SEMI-AUTOMATED IMAGE LABELING FRAMEWORK FOR HONEYBEE (APIS MELLIFERA) DETECTION AND TRACKING IN AUTOMATED BEE HIVE APPLICATIONS

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

BY

ALPER EMRE HAS

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

JANUARY 2023

Approval of the thesis:

SEMI-AUTOMATED IMAGE LABELING FRAMEWORK FOR HONEYBEE (APIS MELLIFERA) DETECTION AND TRACKING IN AUTOMATED BEE HIVE APPLICATIONS

submitted by **ALPER EMRE HAS** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

ABSTRACT

SEMI-AUTOMATED IMAGE LABELING FRAMEWORK FOR HONEYBEE (APIS MELLIFERA) DETECTION AND TRACKING IN AUTOMATED BEE HIVE APPLICATIONS

Has, Alper Emre
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Hande Alemdar

January 2023, 77 pages

Beekeeping is a valuable field for both scientific research and agriculture. While beekeepers have traditionally relied on close observation, often with the help of vets, this method can provide limited empirical results. The high cost of monitoring tools presents a challenge in the beekeeping industry. To address this challenge, we implement an animal detecting and tracking algorithm as part of a framework that utilizes low-cost cameras and low-power computational equipment. This leads to the development of a semi-automated Image Labeling Software (ILS) that reduces the workload of labeling for similar frameworks. Our ILS was used in the Beehive 4.0 project, which uses an artificial intelligence-based approach for observing honeybees (Apis mellifera) in a controlled environment.

Using a low-resolution camera, we simultaneously record the test group and control group. We trained a standard YOLOv3 on our dataset and use this for object detection. To increase accuracy and reduce false positive detections, we also identified the contours of bee clusters. We then used k-means clustering to eliminate redundant detections. This study demonstrates that object detection and tracking on low-cost camera-recorded videos can be improved by using the proposed model in animal keeping. This will enhance labeling quality while reducing the labeling burden in reporting setups when the number of animals is known.

Keywords: semi-automated image labeling systems, honeybee, object detection, object tracking, YOLOv3

ÖZ

OTOMATİK ARI KOVANI UYGULAMALARINDA BAL ARISI (APİS MELLİFERA) TESPİTİ VE TAKİBİ İÇİN YARI OTOMATİK GÖRÜNTÜ ETİKETLEME SİSTEMİ

Has, Alper Emre
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Hande Alemdar

Ocak 2023 , 77 sayfa

Arıcılık hem bilimsel araştırmalar hem de tarım için değerlidir. Arıcı tarafından arı kovanını veterinerler yardımıyla yakından gözlemlemek için yaygın olarak kullanılan bir yöntem olmasına rağmen, sonuçlar ölçülebilir değildir. İzleme araçlarının yüksek maliyetleri, arıcılık endüstrisi için bir dezavantajdır. Bu çalışmada, düşük maliyetli kameralardan, düşük güç gerektiren hesaplama ekipmanından oluşan bir sistemi desteklemek ve son olarak benzer amaçlar için etiketleme iş yükünü azaltan yarı otomatik bir ILS elde etmek için bir hayvan algılama ve izleme algoritması uygulanmaktadır. ILS, Kovan 4.0: Bal Arısı (Apis mellifera) İzleminde Yapay Zeka Temelli Nesnel Yaklaşımları projesinde kullanılmıştır.

Düşük çözünürlüklü bir kamera ile hem test grubu hem de kontrol grubu aynı anda kayıt edilmiştir. Bu çalışmada YOLOv3, çalışmamızın veri seti üzerinde eğitilmiş ve nesne tespiti için kullanılmıştır. Buna paralel olarak, yanlış pozitif algılamayı azaltmak ve doğru nesneye doğru yolu atama hassasiyetini artırmak için arı kümelerinin konturu belirlenmiştir. Bundan sonra, kullanılmış olan aynı nesneye işaret eden algılamayı ortadan kaldırmak için k-means kümeleme algoritması uygulanmıştır. Bu çalışma, düşük maliyetli kamera kaydı videolarında nesne algılama ve takibinin, toplam hayvan sayısı bilindiği durumlarda raporlama kurulumları için etiketleme yükünü azaltırken etiketleme kalitesini artırmak için hayvan bakımında kullanılabilecek bir etiketleme uygulamasını geliştirmiştir.

Anahtar Kelimeler: yarı otonom görüntü etiketleme sistemleri, bal arısı,nesne tespiti, nesne takibi, YOLOv3

To my lovely wife and son.

ACKNOWLEDGMENTS

I would like to thank my MSc. thesis supervisor Assoc. Prof. Dr. Hande Alemdar for her guidance and support. My mentor and supervisor Assoc. Prof. Dr. Hande Alemdar enlightened me with academic knowledge and gave me valuable advice and motivation whenever I needed it the most. Also, I would like to thank all project members for their great determination and efforts during the project development especially my computer vision partner R.A. Oğuz Gödelek for his support.

This study is dedicated to my institution mentors for their constant guidance and to my lovely family whose full support helped me so I have completed this dissertation. My lovely wife and son supported me all the time and be there for me whenever I need them.

This study is developed under the project Bee Hive 4.0:Artificial Intelligence Based Approach to Honeybee (Apis mellifera) Observation and partially funded by The Scientific and Technological Research Council of Turkey under grant number TUBITAK 122E014.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

HCI	Human-Computer Interaction,
2D	2 Dimensional,
3D	3 Dimensional,
CNN	Convolutional Neural Network,
VGG	Visual Geometry Group,
ResNet	Residual Neural Network,
ILS	Image Labeling Systems,
FP	False Positive,
FN	False Negative,
TP	True Positive,
TN	True Negative,
NMS	Non-Maximum Suppression,
BB	Bounding Boxes,
IoU	Intersection Over Union,
MOT	Multiple Object Tracking,
MOTA	Multiple Object Tracking Accuracy,
MOTP	Multiple Object Tracking Precision,
MPT	Multiple Pedestrian Tracking,
TBD	Tracking-by-Detection,
IDSW	ID Switches
TO	Tracked Object Count
RO	Recovered Object Count
GT	Ground Truth Object Count

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

The decline in honeybee populations is a critical issue that poses a significant threat to ecosystems. To address this challenge and the lack of automation in biological and agricultural studies, our study aims to develop a framework for an Image Labeling System (ILS). The ILS will integrate object detection and tracking algorithms to enhance bee detection accuracy and real-time object tracking in controlled environments. Through this approach, our proposed system can provide an automated solution to observe honeybee hives and contribute to maintaining honeybee populations. By developing a reliable and efficient system to monitor honeybee health, our proposed framework can assist beekeepers and researchers in preserving honeybee populations.

Our study focuses on animals kept in controlled environments, such as honeybees, farm animals, and poultry, which are in a stationary environment and have a known population. This requires a different approach to solve, with less computational and economic cost.

We aim to minimize the workload of the labeling process while ensuring high labeling accuracy in our ILS framework. The ILS will be tested on laboratory and field trials to validate its practical applications and demonstrate its cost-effectiveness. This will reduce the financial burden of scientific studies and make it accessible to commercial beekeepers for automated monitoring of their colonies.

It is estimated that up to 75% of biomass and species diversity could be lost globally, leading to a 68% decrease in populations of various species of mammals, birds, reptiles, fish, and amphibians between 1970 and 2016 [13, 14]. This 6th mass extinction also referred to as rapid ecosystem collapse, is driven by factors such as climate change, habitat fragmentation, pollution, and harmful chemicals [15]. Honeybees, crucial for pollination, are responsible for 90% of commercial agriculture pollination and are facing massive declines due to colony collapse disorder and environmental factors [16].

The Bee Hive 4.0 project will provide solutions for more comprehensive investigation and quantitative data collection in studies of honeybee behavior and the effects of pesticides and environmental contaminants on honeybee colonies. The cost-effective systems developed in this project will benefit scientific studies and be accessible to commercial beekeepers.

1.2 Proposed Methods and Models

This study presents a framework for an Image Labeling System that leverages object detection and tracking capabilities to improve the precision of honeybee detection for real-time object tracking in a controlled environment. To achieve this, we design and implement the YOLOv3 algorithm as a detector to identify honeybees in a controlled environment where the number of objects is known. The output of the detector is then fed into a tracker that uses the Kalman Filter [17] and the Hungarian Algorithm [18] to track the movement of the bees. The tracker operates on the principle of "tracking by detection" and is enhanced by a reassigning strategy to minimize ID switches and error deviations.

To aid in the labeling process, we create an Image Labeling System tool that is powered by the tracker algorithm. The tool operates by examining the location information of the tracked honeybees and their assigned IDs to perform the labeling process. The tool uses the tracking result from the tracker as the foundation for the annotation file, with only modifications and potential errors presented to annotators. With the help of the developed ILS, annotators can easily rectify errors and import new labels, reducing the time needed for annotating images while ensuring high-quality labels.

1.3 Contributions and Novelties

Our thesis presents the following key contributions:

- A framework for honeybee (Apis melifera) detection that incorporates k-means clustering.
- Implementation of multiple object tracking algorithms using the "tracking by detection" approach for small objects such as honeybees.
- Development of a semi-automated Human-Computer Interaction (HCI) labeling software to reduce the workload of the labeling process.

1.4 The Outline of the Thesis

In Chapter 2, we provide an overview of recent advances in object detection and tracking literature, comparing state-of-the-art detectors for the honeybee dataset. We also examine related works in the field of Image Labeling Systems (ILS) and their capabilities.

Chapter 3 describes our method, which includes the use of the YOLO object detector and tracker. We explore the impact of annotation errors on object detection and propose a solution to correct one type of error using our semi-automated video honeybee ILS.

Chapter 4 details the experiments we conducted, including the experimental setup and procedures, and evaluates the results.

Finally, in Chapter 5, we conclude the study and suggest areas for future work to improve the system.

