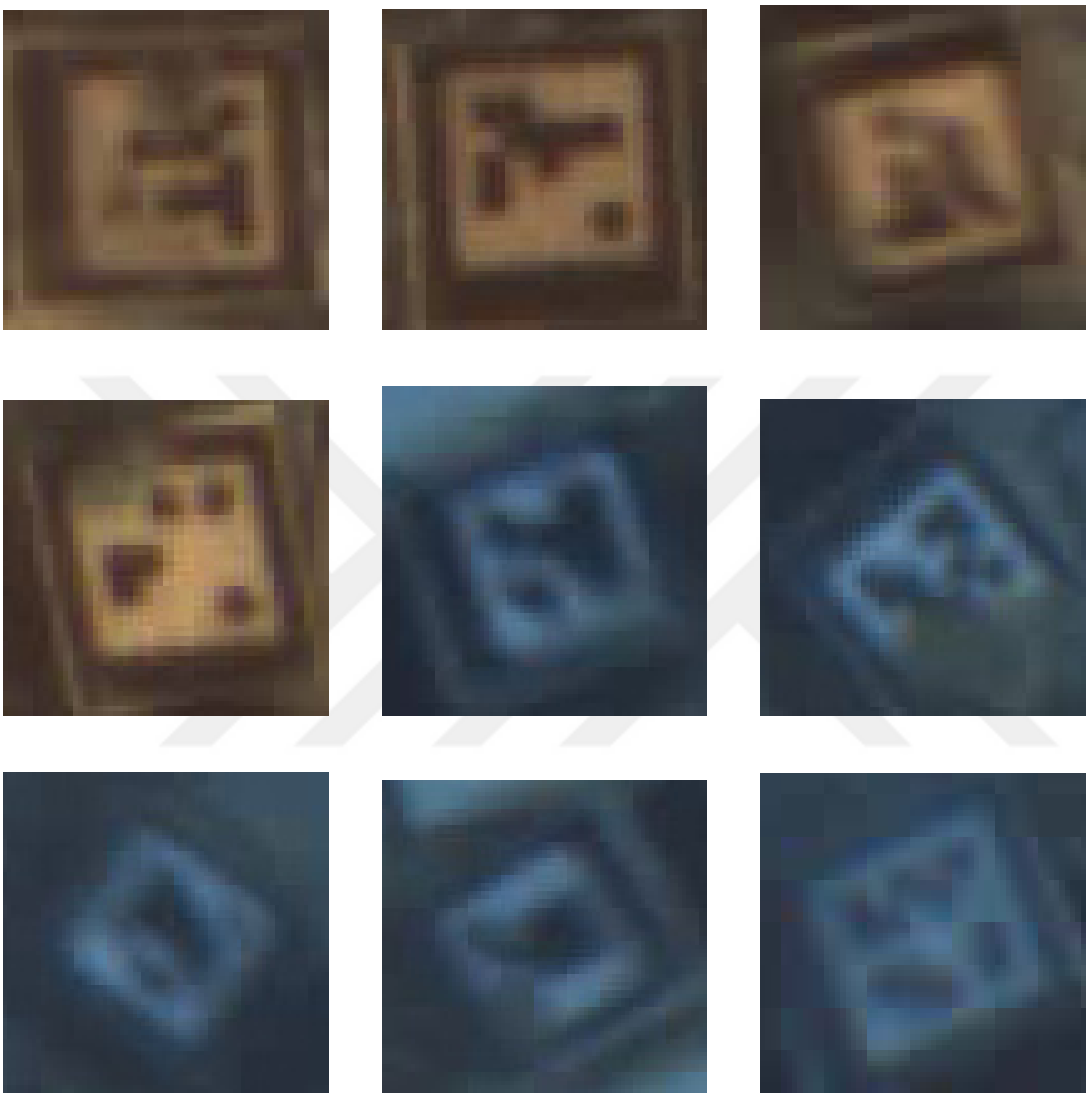


Figure 4.2: A collection of BEEtag markers taken from experiment footage. Complicating factors such as deformations, motion blur and occlusions can all bee seen

of visible markers due to the movements of the bees. The distribution of test images based on how many visible tags they contain are shown in Table 4.1.

Table 4.1: Test set statistics

Number of tags = n	3	4	5	6	7	8
Number of images with n tags	2	3	10	11	18	6

Our test dataset contains 50 such images, which together contain a total of 308 tag objects in them. Each image contains between 3 and 8 tags, with a median of 6; the most frequent number of tags in an image is 7. Example tag crops from the test set can be seen in Figure 4.2. A sample test frame is given in Figure 4.3

4.3.2 Source Data for Image Generation

We use a small number of source images to generate our training data. These source images include 20 different background images, and 19 different object images of 19 distinct tag objects, that is to say we include one image for each distinct tag. Our background images are diverse in spite of their low numbers, we include similar and different experimental setups, unrelated honeycomb images and even some images of writings. This diversity, along with our various augmentation schemes from the previous chapter, helps ensure that models trained on data generated from these source images have the best chance to be general and versatile as well as highly performant.

4.4 Experiments on Honey Bee Behavioural Experiment Data with BEETags

4.4.1 Single-Class Object Detection Experiments

Our first set of experiments aims to validate that training data generated based on such a small number of images can actually be used to effectively train an object detection model, as well as to explore the relation between the amount of images generated for training and overall model performance, and possibly to explore the maximum amount of data that can be generated from a certain quality and quantity of source



(a) Sample test frame



(b) The same frame with detection results drawn in

Figure 4.3: A sample test frame and the locations of markers within it

images that is still informative for the models being trained. We use the yolov9 [41] object detection model's "small" variety for our object detection tests. This is a recent model and the small version can achieve excellent performance with limited amounts of data, and will thus provide us with useful results regarding the effectiveness of our data for learning. Figure 4.4 shows examples of markers being detected on bees.

Given that we already know that visual markers require a minimum size in terms of pixels per edge to be identified correctly, we create and test our models on two different versions of the datasets. Our aim in having this replication to explore whether having a separate detection stage that does not require identification as a subtask can work with markers that appear smaller in images, which would also reduce the minimum resolution required of images for marker detection. In the first group we create images with the resolution 2000×1500, and scale down object images proportionally to have 25 pixel long edges on average. In the second we instead create images the are 1000×750, proportionally scale down the object images as we had done for the first group, to have around 13 pixel long edges, on average. For both of these variations we create five sets of training and validation data; each of these sets contain a different number of images, starting with 2560 and increasing all the way up to 10240 images. The exact number of samples present in each set as well as their training/validation splits are presented in Table 4.2.