CHAPTER 2

BACKGROUND AND RELATED WORK

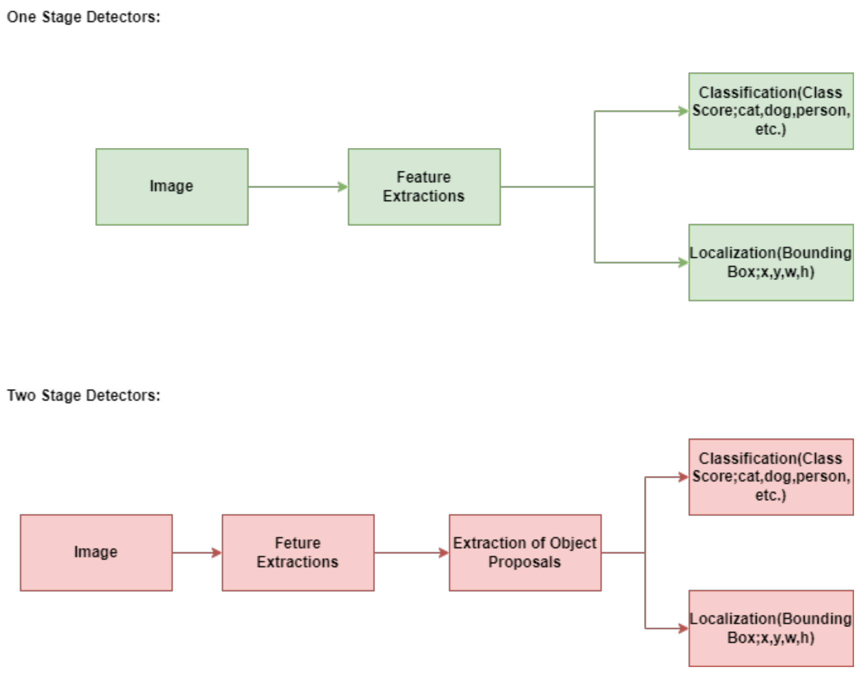
In this chapter, we will introduce the initial steps required to create the dataset, implement object detection and tracking techniques, and provide the necessary background information. We will also examine previous research on ILS.

2.1 Introduction

When discussing image classification, the question of how to accurately represent real-life object detection using an artificial vision system arises. The primary focus is on finding an effective method. However, detection alone is not sufficient to solve the problem, as it also requires the additional step of "classification" which has to be done regardless of position, pose, scale, color loss, conformation, illumination, and many other variables. To address this, visual object detection techniques have been developed over the years, trying to learn real-world objects' internal representations. In the past, these representations were produced by hand-crafted feature extractors that were intended to provide output to be fed to a trainable classifier [19].

With the advancement of deep learning methods, feature extraction has also improved. The developed methods focus on hierarchical representations, which have multiple trainable stages that are stacked on top of each other and extract features at each level. Convolutional networks are particularly competitive in this task, especially when compared to their predecessor gradient-based supervised learning methods. As the convolutional network depth increases, with an increased number of convolutional layers, the use of small convolutional filters in whole layers with lower complexity, and with further improvements such as the ResNet [20], this area of research has become even more important.

When considering the way of detection conducted by the CNN-based method, we could approach the object classification and bounding box-regression process in two categories: One-Stage Detectors and Two-Stage Detectors. The representation of the process both for one and two-stage detectors is shown in Figure 2.1.



One Stage Detectors:
Image
Feature Extractions
Classification(Class Score;cat,dog,person, etc.)
Localization(Bounding Box;x,y,w,h)
Two Stage Detectors:
Image
Feture Extractions
Extraction of Object Proposals
Classification(Class Score;cat,dog,person, etc.)
Localization(Bounding Box;x,y,w,h)

Figure 2.1: The feature extractions for one and two-stage detectors

Two–stage detectors divide the task into two successive phases. The first phase includes proposing possible detection to the second phase in which the second phase tries to recognize the proposals. These two steps are called the body and head respectively. The design of the head is generally heavy since it targets to provide the best accuracy [21].

One-stage detector feature extraction initially creates a grid for the subject image. Object detector extract features for classification and bounding box configuration. After feature extraction is performed post-processing is conducted. These processes are repeated until no proposals are left and the detection is completed.

2.2 Setup and Preliminary Works

2.2.1 Arena Setup

The Beehive 4.0 Project has planned to be carried out in the natural habitat of the honeybees, but preliminary monitoring studies have to be carried out in a controlled environment. In this phase, the close monitoring of the individual bees and their social activities will be trackable. Determination of the activity is essential to understand the farming activity and therefore the honey collection capacity. The decrease in the activity of field bees means a decrease in honey yield, as expected. Besides the activity, the determination of the behaviors is also very important. Because this behavior of the honeybees, defined as eusocial creatures, is strongly related to the colony's existence. These behaviors are defined as the interactions between honeybees with the touching of the antennae, the transfer of nutrients by the proboscis (trophallaxis), and the clustering behavior.

The setup has been prepared as seen in Figure 2.2, the distance between the two glass surfaces is 7 mm to prevent the bees from overlapping and the dimensions of the area where the bees can roam are confined as 200 x 200 mm.

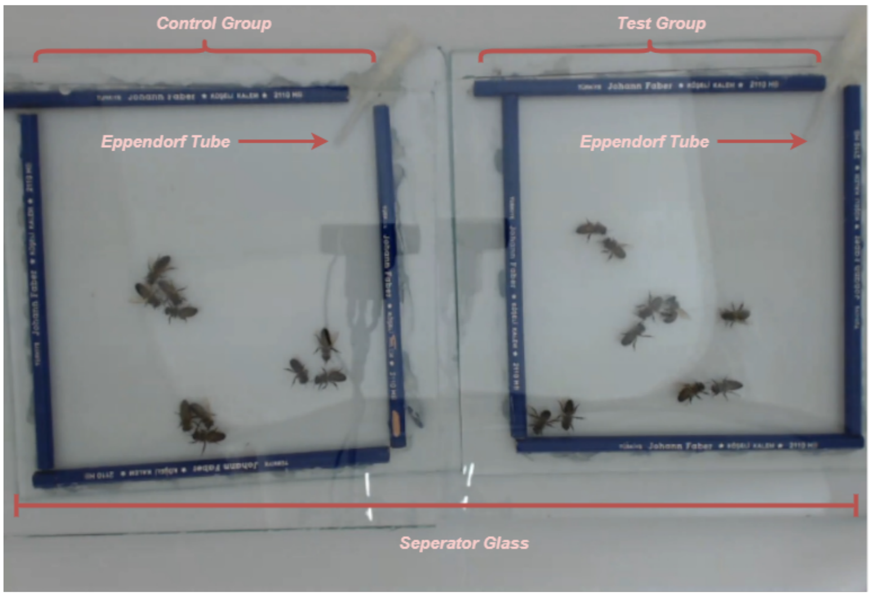


Figure 2.2: The Arena Setup of the honeybee monitoring

The Arena Setup is separated into two groups on the left side the control group takes place, and on the right side, the test group exists. Both setup group has a feeder that is made from an Eppendorf tube with a cut end to feed the bees with sucrose solution impregnated with cotton.

The Arena Setup was utilized to record honeybee behaviors using a web camera during all separate tests. The Logitech C922 Pro HD Stream webcam was employed in the experiments. This webcam has the capability to record at 1080p/30 fps to 720p/60 fps with a 3-pixel resolution through a 78-degree diagonally field of view (dFoV) glass lens. The webcam was placed in a central position within the arena setup, approximately 50 cm above the ground, and oriented at a vertical angle. During the recording process, the configuration was set to 1080p/30 fps. Additionally, regular LED lighting commonly used in daily settings was employed in the arena setup.

2.2.2 Database Generation

We labeled a total of 40 frames, each containing 20 honeybees, to train our YOLOv3 object detector. These labeled frames only include the positions of the honeybees. Then, we ran the algorithm and obtained the tracking results for 10,000 frames. By utilizing the developed ILS, we fed the tracking results, which were the output of our algorithm, and recorded the errors of the algorithm. As a result, we generated 10,000 correctly labeled and tracked frames as the ground truth to test for future improvements. Upon completion of the project, we plan to publish this dataset publicly.

2.3 Background

Data sets that are used in object detection and tracking problems (including image processing solutions) are not very informative without any pre-processing steps. These steps may include a variety of methods and techniques like; reducing computation complexity by converting to gray-scaled pictures, standardizing the width and height of pictures to feed machine learning algorithms (especially CNN algorithms), data augmentation by feeding existing data sets with a different form of the same objects regarding shape, size color, and rotations to give more hint the algorithm and many mores removing background, reducing noises, brighten or darken the images, etc. All of these techniques have been used to have discriminative and computationally more efficient images from raw images. These pre-processed images are capable of feeding machine learning algorithms with more information kept with raw images in the data set. After a series of steps, this information floats to the surface more accurately called feature extraction. Feature extraction pipelines could have fewer or more methods for data augmentation.

In the rest of this section, proposal generation and feature extraction approaches will be discussed. For proposal generation, it is common to use more conventional methods while for feature extraction approaches learning-based methods are more akin.

2.3.1 Window Scoring Object Proposal Generation

The windows scoring methods focus on proposing numbers of candidate windows referring to their objectness scores generated by the methods themselves. They first generate a sample set of candidates regarding the saliency map after this they measure the objectness scores by combining multi-scale saliency, color contrast, and super-pixels [22]. Super-pixels also provide another algorithm when proposing an object by generating super-pixels called Selective Search. Selective Search uses diverse strategies to overcome the issues with images that may have regions from objects with only color, texture, or enclosed parts [23].

2.3.2 Learning-Based Object Proposal Generation

Object proposal generation, as a pre-processing technique, has been widely used in current object detection pipelines to guide the search of objects and avoid exhaustive sliding window searches across images.

The learning-based object proposal is a pre-processing technique that is used in object detection pipelines to overcome the drawback of the sliding window search for images which brings a heavy workload. In this subsection, the changing algorithms with the advanced methods for the convolutional layers for the image will be covered. Additionally, ResNet and YOLO algorithms will be introduced.

2.3.2.1 2D CNN Architectures

Convolutional Neural Networks (CNN) are not very new technology, although they are relatively popular lately. The first publication about recognizing handwritten digits in images by CNN named was LeNet [24]. The CNN applications dated back to the 2000s but the popularity of these algorithms increased especially in 2010 thanks to GPU usage capabilities.

In addition to the GPU capabilities, the characteristic of the CNNs played a big role in the improvement of the popularity of CNN. One of the threats of CNN is having considerably low operations to complete a calculation with fully connected networks. One example could be considered for a simple operation like edge detection, if a mathematical operation has to calculate and input with 320 x 280 the CNN will calculate it with 319 x 280 x 3 = 267.960 operations on the other hand with fully connected networks the same operations would be completed 320 x 280 x 319 x 280 = 8 billion operation approximately [25]. The advantage of the CNN is not limited to time complexity but also the flexibility by accepting with getting input with multiple dimensions and different sizes.

Incidentally, to the mentioned performance enhancer the CNN operations can be completed faster with post operations like dimension reduction. The well-known stride and pooling techniques are used mostly for this purpose. Stride is a component of CNN that modifies the amount of movement of a CNN filter (kernel) on a frame (on image pixels). In more simple terms, it tells the kernel how many steps will be taken when traversing an image. Pooling is a component to reduce the spatial volume of an image. The pooling is generally applied after applying the convolution. Pooling has various types like maximum, average, stochastic, spatial pyramid, and def-pooling, the most used ones are maximum pooling and average pooling. These are obvious with their names, maximum pooling gives the network the maximum value of pixels and the average method gives the average of the values for a corresponding pixel.

2.3.2.2 2.5D CNN Architectures

2D images have two dimensions, x, and y, for a flat view of height and width information. In addition to the 2D image 3D images have an additional dimension z-axis for the depth information. For a sensor, a true 3D image provides six degrees of freedom for an object. So while grabbing the position of an object at a time with any conventional image-capturing technique provides a visual relationship between the object and the view side of the camera but can not provide the rotational information. So these images are called 2.5D images [26].

Since the first implementation and publication mainly focused on 2D objects the CNN is mainly designed for 2D objects. The 2D colored 2D objects have three channels called RGB; Red, Green, and Blue initials. The success of the CNN algorithm with the 2D colored images direct the researchers to use this method on the RGB-D data sets. This dataset includes the data for the 4th channel for the depth information [27]. These images differ from the 3D images by not including all the data for an object for all the axis besides the image having a fixed value for the z data. As being different from 3D images it has a fixed z data and all the other axis of the images change according to the fixed depth value [28]. The images are called 2.5 D images, and by following the almost same procedure as 2D CNN Roth et all used axial, coronal, and sagittal planes instead of the RGB channels.

2.3.2.3 3D CNN Architectures

The basics of 3D CNN resemble most parts of 2D CNN but are separated in the mean of the input data type. The 3D convolutional networks are generally used in feature extraction for the spatiotemporal or spatial features of 3D images [29]. The views on the type of human recognition of the objects in images are interchangeable among researchers and their research is shaped by this perspective. Traditionally, the 3D model for recognition is divided into two geometry-based techniques [30] and view-based techniques [31]. When focused the 3D-based CNN studies can be divided into three parts:

1. Volumetric part refers to calculating the depth of the object by various tech-

niques. The data used in this method is generated with recording by the stereo, depth-based cameras, or the rendering of an object.

2. View-based parts are provided by descriptors that describes a 3D model by traversing over a dataset of 2D projections in the aspect of "How it looks?"

3. Motion-based part refers to the approach in which the 3rd dimension is provided by the time intervals between frames [26]

Overall the parts of the 3D CNN object recognition come with a high computational cost for the network. To overcome this drawback pseudo networks could be used which use different kernel sizes for spatial and temporal-spatial features respectively 1x3x3 and 3x1x1 instead of using a standard kernel size of 3x3x3 which helps to decrease the complexity.

2.3.2.4 Residual Network

The deep networks Residual Network (ResNet) was proposed by He et al [32]. The change that occurred with this improvement was a big change for the CNN architectures with ResNet introducing an efficient way for deep networks. The winner of the competition 2015-ILSVRC, ResNet introduced 152-layered deep CNN is shown on Figure 2.3

ResNet performed with less error on the image when compared to 34-layer plain Net with 50, 101, and 152 layers. Additionally, it gained %28 improvements over the COCO dataset, considered the benchmark dataset for image recognition problems [33]. This performance of ResNet in localization and image recognition indicates that representational depth has very deep importance for visual recognition tasks [1].

2.3.3 YOLO

You Look Only Once (YOLO) [34] is a nearly real-time, single-stage object detection algorithm that differs from the CNNs in the way it produces the predictions of the objects both by approach and number. As given that the Fast R-CNN generates

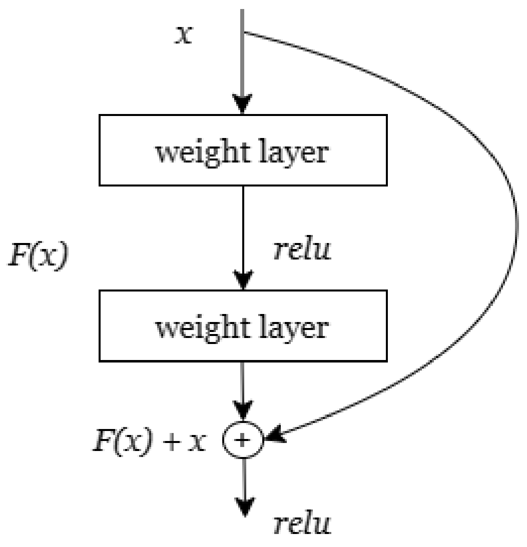


Figure 2.3: A sample residual block of ResNet [1].

thousands of region proposals with the selective search the number of predictions and the way it flows is not a very good fit for real-time applications, unlike YOLO which only predicts an adequate number of BBs for each image. The abstract way of how it works is that YOLO divides an image into a number of grids and makes the predictions if the object's center is placed in that cell. That is a regressing of the 2D image in a way to detecting to the object.

After the introduction of the YOLO, another algorithm SSD was proposed by Lie et al [35] that embraces some of the approaches of RPN, YOLO, and Faster R-CNN to detect multiple object class as single stage detector. As following the newer and different versions of the YOLO developed over time and their general architecture comparison could be seen in Table 2.1.

Table 2.1: Structure compassion of YOLO versions

	YOLOv3	YOLOv4	YOLOv5
Neural Network Type	Fully Convolution	Fully Convolution	Fully Convolution
Backbone Feature Extractor	Darknet-53	CSPDarknet53	CSPDarknet53
Loss Function	Binary Cross Entropy	Binary Cross Entropy	Binary Cross Entropy and Logits Function
Neck	FPN	SSP, PANet	PANet
Head	YOLO layer	YOLO layer	YOLO layer

Additionally, the YOLO object detectors approach a problem like it is a regression problem. The main idea of the YOLO is to use the whole image to provide single output via the network with classes and bounding boxes. It takes the images and divides them into predefined grids like SxS to form the bounding boxes. After processing these with a series of layers a bounding box is specified around the object.

The bounding box is formed according to the highest confidence for an object. Overall, the structure is a relatively simple convolution layer (53 convolutional layers), max pools, and YOLO layers. In this problem, the part where the final classification takes place YOLOv3 uses a logistic classifier with binary cross-entropy in which a class can be assigned to a single object detected which can help to overcome when the objects are overlapped or occluded.

2.3.3.1 Bounding Box Prediction

Object detectors, detect objects by generating proposals over candidate objects and processing the proposals. The YOLOv3 model has two phases mainly, feature extraction and bounding box prediction. After the feature extraction phase is completed the prediction phase starts. In the prediction phase, the network uses several layers as the convolution layer, up-sample layer, skip connection, and with 2 strides for the down-sample layer. YOLOv3 differentiates by not using the pooling layer instead additional convolutional layers are used to down-sample feature maps. Because the usage of the convolutional layer to down-sample feature maps prevents the loss of low-level features that the pooling layer has to exclude. So it can help to increase the ability of the object detector to detect small objects via capturing low-level features by using convolutional instead of max pooling.

The input of the Network is a batch of the images of a defined shape as (n,w,h,c). That recalls n for images as input, w for the width of the image, h for the height of the image, and c for color RGB channel. The numbers are given with the batch of images have to preferred as dividable to 32. But this does not directly restrict the size of the original image and there is no need to resize the images before feeding the network because all of them will be resized according to network input size before feeding the network. Increasing these numbers could increase the accuracy of the model after training but it brings complexity cost with itself. In this study, the width and height input (network input size) has been given as 416x416 which is also dividable into 32. This value also could be stated with an aspect ratio but it would be used in more heterogeneous data sets because of the variety.

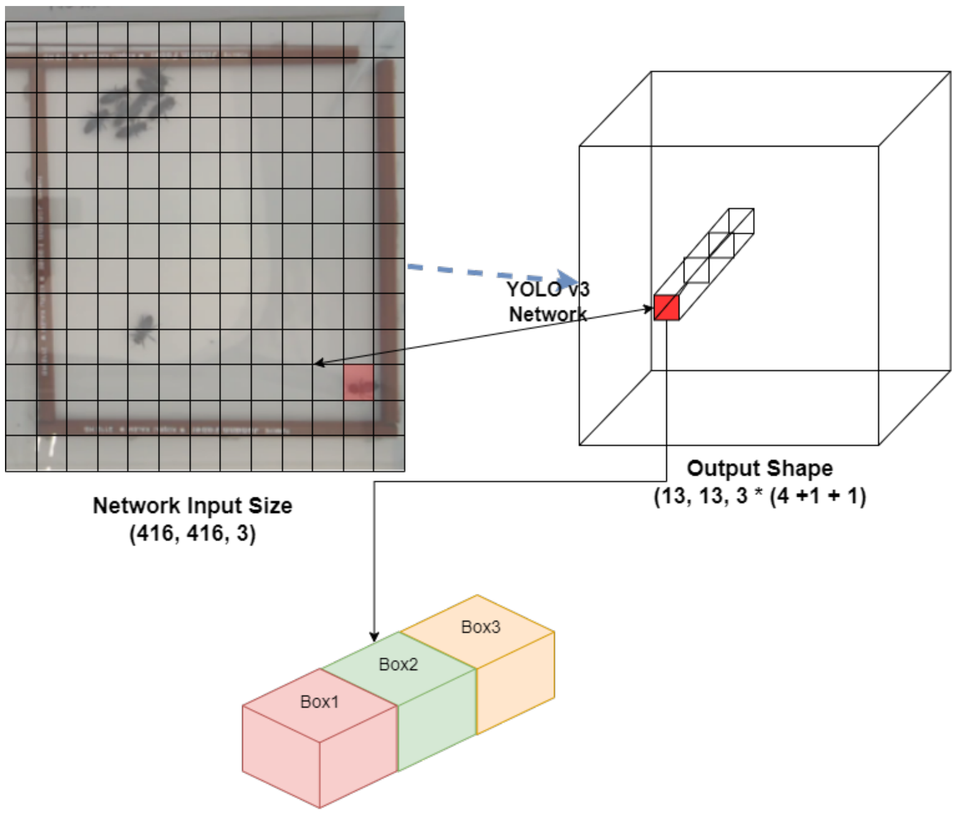
YOLOv3 detects objects at three different scales and three separate positions in the Network. These positions on the layers are 82, 94, and 106. Network down-samples input images for fast close to real-time predictions by strides. The stride factors input is 8, 16, and 32 at these separate positions. So strides of the Network can provide a state with a smaller sized image than the input layer image to the output layer. When the down-samples feeds the stride is selected as respectively 32, 16, 8 the input size of the network will be 416 x 416, and the output size is 13 x 13, 26 x 26, and 52 x 52. These sizes of the strides with 3 scales are used to assign different-sized objects 13 x 13 is responsible for the detection of large objects, 26 x 26 for medium, and 52 X 52 for small objects. That is also an explanation as to why the input network size has to be dividable into 32 which also ensures it is dividable to 16 and 8 as well. That responsibility sharing with three layer detection technique supports YOLOv3 to perform better for object detection when compared with previous versions.