Table 4.2: Dataset sizes for training and validation sets

Training	2048	3072	4096	6144	8192
Validation	512	768	1024	1536	2048

To see the suitability of our dataset for training, we use our human-annotated test set to determine how well its results translate to performance on real-world data. We start off with the results of using 1000×750 images for detection. We give Precision-Confidence, Recall-Confidence, Precision-Recall and F1 Score-Confidence curves for the results on the test set in Figure 4.5. One notable feature in these results is that for all training set sizes, the Precision-Confidence curve reaches its maximum value of 1 at or very close to 0.85 confidence. Another is that while performance is stagnant for the first four sets as dataset size increases, there is a sharp increase for the dataset

with 8192 samples, with a twice to three times increase on all metrics.

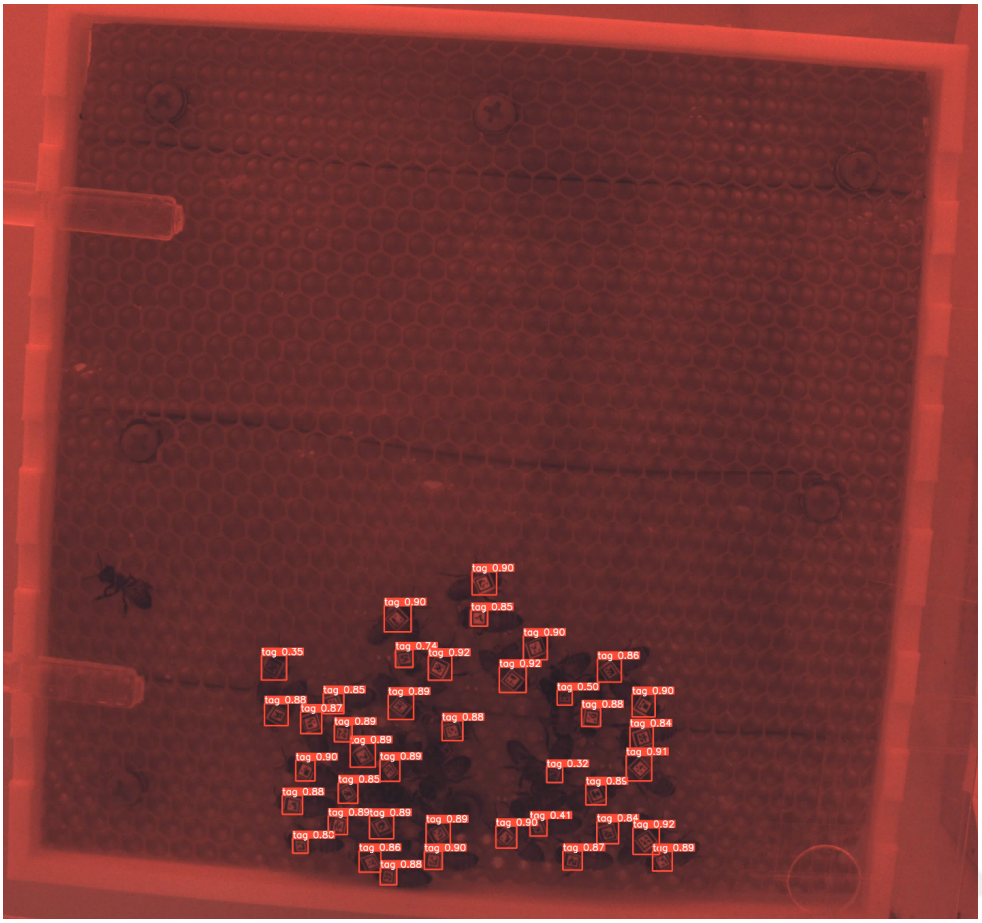
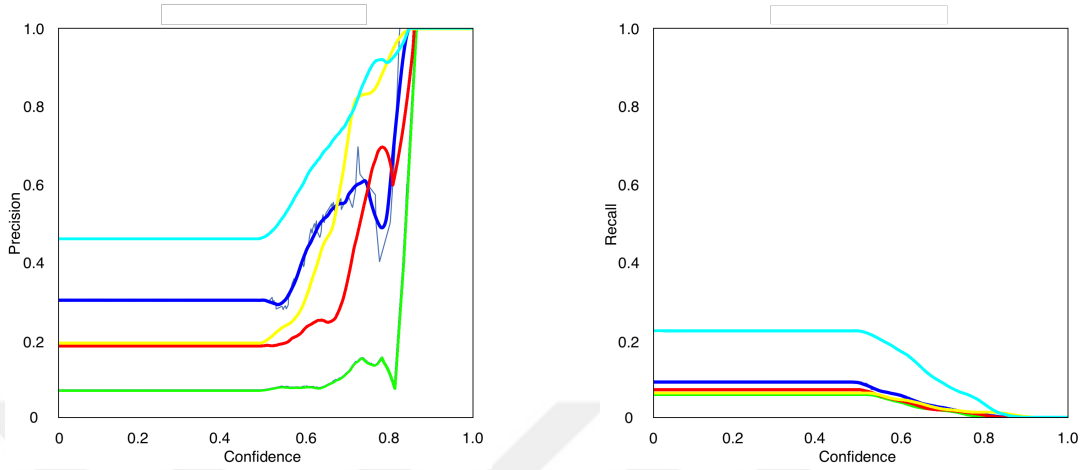


Figure 4.4: Bees in an experiment setup, with their markers detected

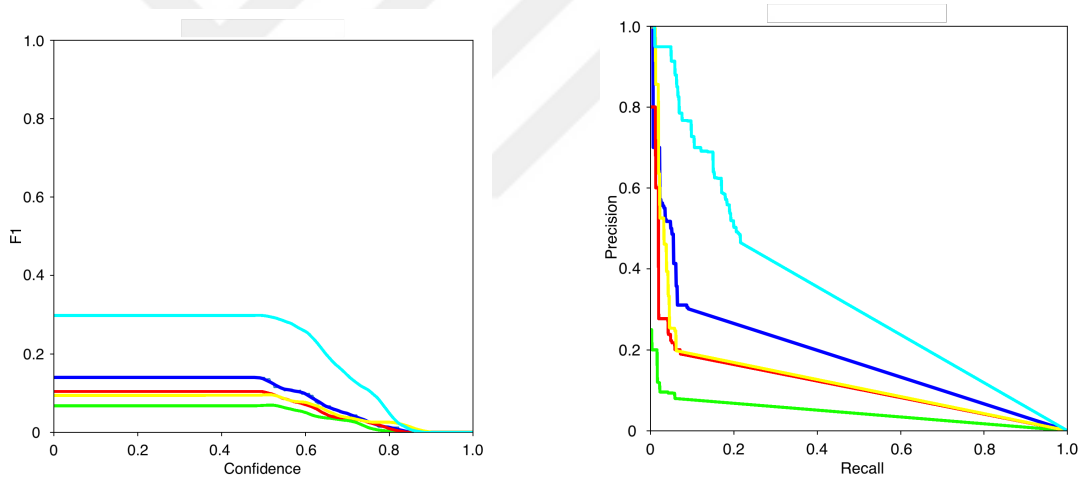
We present the same metrics for our models trained on datasets of 2000×1500 images in Figure 4.6. Performance on all metrics are higher than was for 1000×750 images, as is expected. The first feature of note is that for all models regardless of training set size, the precision value is very high for all training set sizes, being exactly at or very close to 1 for all confidence values. A second noteworthy feature is that similarly to the 1000×750 images, there is a jump (though not as large, proportionally) in recall values and F1 scores; but occurs at 6144 sample images rather than 8192 sample images this time.