The YOLOv3 works differently with the aspect of the kernels for these three separate places. For every downsampled input image 13 x 13, 26 x 26, and 52 x 52 the network uses a one-by-one kernel for each of them. This leads to having the same spatial dimensions on feature maps. The shape of the next kernel in the YOLOv3 Multi-Scale Detector is calculated as:

$$k = n \times n \times (b \times (4 + 1 + C)) \tag{2.1}$$

The feature map extracted has cells according to network input configuration. YOLOv3 predicts 3 BB from the cells of the feature map. The representation of this prediction could be seen in Figure 2.4

As default YOLOv3 predicts 3 BB for every cell on the feature map. These BB parameters are height of the bounding box t_h, width t_w, coordinates of the BB (t_x,t_y), probabilities of belong a class p_i and objectness score t_0. The value for t_j is the confidence for a BB containing an object in it. After completing this phase on the



YOLO v3 Network
Network Input Size (416, 416, 3)
Output Shape (13, 13, 3 * (4 +1 + 1)
Box1
Box2
Box3

Figure 2.4: YOLOv3 predicting the bounding boxes

NMS phase the final box dimension of the BB is calculated. The BB confidence has to be calculated before moving the prediction phase.

Since the YOLOv3 predicts 3 BB, which can be seen in figure 2.5 for every cell of the feature map, the next step is to choose the cell that contains the center o the object. During the training phase, there is one ground truth BB responsible to detect a single object. That BB has its own center point on a specific cell. So it is important to detect which bounding box it is. To accomplish this YOLOv3 predicts 32,16 and 8 strides at 3 separate scales. So in that stage, which cell has the center coordinates of the object is the cell responsible to detect the object.

While predicting the BB in YOLOv3 prior BB which is called also Anchor Boxes are used. An anchor Box is a set of predefined BBs that has a certain height and width. These boxes are reference points for object detection especially with the feature to calculate real width and height for predicted BB. As the total of 9 anchor boxes is used for each box with different scales. While using these anchor boxes it is used

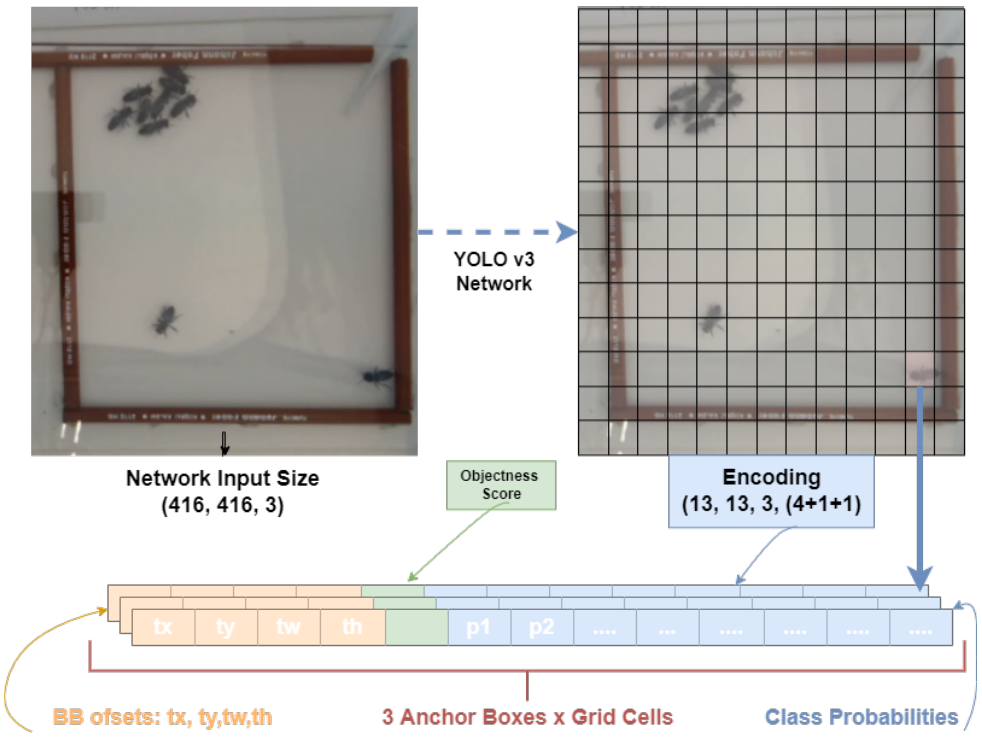


Figure 2.5: YOLOv3 output layer state with parameters

according to the scale as respectively fits 3, second 3, and third biggest anchor boxes for scale 1, 2, and 3. That means for every output layer every scale of the feature map predicting 3 BB by using the 3 anchor boxes. While determining the BB YOLOv3 uses k-means clustering for the 9 clusters and 3 scales arbitrarily and divides the clusters evenly for the scales [36].

The numbers of BB for scales are divided:

Scale 1 : 13 X 13 x 3 = 507 bounding boxes

Scale 2: 26 x 3 = 2028 bounding boxes

Scale 3 : 52 x 52 x 3 = 8112 bounding boxes

The total number of bounding boxes is 10647.

To predict the height and width of the BB, YOLOv3 calculates and formats the of-set of the BB using the formula [36] and the representation of the state o the figure 2.6.

The output of the network is t_x,t_y,t_w,t_h and from this output t_x,t_y processed with sigmoid function and addition with c_x,c_y that top left center coordinates of the grid

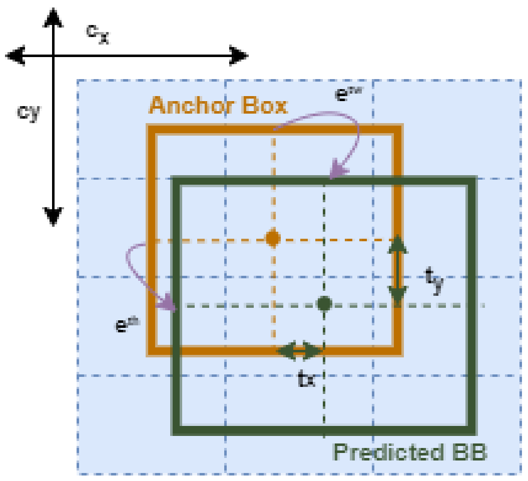


Figure 2.6: YOLOv3 output layer state with parameters

it is transformed to b_x,b_y,b_w,b_h. The size of the BB is achieved via multiplying the anchor box size with the network output t_w,b_h as the power of as of-set.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

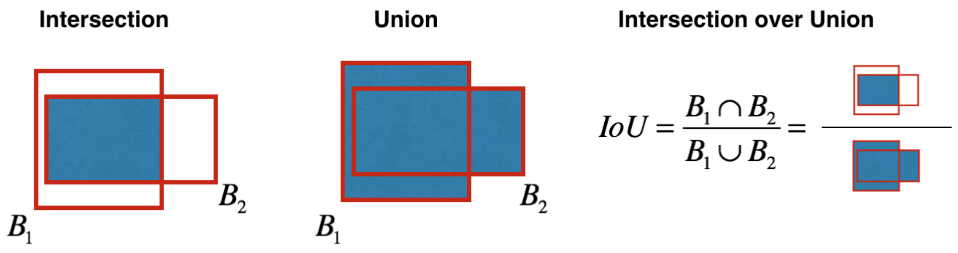


Figure 2.7: YOLOv3 Intersection over Union [2]

After predicting all candidate BB for every scale, the YOLOv3 presents over thousands of the BB. That rise the need to choose one bounding box and eliminate the rest of them. Since the Bounding Boxes have boundaries not only single points YOLOv3 uses IoU instead of spatial distances. IoU is a Jaccard Index-based measurement that coefficient of similarity for two data [37]. YOLOv3 uses the same concept as the IoU and measures the intersecting area between predicted bounding box B_p and ground truth bounding box B_{gt} divided by union area which can be formulated and seen in Figure 2.7:

After passing the values and calculating the ratio of overlap the IoU score is. The score is compared with the given threshold (θ), t, if $IoU \geq \mathrm{t}$ then the detected object is correct else the detection is considered incorrect and discarded.

2.3.4 Kalman Filter

Kalman filtering is a mathematical method for estimating the state of a system from a series of incomplete or uncertain measurements. The method is named after Rudolf Kalman, who first described it in 1960. Kalman filtering is widely used in control systems, navigation systems, object tracking, and other applications where accurate state estimation is critical.

There are two main types of Kalman filters: continuous-time filters and discrete-time filters. Continuous-time filters are used when the system and measurements are modeled as continuous-time signals, while discrete-time filters are used when the system and measurements are modeled as discrete-time signals. The latter one is widely used in practice, so it is also known as the Discrete Kalman filter (DKF).

One of the most common applications of Kalman filtering is position estimation [3]. In this application, a Kalman filter is used to estimate the position of an object based on a combination of sensor measurements (such as GPS or accelerometer data) and a model of the object's motion. The filter uses a combination of prediction and correction steps to estimate the most likely position of the object at any given time, taking into account the uncertainty in the sensor measurements and the dynamics of the object's motion which can be seen in Figure 2.8. The Kalman filter can also be extended to estimate the velocity and other state variables of the object.

The basic principle behind the Kalman filter is the prediction-correction step. In the prediction step, the filter uses the current state estimate and the system dynamics model to predict the state of the system at the next time step. In the correction step, the filter uses the new measurements to update the state estimate, taking into account the measurement uncertainty.

The Kalman filter is a powerful tool for state estimation, but it does have some limitations. One limitation is that the filter assumes that the system dynamics and mea-

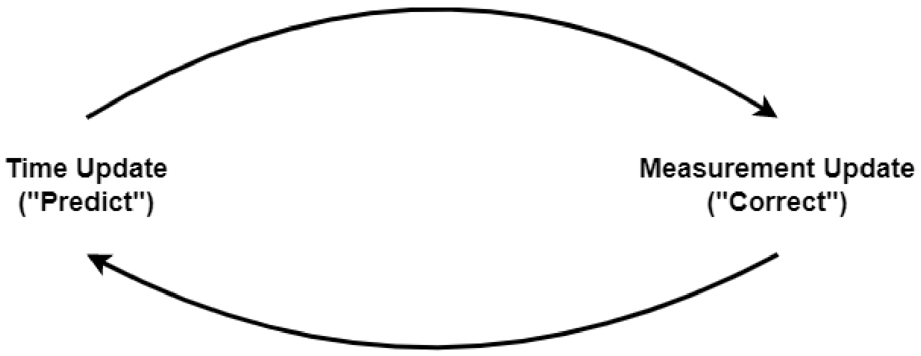
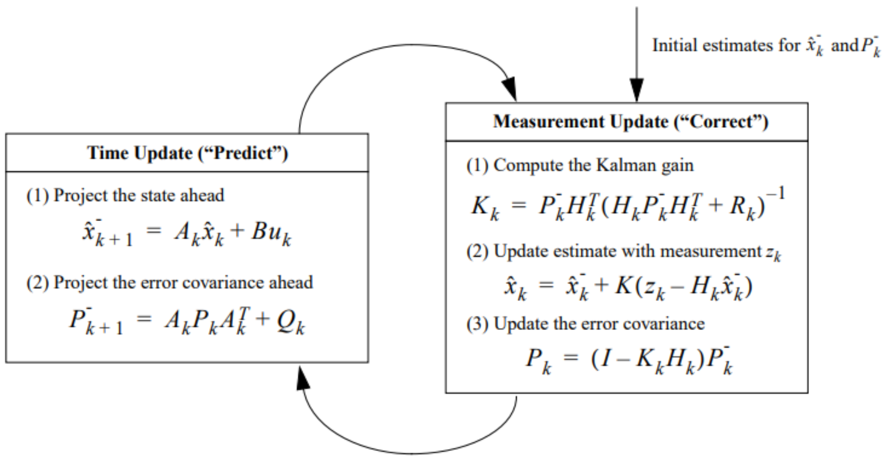


Figure 2.8: Discrete Kalman Filter predict correct cycle

surement models are known and linear. In practice, these models may not be known exactly or may be nonlinear. In these cases, modifications to the Kalman filter, such as the Extended Kalman filter or the Unscented Kalman filter, can be used to handle nonlinear systems.



Initial estimates for $\hat{x}_k^-$ and P_k^-
Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_{k+1}^- = A_k\hat{x}_k + Bu_k$
(2) Project the error covariance ahead
$P_{k+1}^- = A_kP_kA_k^T + Q_k$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^-H_k^T(H_kP_k^-H_k^T + R_k)^{-1}$
(2) Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K(z_k - H_k\hat{x}_k^-)$
(3) Update the error covariance
$P_k = (I - K_kH_k)P_k^-$

Figure 2.9: Complete Kalman Filter process [3].

The measured variables do not occur in a very stable case except for the measurement environment which established almost full control over the environment state. With in the not fully controlled environment, the variables do not follow a stable state. For moving object detection, especially in like this study, the objects have to be tracked and they do have variables to be measured during the experiment phase like position and velocities. The overall recursive process can be seen in Figure 2.9

A moving object is in a state of motion and this motion could be computed by estimating the position of the detected moving object via extracting the object boundaries [38]. During the motion tracking the motion of the object can be elaborate by obtaining the vectors from a sequence of a 2D image from a sequence of frames from the same video [39]. The Kalman filter uses the vectors by measuring the variables for positions and velocity of the moving object for a sequence of frames for a video. The Kalman Filter basic approach is assuming the velocity of the object is stable if the other variables are neglected. The Kalman Filter position estimation method has been used in this study and explained in this chapter by assuming the velocity of the honeybees is stable which actually not happening when the pests are used over the test group [40].

2.3.5 Hungarian Algorithm

The Hungarian Algorithm, also known as the Kuhn-Munkres algorithm, is a combinatorial optimization method that can be used to solve the assignment problem in a computationally efficient manner. In the context of object tracking, the Hungarian Algorithm can be utilized to assign a unique object to each tracking hypothesis, or vice versa, in a multiple object tracking scenario. The assignment problem can be formulated as a cost matrix, where each element represents the cost or likelihood of a specific object-hypothesis pairing. The goal of the algorithm is to find the optimal assignment, such that the overall cost is minimized.

The Hungarian Algorithm operates by iteratively selecting the smallest element in each row and column of the cost matrix and subtracting it from all other elements in that row and column. Next, it examines the remaining matrix for any zeroes and assigns them to the corresponding object-hypothesis pairing. If the matrix contains no more zeroes, it proceeds to create a set of lines covering all the rows and columns that contain at least one assigned element. The algorithm then finds the smallest element not covered by these lines and subtracts it from all uncovered elements. The process of creating lines and subtracting the smallest uncovered element is repeated until a complete set of lines is formed, at which point the algorithm terminates and the optimal assignment is determined.

In object tracking, the Hungarian Algorithm can be applied to assign object detections to track hypotheses, or to assign track hypotheses to object detections, depending on the specific implementation. The algorithm has the advantage of being able to handle a large number of objects and hypotheses and provide a globally optimal solution in polynomial time. It is widely used in various object-tracking applications such as video surveillance, robotics, and self-driving cars.

It is important to note that the Hungarian Algorithm can only be applied to the case when the number of objects is equal to the number of hypotheses. When the number of objects is different from the number of hypotheses, we need to use methods such as the Munkres-Kuhn Algorithm.

2.4 Related Works

The related work section focus on image labeling works for computer vision applications. Since computer vision technology has very rapid improvements with the increasing computation power the need for labeled images has increased. The need draw attention to the ease of the burden of the labeling process which indicates the Image Labeling Systems (ILS). The development of machine-learning-based computer vision systems has a big share in aspects of time and money [41].

2.4.1 Image Labeling Process

The fundamental idea of image labeling is mapping visual features with donated and spatial labels that explain the content of the image clearly [42]. The labeling process was conducted by human annotators by hand and to ease the burden ILS is used to exclude human interaction with the labeling process. The purpose of ILS usage can be summarized as:

1. Increasing the labeling performance of the annotators,
2. Supporting the creation of a large, labeled dataset,
3. Ensuring the labeling quality by decreasing annotators' revision mistakes.

ILS frameworks provide varied ways of quality assessments for class or instance interpretation errors, including similarity errors that can cause under or over-representation and create bias. Widely used methods comprise annotating the same image multiple times, and voting of multiple annotators [43].

2.4.2 Image Labeling Software

A wide variety and numerous object labeling software exist in literature. Some of these ILSs are in use the current date and some others are outdated. The overall number of the ILS is around 58 [42] according to one of the widely conducted surveys on the ILS. It is not very practical to mention all of the ILS in this study, so only some of them have been introduced here according to these criteria: public availability, development purpose, and universality. According to this evaluation, the subsetted ILSs are shown in table 2.2.

Table 2.2: Image labeling software comparison

Tools	Year	Annotation	Detector	Tracker	Multiple Domains	Label Quality	Open Source
LabelME [44]	2008	Linear Interpolation	No	No	Yes	Yes	Yes
VIPER-GT [7]	2008	Linear Interpolation	Yes	No	Yes	Yes	Yes
GTVT [9]	2009	Tracking Algorithm	Yes	Yes	No	No	No
GTTool [8]	2012	Tracking Algorithm	Yes	Yes	Yes	No	Yes
VATIC [6]	2013	Linear Interpolation	No	No	Yes	Yes	Yes
Labelimg [5]	2014	Linear Interpolation	Yes	No	Yes	Yes	Yes
VIA [45]	2016	Tracking Algorithm	No	Yes	No	No	No
Web Annotation [11]	2018	Linear Interpolation	Yes	No	No	Yes	No
FAST-GT [10]	2019	Tracking Algorithm	Yes	Yes	Yes	Yes	Yes
Image Tagger [12]	2019	Linear Interpolation	Yes	Yes	Yes	Yes	No
Honeybee Labeler	2022	Tracking Algorithm	Yes	Yes	Yes	Yes	Yes

The ILS on the table mostly offers two features to ease the burden of labeling with label propagation and linear interpolation. Label propagation is useful for still images of videos while linear interpolation is more helpful if the objects are moving over the frames of the video. Because label propagation has the functionality to label an already labeled object in the following frames and linear interpolation can adjust the position of the object in consecutive frames.

Some of the ILS have the feature to detect and track objects to be labeled. That feature is very helpful if the subject database contains multiple object classes and moving objects with a huge number of images. That feature provides a pre-analyze of the image before labeling and labeling the object for the user to revise it if necessary.

The multiple domains are also an important component of the ILS since some datasets are not evenly distributed or configured which could need crowdsourcing applications.

Label quality is the last but may be one of the most important features of the ILS. Since the labeled data will be used in use of training a model, the success of the model depends on the quality of the labeling. Some of the ILS in the table has the feature to check the quality of the labeling process and some of them have not.