Overall, the strongest conclusion we can draw from these experiment results is that processing our images with higher resolution disproportionately increases the performance of the model. We consider numerous factors that could all contribute to such a



(a) P-confidence

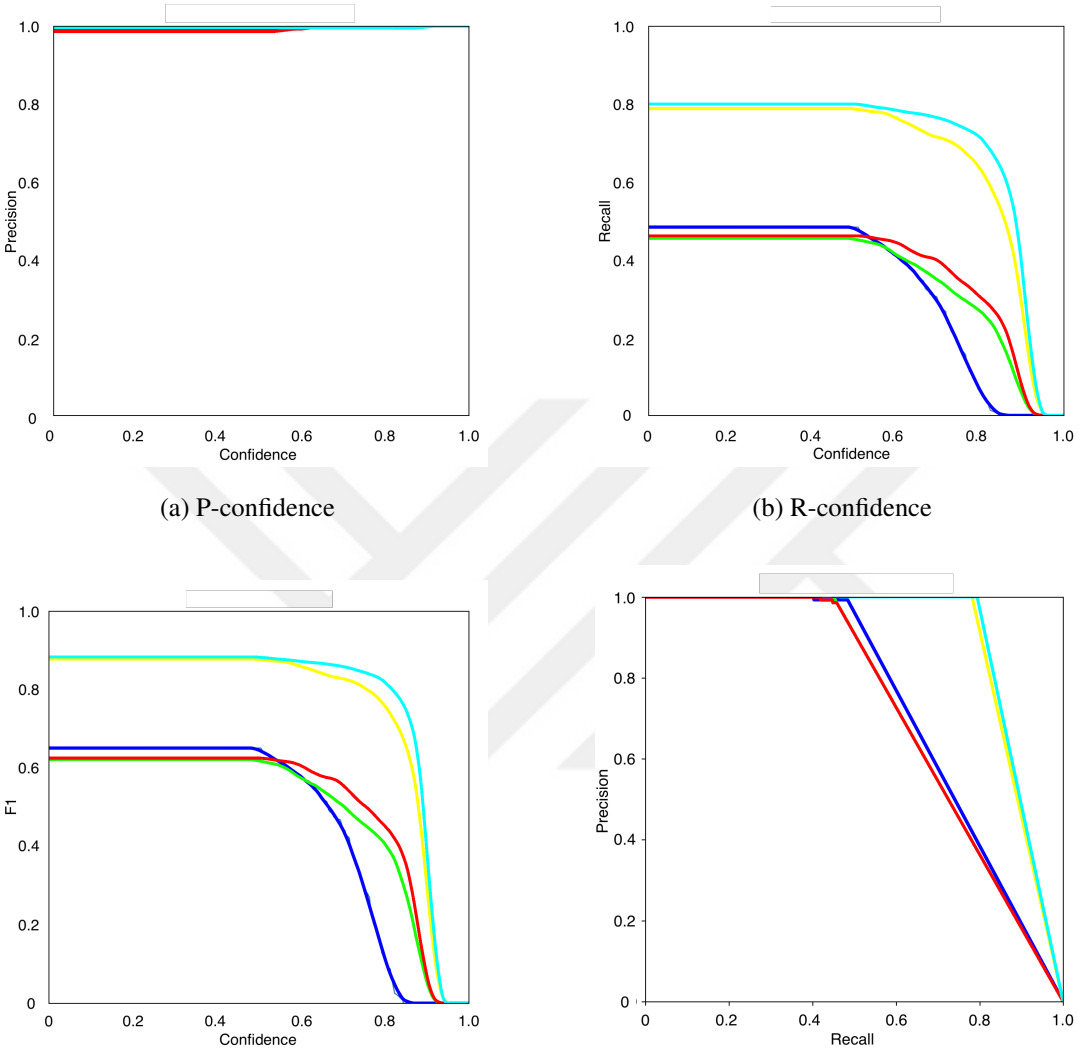
(b) R-confidence



(c) P-R

(d) F1-Confidence

Figure 4.5: Graphs of metrics for test results on 1000×750 images. The colours blue, green, red, yellow, cyan represent the results for 2048, 3072, 4096, 6144, and 8192 training samples, respectively



(a) P-confidence

(b) R-confidence

(c) P-R

(d) F1-Confidence

Figure 4.6: Graphs of metrics for test results on 2000×1500 images. The colours blue, green, red, yellow, cyan represent the results for 2048, 3072, 4096, 6144, and 8192 training samples, respectively

stark outcome. Firstly, as the images get scaled smaller and smaller, so do the objects that are contained within them. Ordinarily, this reduction would limit the maximum performance that an object detection model, or for that matter a human, may display on a given detection task. However, we also consider that at 1000×750 resolution, each tag in the image is approximately 12 pixels wide and high. Therefore we also consider the possibility that some tag objects in these test images may fall below some critical detectability threshold in size, that is, they may become too small in size to be reasonably distinguished as the object that they are at all by automated systems *or* humans. A second consideration stems from the method we use for generating the training datasets. During generation we add a single constant-variance noise, and use fixed-size constant Gaussian kernels at various steps for blurring input and intermediate images. The effect that these processes have on the objects within training images may be higher when compared to the sizes of the objects themselves, that is, the same blurring parameters that works well for a larger object within a larger image may alter a smaller image too much. This can result in difficulties in fitting a model to the data, or as we have observed, drifting the objects in training images too far from their real world counterparts, resulting in lower test performance despite having higher performance during training. We expect that both of these factors pay a role in the results that we have observed for our smaller images.

We consider these results sufficient to demonstrate that object detection models trained on data generated using our method can perform well on real-world scenarios; given that a sufficient number of samples are generated, even if only a small handful of source images are used for the generation. Given the results we have observed in our experiments so far, we will use the models trained on the larger 2000×1500 images for comparisons with later experiments.

4.4.2 Conventional Tag Detection Experiments

We compare the detection results we have obtained in Section 4.4.1 with a previous method used for detecting the tags that we have used (or indeed, any other visual fiducial marker). This requires exploration and setting of various parameters of the system manually, and that needs to be done separately for images under different

conditions (such as lighting, image clarity, background conditions), making the whole system arduous and fragile to operate. The exact details of this method can be found in Appendix **??**. The results we have obtained from this system (in the column titled IPC) for detection, is given in Table 4.3, in terms of precision, recall and F1 score. The reason that our results are given in terms of these metrics is due to the fact that these are the metrics that are applicable to the results produced by the image processing BEEtag detection method. Since this method does not output explicit confidence values, metrics that rely on a confidence value being available are not applicable to this method, and we instead present our results in terms of metrics that we can use to compare the methods under investigation. For calculating the values presented here, we have found and used a set of parameters that result in the best results over the whole test set. Note that for comparison with the object detection models, we consider any object that has been detected by the system as a valid detection, regardless of whether or not it has been assigned the correct class, since the detection systems we are currently comparing it with consider all tags to belong to a single class.