2.4.3 Labelme

Labelme is an open-source annotation tool written in Python to ease manual image polygonal annotation for computer vision works. The GUI of the label me can be seen in Figure 2.10

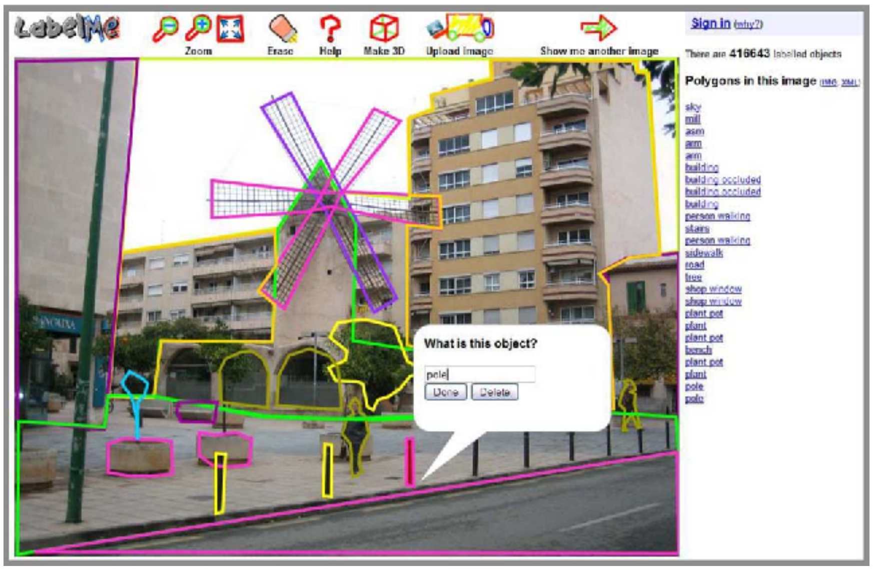
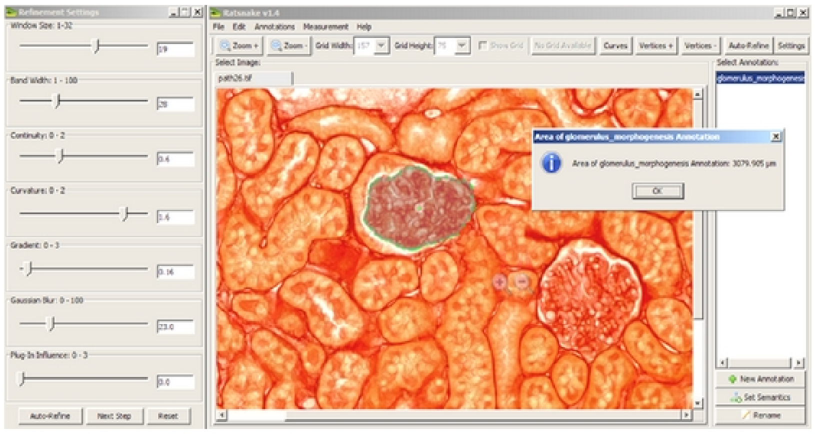


Figure 2.10: Labelme GUI photo [4].

Labelme can create shapes, polygons, circles, rectangles, lines, and points. Also, it could export the labeling job as JSON files. It has features to convert annotations to PASCAL VOL. Current versions of Labelme is not supporting YOLO and COCO formats but an additional python script provides to convert the COCO format.

2.4.4 LabelImg

Labeling is a medical purposed developed in Java ILS. It has been integrated into Retsnake as a coded plugging. Due to it being specifically developed for medical purposes did not have very common usage.



Area of glomerulus_morphogenesis Annotation: 3079.905 μm

Figure 2.11: Labelimg GUI photo [5].

The features of this ILS have been mentioned in the table of ILS before. The GUI can be seen in Figure 2.11 on which a kidney biopsy image analysis has been implemented.

2.4.5 VATIC

VATIC is an open-source and interactive ILS developed with academic research study. The abbreviation of VATIC stands for the "Video Annotation Tool From Irvine, California". The purpose of the tool is to build a huge video dataset that could be used on the cloud.

The main feature, it had automatic quality assurance for good annotations and has a cross-domain crowdsource feature that increases the usage of this ILS. 2.12

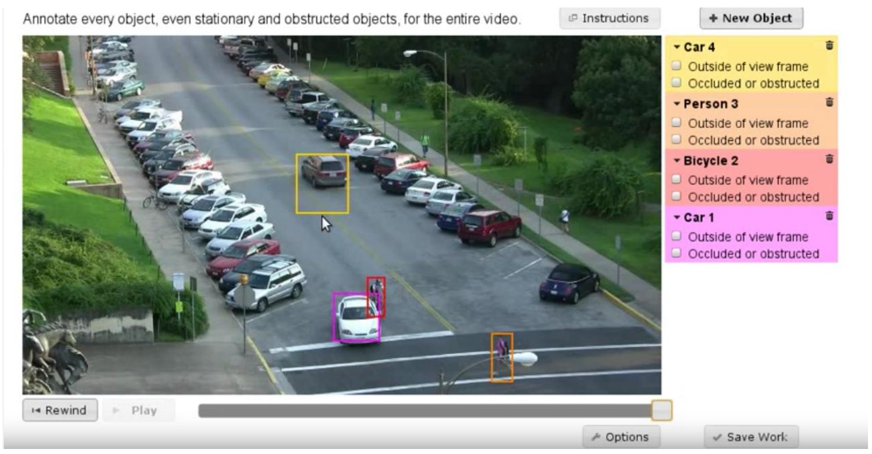


Figure 2.12: VATIC GUI photo [6].

2.4.6 ViPER-GT

ViPER-GT is an abbreviation of a tool named "Video Performance Evaluation Resource Kit's Ground Truth". The main purpose of this tool is to compose videos for use as ground truth. It has varied functions to create ground truth from videos including the metadata and the keywords of the video databases. The GUI of the tool can be seen in Figure 2.13

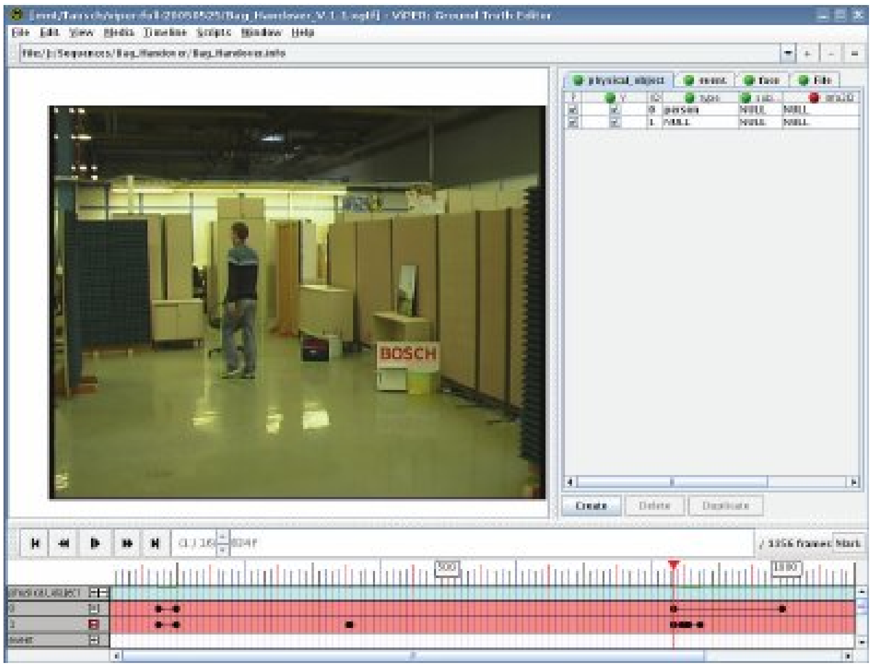
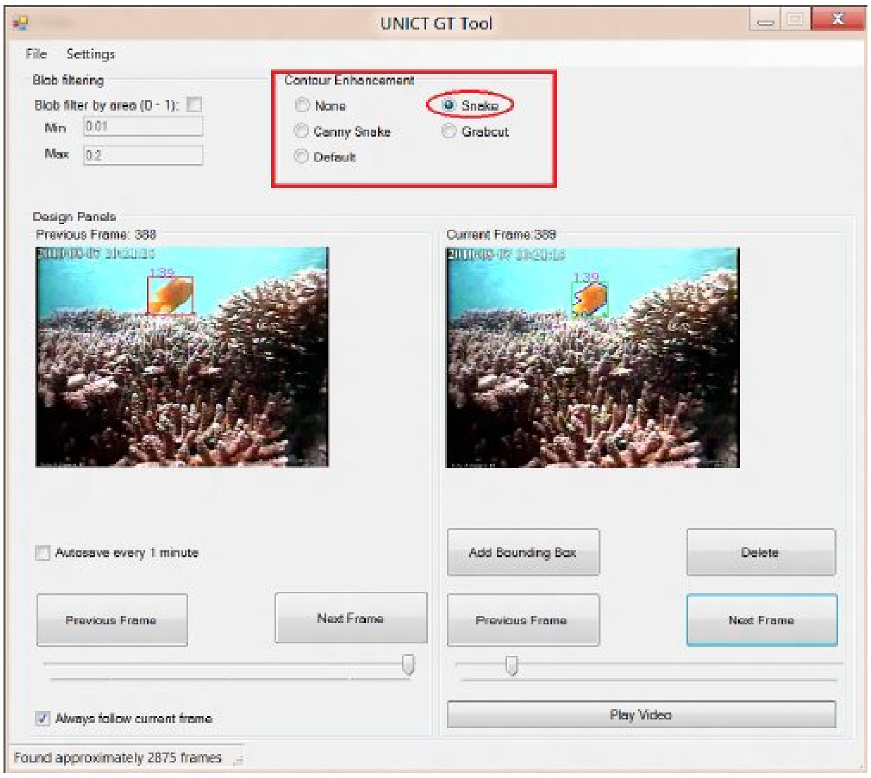


Figure 2.13: ViPER-GT GUI photo [7].

This tool supports different metadata formats like GTF, RDF, and a list of objects that are also called descriptors. The descriptors refer to a composed of attributes that can describe an object event or other targets. The current set of descriptors has various features for the target object like human faces, moving people, text, and scene cuts.

2.4.7 GTTool

GTTool is developed to generate ground truth data for object detection, object tracking, and object recognition applications. When it is compared to ViPER-GT it provides improvements on the user experience side like hotkey, shortcuts, dragging, and dropping. Also, it includes some of the computer vision algorithms to automate the object detection and counter-detection process across the frames.