Table 4.3: Comparison of detection performances

	1000 × 750 images					2000 × 1500 images					IPC
Samples	2048	3072	4096	6144	8192	2048	3072	4096	6144	8192	-
Precision	0.30	0.09	0.19	0.20	0.45	0.99	**1.00**	0.98	**1.00**	0.99	**1.00**
Recall	0.09	0.06	0.07	0.06	0.22	0.48	0.45	0.46	0.78	**0.80**	0.17
F1 Score	0.14	0.07	0.10	0.10	0.30	0.65	0.62	0.63	0.88	**0.88**	0.29

Table 4.3 demonstrates that even with low amounts of original images and small training set sizes, we can generate datasets which can result in models that outperform traditional manually tuned models. One reason is that even with a small number of source images, we can bring about a wide variability in positions, orientations and background-relations of the objects that appear in our generated training images, which can then be picked up by any models using the training data, inducing greater generalisability. Another reason for the superior performance of our models is that by grouping the similar objects (in this case the tags) into a single class, we have separated the task of finding these objects in an image from the difficult task of discerning the subtle differences between these very similar object in order to distinguish them.

4.4.3 Single-Class Detection with Conventional Method as Classifier

In the previous experiment we have compared a single-class detection model trained on data we have generated with a conventional method of detection for the tags, based on image processing transformations. This comparison was based on how many of the tags were detected by both methods, disregarding class (indeed, the models trained in the first experiments do not output identity information regarding the tags that they detect). Now we build an end-to-end system that will both detect the locations and find the identities of tags.

To build such a system we need to combine our detection model with a classifier for our tags. We can use any method that can identify the tags for this. In this experiment we combine the detection models from 5.3.1 with a conventional system for identifying tag images. This system is very similar to the one used for comparison in 5.3.2, but with the detection parts removed (which will be replaced by the models from 5.3.1). We then use this combined system to detect and identify each tag in the same test set. The results of this experiment are summarised in

Table 4.4: Comparison of detection performances

	1000 × 750 images					2000 × 1500 images				
Samples	2048	3072	4096	6144	8192	2048	3072	4096	6144	8192
Precision	0.92	0.95	0.97	0.99	1.00	0.99	1.00	1.00	1.00	1.00
Recall	0.07	0.12	0.06	0.03	0.14	0.43	0.36	0.41	0.61	0.62
F1 Score	0.13	0.21	0.11	0.06	0.25	0.60	0.53	0.58	0.76	0.77

Comparing Table 4.4 with Table 4.3 we can see some expected changes resulting from the combination of the methods. Firstly, we see that the results from the combined system has higher precision and lower recall than using only the models from experiment 5.2.1. This is in line with our expectations, since the method we used for detection in 5.2.2 is a very high-precision model, its classifications, in addition to adding class information to our detections, also effectively act as a second filter for the detection results of the initial models. We can also see that the recall values (and consequently the F1 scores) are also much higher than when the conventional

method was applied on its own in 5.2.2. This can be attributed to the fact that working on cropped images produced based on the output of the detection models, the conventional model's input space is much more restricted than when it also had the responsibility to detect an unspecified number of tags in an image. On images that contain only one tag each, and where the size of the tags are relatively stable, covering nearly all of the image for most cases, the conventional method can have its parameters set more tightly to take advantage of the aforementioned restrictions and can thus perform better than on its own.

4.5 Detection Results Summary

This chapter has included the results of our experiments on using our data generation method to train effective object detection models. These result have demonstrated the efficacy and the efficiency of generating training data using this method, as opposed to having a large training set labelled by human annotators. More specifically, we can draw the following conclusion from the experiment results provided in this chapter:

- From Section 4.4.1, we can conclude that training data generated by the method described in Chapter 4 can be effectively use to train object detection models that can generalise well to real-world data.
- The comparisons in Section 4.4.2 demonstrated that models trained on such generated data can outperform even methods tailored specifically for detection on visual fiducial markers in images.
- The results in Section 4.4.3. demonstrate that using our method to separate the localisation from the identification of visual fiducial markers into two distinct steps can effectively improve both localisation and identification results, as compared to other methods.

CHAPTER 5

FIDUCIAL MARKER IDENTIFICATION

In this chapter we explore the task of recognising the unique identities of our visual fiducial markers, which constitutes the second part of our overall tag detection and identification pipeline. Section 5.1 describes how the gap between the two parts of the pipeline is bridged, demonstrating all the necessary pre-processing steps to prepare the input of our tag identification model. Section 5.2 outlines the method used for assigning identities to the detections from the first stage of the pipeline, assigning to each detection exactly one tag ID, which includes an extra "pseudo-ID" for possibly false positive detection results. In Section 5.3 we present the results that we have obtained from our identification model, and provide comparisons with other relevant methods on similar tasks in terms of identification performance and runtime efficiency. Finally, in Section 5.4 we present the final considerations for our overall BeeDet pipeline.

5.1 Pre-Identification Processing

Once the localisation results are acquired from the first stage of the pipeline, work on assigning the identities of the detected tags (or identifying any false positives from the detection model as such) can begin. This work starts with some pre-processing steps on the input images to acquire image data suitable for the identification task, before the detection can be passed into the identifier proper.

The first step of the preprocessing is to crop the detections using the location information provided by the first stage of the pipeline, from the *original* input images. It is important to note here that we take these detection crops from the original, unaltered



(a) Taken from resized image



(b) Taken from original image

Figure 5.1: Two images of the same tag detection, taken from images of two different resolutions.

input images as they were initially given to the pipeline, not from the version that were preprocessed before being given to the first stage of the pipeline. Figure 5.1 shows the differences between crops taken from the resized image and crops taken from the original image. This technique allows us to apply different preprocessing steps to the input data based on the model that they will be used on, as needed. The most important property that we vary between the two models in this way is the image resolution, where building the architecture of the detection model to accommodate full-resolution versions of our images starkly increase both memory and runtime requirements of the model, yet the preservation of image resolution is crucial for achieving higher identification performance.

The cropped images are then all resized to 224×224, and go through processes of histogram equalisation, greyscale conversion and normalisation before they are ready to be identified. Having gone through this processing, the images can then be given to the second stage of the pipeline, the identification model, as inputs.

5.2 The Identification Stage

The identification stage of our model is built on the basis of the family of object classification models known as MobileNet. To build the identification model, we use small variation of the MobileNet v3, as described by Howard et al. [42]. To be able to make the most of the training data and resources we have, we start our training using pretrained weights for our model. The weights that we use for that purpose have been acquire by training pretraining the model on the ImageNet-1K dataset, which contains 1281167 training images, 50000 validation images and 100000 test images. The nearly 1.45 million samples in this dataset come from 1000 different classes.

Having acquired the weights, we start fine-tuning our model from these weights. We train our models for 100 epochs, then take the one displaying the best validation accuracy as the final output of our training process. Here we note that both the training and the validation data are artificially generated, with a very small amount of human annotated data mixed in. The human-annotated data is used for test purposes almost in its entirety. Also note that during training, the data that we use for training the model is generated on-the-fly, a new group of samples are created for each batch during the training of the model.

We use this model in the overall method as the marker identifier module for our pipeline. This means that we actually express and solve the problem of decoding marker IDs, which is generally done by attempting to extract bitwise information about the code that the tag represent, as an object classification problem, which leads to a more robust decoding model due to the fact that the need to rely on exact bits of the marker is no longer necessary, the model can learn how each tag looks in a more general manner without depending on "decoding" as such. (In fact this way of learning the general look of a marker is likely closer to the way a human (given that they are familiar with the looks of the relevant tags) would recognize a given marker by relying on the larger geometric patterns occurring on it). Once the trained MobileNet v3 Small model is then integrated into the pipeline, it works on tag images detected by the first stage of the pipeline and cropped by the intermediate stage. It then produces identity predictions for the IDs of tags that it was trained for (and one additional class for non-tag objects).

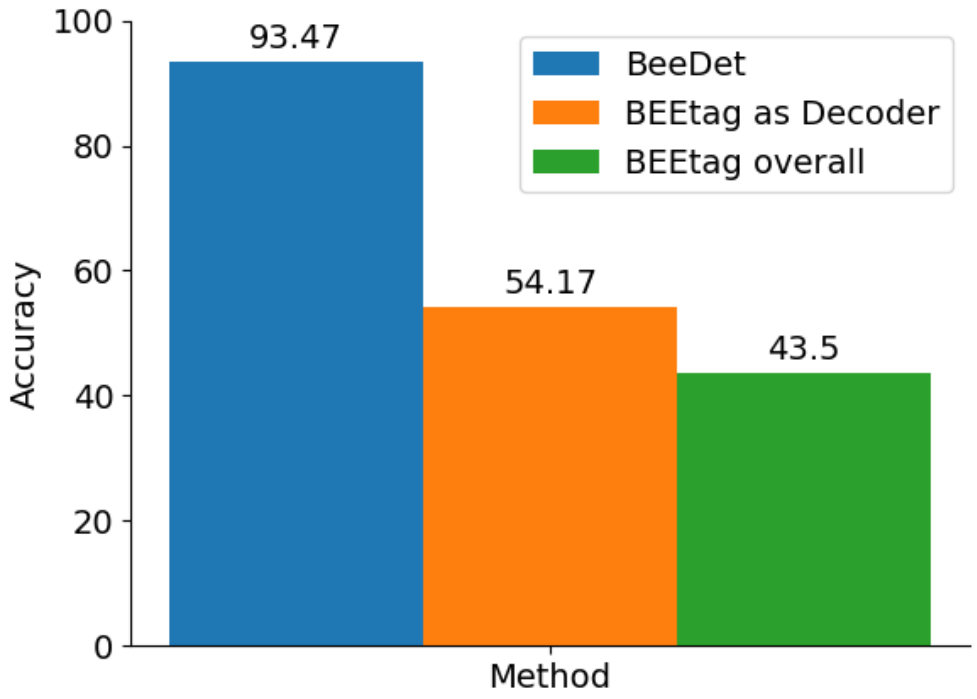


Figure 5.2: Identification accuracy as compared to the previous method

5.3 Results and Discussion

The test data we use to evaluate the identification methods are extracted from the tagged bee images in experimental setups. Before the extractions, however, we also add a number of false detections into the annotation data in places where tags are not present on the images. This addition serves to simulate false positives that may occur in the tag detection model, and to test the identification model's capabilities for correctly classifying possible non-tag crops.

Figure 5.2 gives the test accuracies as measured on human-annotated real-world data, for the identification stage on its own. The "BEEtag as Decoder" data points are acquired by feeding the BEEtag detector provided in the original BEEtag paper the same tag crops as our identification model. Since this method is originally intended to be used directly on unprocessed images, we also provide the results for applying this method on full unprocessed images as "BEEtag overall" results, both here and

throughout this chapter. (A more applicable comparison to these "BEEtag overall" results are provided in the Section 5.4.)

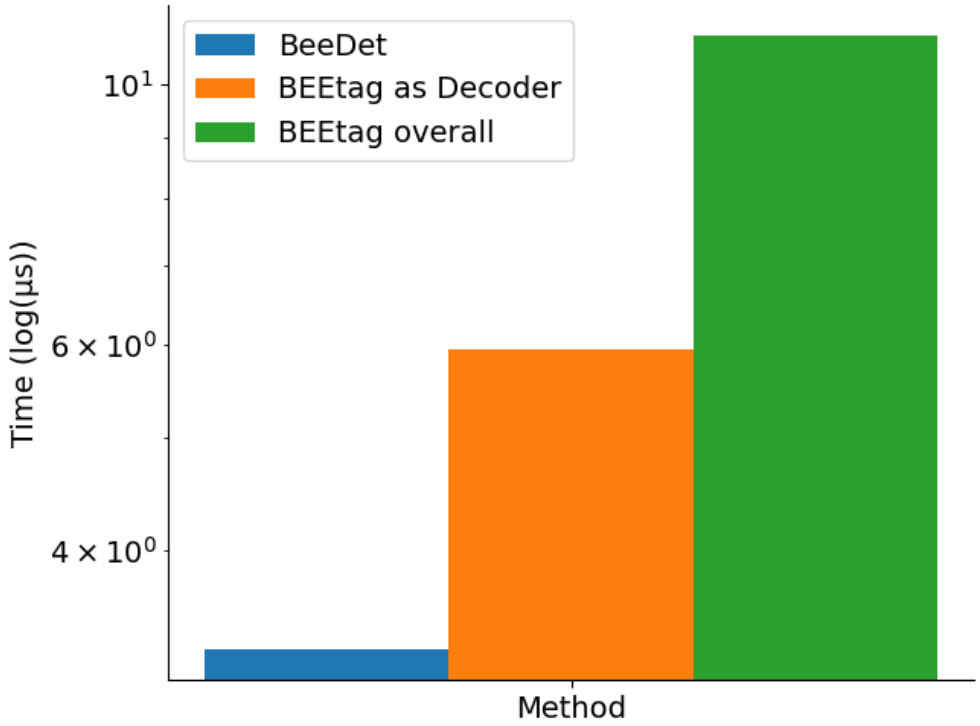
In these experiments, we consider a result correct for our identification model when it assigns the proper ID to an image of a tag, and when it assigns the "other" ID to a non-tag image. For the BEEtag method, we consider a result correct when there is only one tag detected which has the proper ID for that image in an image of a tag, and when it doesn't detect any tag at all in an image that is not of a tag.