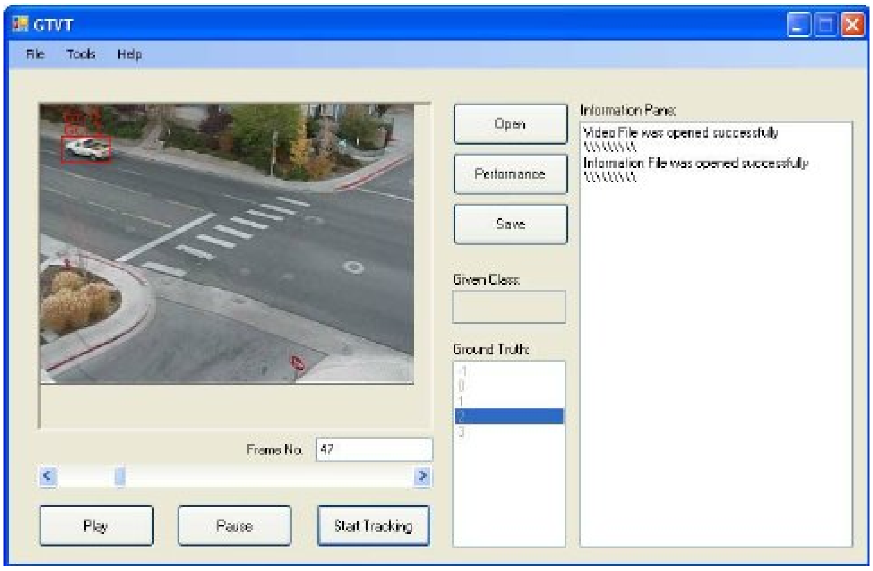
Blob filtering
Blob filter by area (0 - 1): ☐
Min 0.01
Max 0.2
Contour Enhancement
None
[✓ Snake]
Canny Snake
Grabcut
Default
Previous Frame: 388
Current Frame:389
☐ Autosave every 1 minute
☑ Always follow current frame
Found approximately 2875 frames

Figure 2.14: GTTool GUI photo [8].

The GUI of the tool can be seen in Figure 2.14. The developed tool provides more functions than its predecessor ViPER-GT by decreasing the labeling time and keeping the ground truth quality as high as it can.

2.4.8 GTVT

The GTVT stands for the "Ground Truth Verification Tool" which has been developed with the aim of reducing human interaction while supporting human and computer interaction concepts in labeling. The GUI of the tool can be seen in Figure 2.15



Video File was opened successfully

Information File was opened successfully

Figure 2.15: GTVT GUI photo [9].

The GTVT is not starting the class or labels for an object itself instead user defines the class for the ground truth over the created BBs. GTVT copies this labeling to the following frames if the user does not need a revision on the label then it is followed by the tracking algorithm otherwise user interaction is needed. The tool is claimed to reduce the time greatly while labeling but did not provide any results of the user and statistics yet.

2.4.9 FAST-GT

The FAST-GT is a tool that has been developed for the semi-automatic generation of ground truths for object tracking. It is a generic framework that can have different implementations.

The implementation of the FAST GT tool to generate the ground truth for a video sequence containing faces moving in a frame can be seen in the Figure 2.16. The

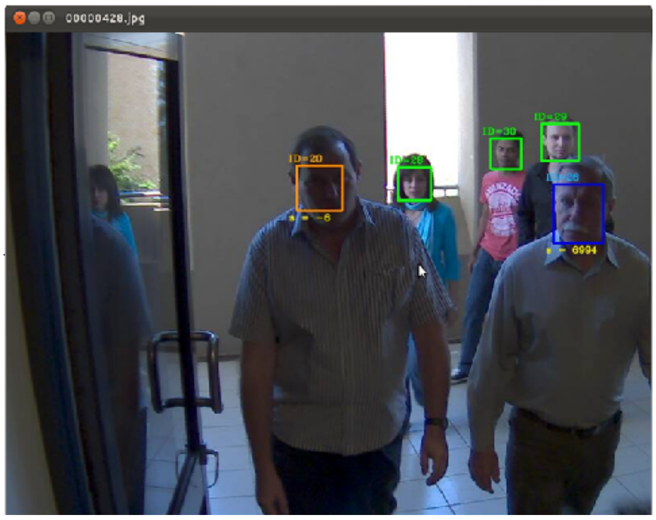


Figure 2.16: FAST-GT result photo [10].

main goal of the study is to decrease the annotation time notably but does not reassure the quality of the ground truths.

2.4.10 Web Annotations

The web annotation application is a web-based image annotation application developed with the purpose of constituting training for machine learning models. The tool is an open-sourced application and also open to usage in crowd-sourcing labeling.

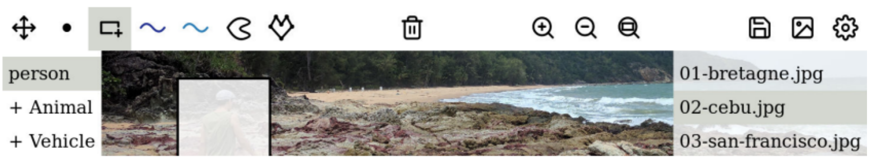


Figure 2.17: Web Annotations GUI photo [11].

The GUI of the web annotation tool can be seen in Figure 2.17. This application is able to work with the Amazon Mechanical Turk services like the LabelMe and Ours labeling tools.

2.4.11 ImageTagger

The ImageTager tool is developed for creating a large datasets that are used in RoboCup Soccer League. The GUI of the ImageTager can be seen in Figure 2.18. ImageTager is an browser-based tool that has capabilities for image labeling, annotation verification, user profile management, label and image categories definition.

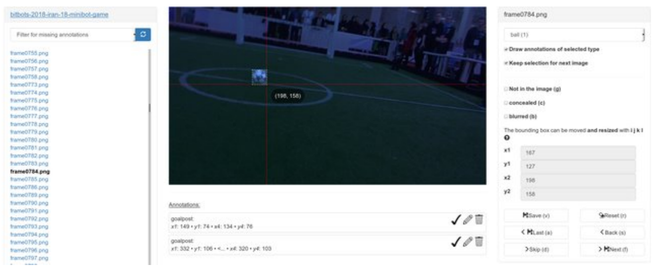


Figure 2.18: ImageTager GUI photo [12].

The tool is developed with python using the web framework Django. The tool has capabilities for manual and automated labeling operations. The range of usage of this tool was soccer context-based and if it could be developed further it has the potential to be used in other contexts.

2.4.12 Conclusion

The developed ILS frameworks can function as web-based or desktop-based, depending on their intended use. Some ILS are designed for generic applications, while others are developed for specific purposes. Additionally, these frameworks have been evaluated based on the presence or absence of detectors and trackers, as well as their label quality assurance features, which may be considered the most important evaluation criterion.

Although most ILS claim to provide a significant reduction in the time required for image labeling, the papers included in this study did not present any quantitative results to support this claim. Furthermore, many of these papers did not address the trade-off between the time savings and label quality.

The general approach of ILS is to alleviate the burden of annotation through reducing direct human involvement in manual labeling. Instead, they advocate for automatic or semi-automatic labeling methods.

CHAPTER 3

METHODOLOGY FOR HONEYBEE DETECTION, TRACKING, AND LABELING PROCESSES

In this chapter, we present our proposed process pipeline for honeybee detection, tracking, and labeling as part of our framework, which is depicted in Figure 3.1. In section 3.2, we outline the methodology we employed to generate the dataset and conduct experiments. We delve into the detection and tracking methodology in sections 3.5 and 3.6, showcasing both established methods and our novel contributions. Finally, in Section 3.7, we introduce the software we developed for the "Beehive 4.0: Artificial Intelligence-based Approach to Honeybee (Apis mellifera) Observation" project.

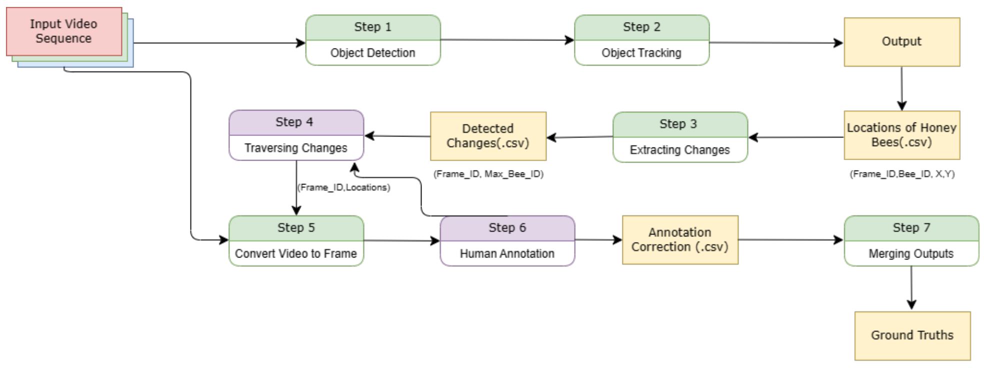


Figure 3.1: The Honeybee Labeling Framework

3.1 Object Detection Methodology

In our study on the object detection of honeybees, we chose the well-known object detector YOLO as the tool to detect honeybees in the Arena Setup. YOLO is a CNN-based object detection method that uses a single pass over the network to propose potential bounding boxes for objects in images and assigns these boxes to the respective object classes.

We used YOLOv3 [36] as our network as it was deemed to be the best fit for the problem definition. This object detection algorithm was developed for an ongoing project, the real-time beehive monitoring system, taking into account factors such as the backbone, loss function, neck, and heads. When compared to other object detectors such as Faster R-CNN [46], YOLOv3 was found to be the most suitable.

We carried out the object detection operations using OpenCV version 4.3.0 and PyTorch [47]. Prior to the study, we used a dataset trained on ImageNet [48] with 40 labeled frames of honeybees to train YOLOv3. Each frame was taken from the Arena Setup, with 20 honeybees separated into 10 for each test and control group.