The runtime measurements for all three methods are provided in Figure 5.3. The runtimes are provided both in terms of average runtime per tag processed, and in terms of total time elapsed during the processing of all the tags in the test set. For the "BEEtag overall" method, which is defined in the same way as it was for the accuracy results, we consider the total time elapsed during the processing of that image to be distributed equally among all tags present in that image.

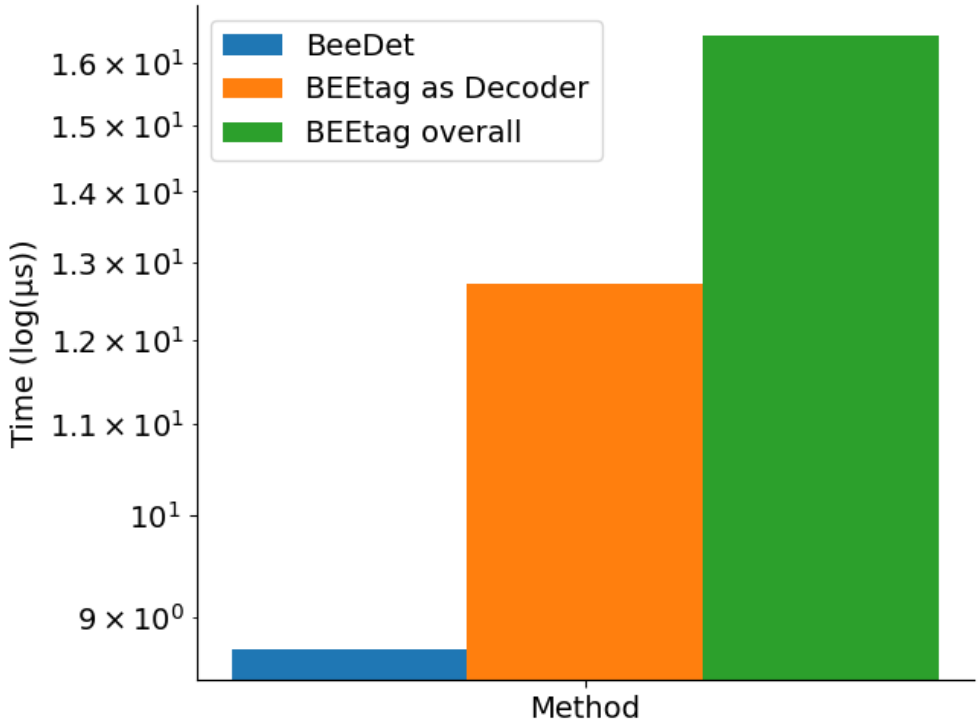
Table 5.1: Comparison of identification runtimes (ms)

	Our Method	BEEtag as Decoder	BEEtag Overall
Total	825.447	332.872	14741.0
Mean	3.383	12.214	60.163
Maximum	4.051	120.47	-
Minimum	3.315	0.627	-
Median	3.362	11.542	-
Range	0.736	119.845	-

Runtime statistics per-processed-tag, as well as runtime variability measures are presented in Table 5.1. Total identification runtime and mean identification runtime are calculated over the entire test set of BEEtags. The *Range* row denotes the difference between the slowest and the fastest processed input images. Since the overall BEEtag method does not distinguish between different tags in an image, per-tag statistics are not available for this method. Table 5.2 provides the same runtime statistics, but calculated over average runtime difference to all inputs for each input, in order to better demonstrate differences in runtime variability.



(a) Average runtime over all images



(b) Total runtime for all images

Figure 5.3: Average and total runtimes over the same test set for all three compared methods.

Table 5.2: Comparison of all-average identification runtimes differences (s)

	BeeDet-ID	BEEtag as Decoder	BEEtag Overall
Total	6.629	1152.552	14741.0
Mean	0.027	6.034	60.163
Maximum	0.685	108.888	-
Minimum	0.016	4.038	-
Median	0.017	4.800	-
Range	0.669	104.850	-

Overall, we can conclude from Figures 5.2 and 5.3 that our method achieves both significantly higher identification performance and much lower time per tag, as well as in total, than the previous alternatives. Table 5.1 quantifies these runtime differences, demonstrating that the runtimes for our proposed method are spread more tightly than the alternatives, and over a much smaller range. We can also see that the proposed method is less prone to particularly dramatic outliers than the compared methods. Table 5.2 reinforces these conclusions by demonstrating the same effects on one-to-all-comparison and group-level variability statistics for BEEtag identification.

5.4 Overall Pipeline Results

Figure 5.4 shows our final, full BeeDet-ID pipeline performance measurements, compared to the initial BEEtag detection method. Figure 5.5 shows the total runtime measurements. These charts, especially when considered with all the information that has been presented thus far in this thesis, we can clearly see the performance advancements we have achieved in our work for visual fiducial markers.

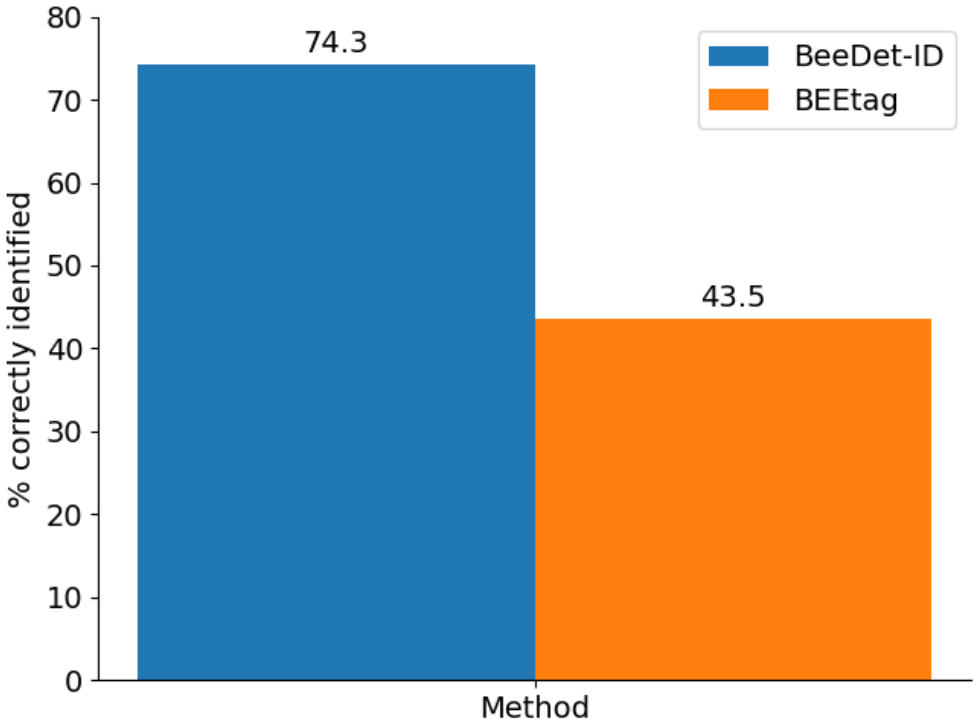


Figure 5.4: Full pipeline results

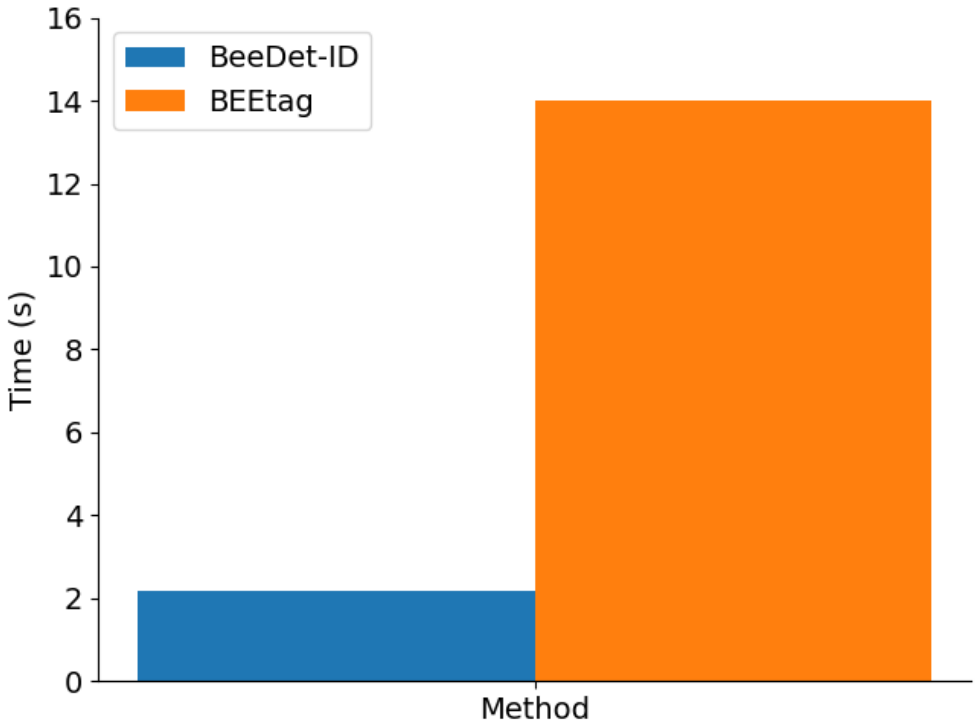


Figure 5.5: Full pipeline results

CHAPTER 6

CONCLUSION

In this thesis, we have strived to create a performant, quick and resource efficient method for detecting and identifying visual fiducial markers. To this end, we have first developed a detection and identification pipeline for visual fiducial markers. We have then created a data generation procedure that can generate training data for our visual fiducial marker detection and identification methods, using only a small amount of source data for the generation. We have demonstrated the viability of our approach testing our system on real life data of a biological experiment performed on bees that have BEEtag markers attached to their thoraxes.

With the training data that we have generated using our method, we train both the detection and the identification stages of our proposed pipeline. We have tested our pipeline trained in this way on real-world data to measure its performance. We have compared its performance to previous methods of visual fiducial detection and identification tested on the same set of real-world data.

Particularly, we can claim that as major contributions in this thesis we have:

1. defined and implemented an end-to-end pipeline for localisation and identification of visual fiducial markers
2. defined a powerful data generation method that can provide very effective data for training of object detection and classification methods
3. demonstrated the viability of our results by preparing a sample set of human annotated data, on which we have shown the viability of our methods
4. finally, we have proposed a method for marker detection that, as demonstrated

throughout this thesis can actually achieve significantly improved performances

6.1 Limitations and Future Work

Even though the the methods that we have presented in this thesis had been demonstrated to be powerful and performant, they also posses some weaknesses and limitations. Among these weaknesses, we consider the most important to be the fact that in order to change, expand or shrink (which we may want to do in order to lower classification errors) the set of tags to be detected by our pipeline, a retraining of at least the classification stage is required. In environments that require frequent changing of tag sets, it would be necessary to update the models as well. We can consider consider the fact that there is an upper limit to the variability in training data that can be created using a small set of original input images another limitation of the overall method of building a detection and identification pipeline using a small number of source images using data generation.

The following possible avenues of exploration may be considered to address the limitations of the proposed methods and improve their power and performance even further:

- Methods for increasing the generality of the detection and identification models such that no large scale retraining is needed for changes in the target set of tags.
- Expanding the methods that provide variability to our generated images in a way that increases the overall learnable information from the dataset.
- Analysing the model's interactions with generated data during training time to further modify our models fro better learning from artifically generated data.

REFERENCES

[1] R. C. Verma, M. A. Waseem, N. Sharma, K. Bharathi, S. Singh, A. R. A., S. K. Pandey, and B. V. Singh, "The role of insects in ecosystems, an in-depth review of entomological research," *International Journal of Environment and Climate Change*, vol. 13, p. 4340–4348, september 2023.

[2] K. M. Scaplen, N. J. Mei, H. A. Bounds, S. L. Song, R. Azanchi, and K. R. Kaun, "Automated real-time quantification of group locomotor activity in drosophila melanogaster," *Scientific Reports*, vol. 9, march 2019.

[3] V. Patel, N. Pauli, E. Biggs, L. Barbour, and B. Boruff, "Why bees are critical for achieving sustainable development," *Ambio*, vol. 50, p. 49–59, april 2020.

[4] K. Bozek, L. Hebert, Y. Portugal, A. S. Mikheyev, and G. J. Stephens, "Markerless tracking of an entire honey bee colony," *Nature Communications*, vol. 12, march 2021.

[5] J. D. Crall, N. Gravish, A. M. Mountcastle, and S. A. Combes, "Beetag: A low-cost, image-based tracking system for the study of animal behavior and locomotion," *PLOS ONE*, vol. 10, p. e0136487, september 2015.

[6] F. Wario, B. Wild, M. J. Couvillon, R. Rojas, and T. Landgraf, "Automatic methods for long-term tracking and the detection and decoding of communication dances in honeybees," *Frontiers in Ecology and Evolution*, vol. 3, september 2015.

[7] A. Gallala, B. Hichri, and P. Plapper, "Human-robot interaction using mixed reality," *2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, p. 1–6, december 2021.

[8] A. Mutka, D. Miklic, I. Draganjac, and S. Bogdan, "A low cost vision based localization system using fiducial markers," *IFAC Proceedings Volumes*, vol. 41, no. 2, p. 9528–9533, 2008.

[9] A. Causo, Z.-H. Chong, R. Luxman, and I.-M. Chen, "Visual marker-guided mobile robot solution for automated item picking in a warehouse," *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, p. 201–206, july 2017.

[10] E. Olson, "Apriltag: A robust and flexible visual fiducial system," *2011 IEEE International Conference on Robotics and Automation*, p. 3400–3407, May 2011.

[11] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 4193–4198, october 2016.

[12] B. Wang, "Lftag: A scalable visual fiducial system with low spatial frequency," 2020.

[13] J. C. P. Cheng, K. Chen, and W. Chen, "Comparison of marker-based and markerless ar: A case study of an indoor decoration system," *Lean and Computing in Construction Congress - Volume 1: Proceedings of the Joint Conference on Computing in Construction*, p. 483–490, july 2017.

[14] M. Kaltenbrunner and R. Bencina, "reactivision," *Proceedings of the 1st international conference on Tangible and embedded interaction*, february 2007.

[15] J. Mooser, S. You, and U. Neumann, "Tricodes: A barcode-like fiducial design for augmented reality media," *2006 IEEE International Conference on Multimedia and Expo*, p. 1301–1304, july 2006.

[16] B. Schoun, H. Oagaz, and M.-H. Choi, "Corner-based square fiducial marker detection for hand-manipulated ar objects," *2021 IEEE International Conference on Intelligent Reality (ICIR)*, p. 31–38, May 2021.

[17] D. P. Mersch, A. Crespi, and L. Keller, "Tracking individuals shows spatial fidelity is a key regulator of ant social organization," *Science*, vol. 340, p. 1090–1093, May 2013.

[18] D. Penaloza-Aponte, S. Brandt, E. Dent, R. M. Underwood, B. DeMoras, S. Bruckner, M. M. López-Uribe, and J. V. Urbina, "Automated entrance monitoring to investigate honey bee foraging trips using open-source wireless platform and fiducial tags," *HardwareX*, vol. 20, p. e00609, december 2024.

[19] T. Gernat, V. D. Rao, M. Middendorf, H. Dankowicz, N. Goldenfeld, and G. E. Robinson, "Automated monitoring of behavior reveals bursty interaction patterns and rapid spreading dynamics in honeybee social networks," *Proceedings of the National Academy of Sciences*, vol. 115, p. 1433–1438, january 2018.

[20] B. Wild, L. Sixt, and T. Landgraf, "Automatic localization and decoding of honeybee markers using deep convolutional neural networks," 2018.

[21] S. W. Wolf, D. M. Ruttenberg, D. Y. Knapp, A. E. Webb, I. M. Traniello, G. C. McKenzie-Smith, S. A. Leheny, J. W. Shaevitz, and S. D. Kocher, "<scp>naps</scp>: Integrating pose estimation and tag-based tracking," *Methods in Ecology and Evolution*, vol. 14, p. 2541–2548, august 2023.

[22] T. D. Pereira, N. Tabris, A. Matsliah, D. M. Turner, J. Li, S. Ravindranath, E. S. Papadoyannis, E. Normand, D. S. Deutsch, Z. Y. Wang, G. C. McKenzie-Smith, C. C. Mitelut, M. D. Castro, J. D'Uva, M. Kislin, D. H. Sanes, S. D. Kocher, S. S.-H. Wang, A. L. Falkner, J. W. Shaevitz, and M. Murthy, "Sleap: A deep learning system for multi-animal pose tracking," *Nature Methods*, vol. 19, p. 486–495, april 2022.

[23] J. Crall, N. Gravish, A. M. Mountcastle, and S. A. Combes, "Beetag: a low-cost, image-based tracking system for the study of animal behavior and locomotion," june 2015.

[24] J. Chai, H. Zeng, A. Li, and E. W. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, p. 100134, 2021.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[26] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast segment anything," 2023.

[27] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," 2015.

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[32] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, 1999.

[33] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.

[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[36] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[37] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2020.

[38] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[39] Z. Ren, S. Lensgraf, and A. Q. Li, “Improving the perception of visual fiducial markers in the field using adaptive active exposure control,” 2024.

[40] J. D. Crall, N. Gravish, A. M. Mountcastle, and S. A. Combes, “Beetag: A low-cost, image-based tracking system for the study of animal behavior and locomotion,” *PLOS ONE*, vol. 10, pp. 1–13, 09 2015.

[41] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, “Yolov9: Learning what you want to learn using programmable gradient information,” 2024.

[42] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, “Searching for mobilenetv3,” 2019.

[43] S. Suzuki and K. be, “Topological structural analysis of digitized binary images by border following,” *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

APPENDICES

A The Conventional Tag Detection Method

Throughout this thesis we refer to the *conventional tag detection method* both as a point of reference and as a source of baseline results with which to make comparison. We describe the working of this method here.

1. Convert the input image to greyscale, if given in colour
2. Apply Gaussian blur with a 5×5 kernel. This suppresses spurious contours in the image, and is vital for both result quality *and* for the runtime performance of the program.
3. Create a binary image from the greyscale one by applying adaptive thresholding. We calculate the threshold values for the neighbourhoods as the mean of the values in the neighbourhood, minus a constant c that is set manually.
4. Find the contours in the binary image. We use the algorithm from Suzuki et al.[43] to find the contours of the image.
5. Fit approximate polygons on the contours of the image. We use the Ramer–Douglas–Peucker algorithm for this.
6. Among all the polygons acquired, discard all that are not quadrangles.
7. Calculate the area of all quadrangles, then discard those that are not between minimum and maximum area values. These two limits need to be set manually.
8. Fit axis-aligned bounding rectangles over every quadrangle, then calculate the absolute difference $|b - a|$ between the lengths of the sides of the rectangles. Discard any quadrangle that have a larger difference between its edges than a predetermined maximum difference value. This maximum difference value needs to be set manually.

9. Crop all the remaining rectangles from the greyscale image.

10. Decode the cropped images into tag identities:

    (a) Resize the cropped images into perfect squares. This may require deforming as well as rotating and scaling the image.

    (b) Create binary images from the cropped resize tag images by thresholding. This step need not be identical to the thresholding in step 3, neither in terms of its thresholding strategy nor the parameters of the strategy.

    (c) Separate the resized image into a 7×7 grid of cells. Note that the BEEtag has a 5×5 payload area, and the total size of the tags with their prescribed borders is a 7×7 grid.

    (d) Decode each cell of the 5×5 grid in the cropped image into a single bit. Since the image in the grid is already thresholded, it is sufficient to take a count of black and white pixels inside the grid cell and decide the outcome by simple majority.

    (e) Return the resulting 25-bit number as the result of the decoding.

11. Compare the decoded image against the list of known tag identities by comparing the bits in the corresponding places. This comparison is done for each of the four possible $\frac{\pi}{2}$. This gives us the Hamming distance between the current tag and all other valid tags, in all rotations.

12. Take the closest tag among the list of valid tags. If the distance to that tag is greater than the minimum distance requirement, which must be set manually, then the tag is considered detected, with the identity of the closest tag in the list of tags. Otherwise, the detection is discarded.