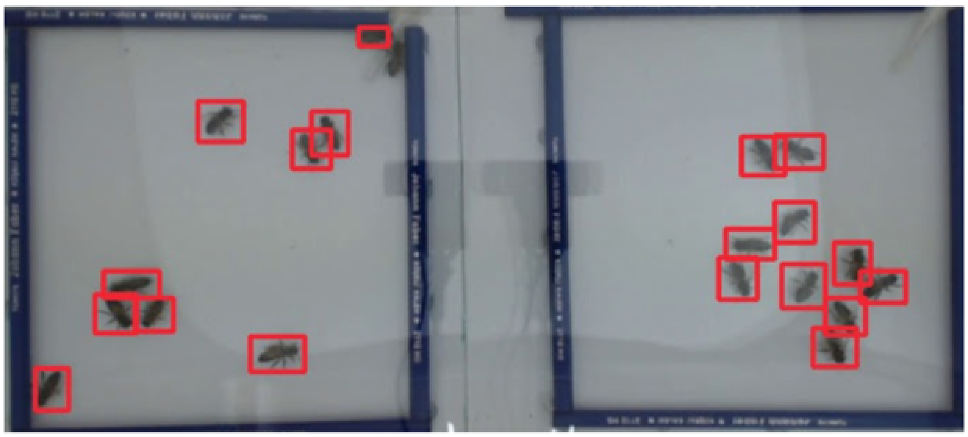


Figure 3.2: A picture of YOLOv3 used to detect the honeybees on arena setup

The Bee Hive 4.0 project aims to design a field setup that is low cost, sustainable, and affordable for beekeepers. To meet this goal, the videos captured during the data generation phase were recorded at a low-resolution using a low-cost webcam configured to 1920x1080/30fps, which would not require excessive computing power and time during field use.

The ultimate goal of the project is to understand the effects of pesticides on honeybee colonies and their behaviors. To achieve this, we had to track honeybees in the colony using an object-tracking method. Before discussing the tracking method, we first aimed to detect honeybees with high precision. To do this, we used YOLOv3 on the dataset.

The results produced by YOLOv3 on the dataset were promising, as seen in Figure 3.2 on a 5-minute test video. However, when applied to videos with more complex honeybee behaviors, YOLOv3 sometimes had difficulties detecting the honeybees, particularly in cases where honeybees formed clusters, as seen in Figure 3.3. This caused miss detection and miss tracking of individual honeybees, which was a problem as the study aimed to measure the effect of pesticides on the honeybees' walking speed, feeding, and social behaviors. Additionally, the interchanging of detection (Track ID) was often caused by the clustering behavior and the interactions of honeybees in the setup.

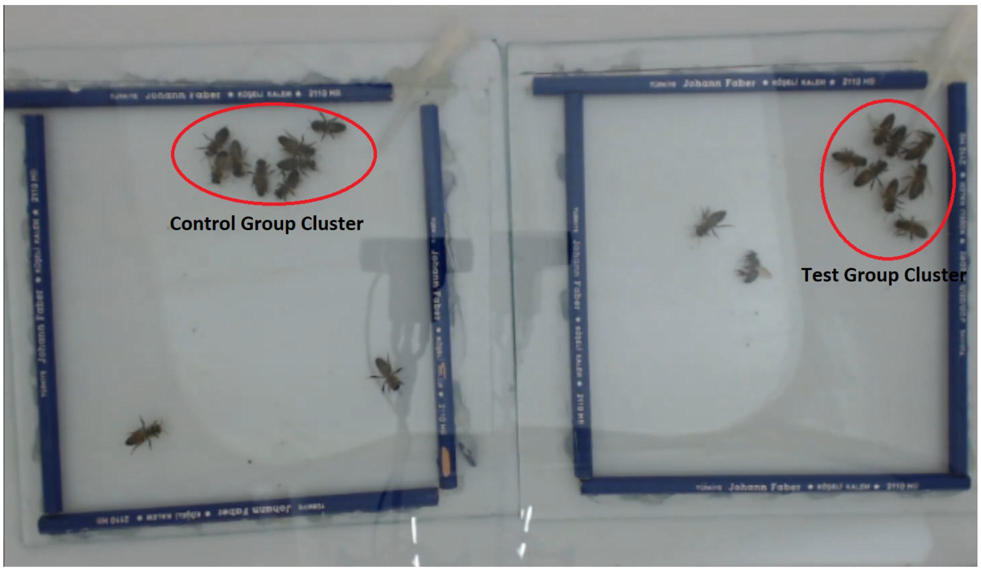


Figure 3.3: Honeybee clusters on the data set

Further analysis revealed that factors related to the natural elements of YOLOv3, such as anchors, bounding boxes (BB), and Non-Maximum Suppression (NMS), were impacting the detection results. The objects in the scenes, particularly the honeybees, are relatively small and as they come closer together, the BB and NMS filtering become crucial in determining which bee is detected, which is a crucial aspect of object detection.

3.1.1 Bounding Box Predictions

In our proposed framework, we utilize the YOLOv3 object detector to produce both accurate and erroneous object detections during the detection process. We classify these detections as True Positives (TPs), False Positives (FPs), and False Negatives (FNs). According to previous research [49], True Negatives (TNs) are not commonly used in object detection frameworks, as they are not relevant to object detection problems.

Specifically, in our case:

1. If a honeybee is detected and is present in the video, then it is considered a TP.
2. If a honeybee is detected but was not present in the video, it is considered a FP.
3. If a honeybee is present in the video but is not detected or is detected but under a threshold score and discarded, it is considered a FN.

In our proposed framework, we evaluate the performance of the YOLOv3 object detector using the concepts of precision and recall. We measure the accuracy of the model in detecting the related objects by computing the ratio of correct detections, which we call precision. To evaluate the model's efficiency in representing the ground truths, we calculate recall.

Our findings reveal that precision and recall are inversely related and can be visualized through a precision-recall curve for different confidence values of the predicted bounding boxes (BB). We observe that when the confidence of the object detector is low, we see a high false positive (FP) score, leading to a high precision score, however, this also leads to a high false negative (FN) score and low recall score. On the other hand, increasing the number of accepted positives results in an increase in recall, but it also results in an increase in false positive (FP) score and a decrease in precision. This trade-off between precision and recall is widely reported in the literature [50].

The precision and recall scores could be calculated as follows:

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3.1}$$

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.2}$$

In this thesis, we designed an object detector to detect multiple small objects in a single frame. After the feature extraction stage, our proposed framework, using the YOLOv3 network, generates bounding boxes (BBs) for the objects, some of which may be false positives.

To eliminate these false positives, we included an optional contour-based elimination schema in our framework. As the video background is static, the separation of background and foreground is relatively simple compared to videos with dynamic backgrounds.

First, we subtracted the background of the image from the current frame, taking advantage of the limited and static movement area of the honeybees. This resulted in a foreground mask that only contained the pixels of the objects in the frame. This approach is particularly useful when the sensor is a static camera.

Next, we updated the background mask during the detection process to detect the contours of individual honeybees and their clusters. For this, we utilized the OpenCV background subtraction and contour detection functions. By considering the continuous color or intensity state along a boundary, we formed a curve on the frame to improve accuracy. Binary images were used for improved contour detection, where the objects were detected as white and the background as black.

Having successfully detected the contours of the objects, we eliminated false positives by discarding points outside of the contour area. While the background is contrasted with the honeybees, false positives are rare due to the controlled environment. However, we observed that elements such as dots or dirt on the frame may result in increased computation time and labeling effort.

Finally, to further eliminate filtered BB proposals, we used Non-Maximum Suppression (NMS), a conventional method in this study. NMS selected the object detection results using the Intersection over Union (IoU) metric, repeatedly selecting the proposals with the highest confidence score and eliminating proposals with a higher IoU with the selected one above a threshold value. The steps are as follows:

1. Select one prediction that has the highest confidence score and pop it from the prediction list and push it to the final prediction list keep.
2. Compare the added prediction with all the predictions present in the prediction list. Calculate the IoU of this prediction with all other predictions. If the IoU is greater than the threshold for any prediction remove the prediction from P.
3. If there are raw predictions in the prediction list, then go to Step 1 again, else return the list of processed predictions.

Initially, we used NMS to merge multiple BB proposals into one. However, during long-term video analysis, we observed that NMS had a negative impact on the network's performance. The formation of honeybee clusters, which often occurred as a ball, decreased the success rate of detection and tracking, and cluttered the bee's location data. These clusters were created by interactions and communication between individual honeybees. When bees interacted with each other, especially in large clusters, they got so close to each other that some true detection results were eliminated by NMS.

To address this issue, we employed the K-Means clustering method during the elimination process. With the help of a controlled environment, where the total number of objects was known and constant, we created a set of sets consisting of BB clusters by using K-Means clustering with K equal to 20, which was the total number of honeybees on the scene. From this list, we selected the proposal with the highest confidence result from each cluster, thereby eliminating high-confidence BB proposals without suppressing low-confidence BB proposals for the presented objects on the scene.

3.2 Object Tracking Methodology

Multiple Object Tracking (MOT) in nearly real-time is a significant topic for computer vision [51]. MOT objects can range from cars, pedestrians, animals, and many more. However, the focus of MOT has mainly shifted to the subdomain of Multiple Pedestrian Tracking (MPT) due to the non-rigid nature of pedestrians and their potential business value. The methods used in MPT can be classified into two approaches: Model Free Tracking (MFT) and Tracking by Detection (TBD) [52].

We found that MFT requires manual initialization to track multiple pedestrians, while the algorithms then track the initialized objects in subsequent frames. In contrast, TBD can perform various scenes using a pre-trained detector without the need for manual initialization of objects on the scene, making it more efficient for individual tracking. The TBD framework, with detectors such as the COCO dataset, achieves a mean average score of 46.9

Given the accuracy and prevalence of using TBD in MPT, we decided to use the TBD method in this study. We needed to analyze the behavior of honeybees both as individuals and as a community under the influence of pesticides, so we conducted the Object Tracking section of this study to track every individual honeybee in the Arena Setup. The tracking-by-detection method was used for this purpose.

3.2.1 Object Tracking

We present a deep learning application for object tracking that generates unique identifications for each object detection and tracks them in subsequent frames. We automatically identify the tracks among a video by interpreting them as a set of trajectories. Figure 3.4 illustrates the merged trajectories of honeybees used in this study.

In this work, instead of using a square surrounding the tracked object, we utilize a honeybee identification code. Figure 3.5 showcases a random state of our tracking approach.

The object tracking concept can be divided into two categories based on the number and variety of objects in the scene, Single Object Tracking (SOT) and Multiple Object Tracking (MOT). SOT focuses on a screen where an object is initialized for manual tracking and only considers the initialized one-class object until it leaves the scene or its detection is lost. MOT algorithms, such as Faster R-CNN [53], Single Shot Detector [35], YOLO9000 [54], R-CNN [55], and R-FCN [56], generally treat this problem

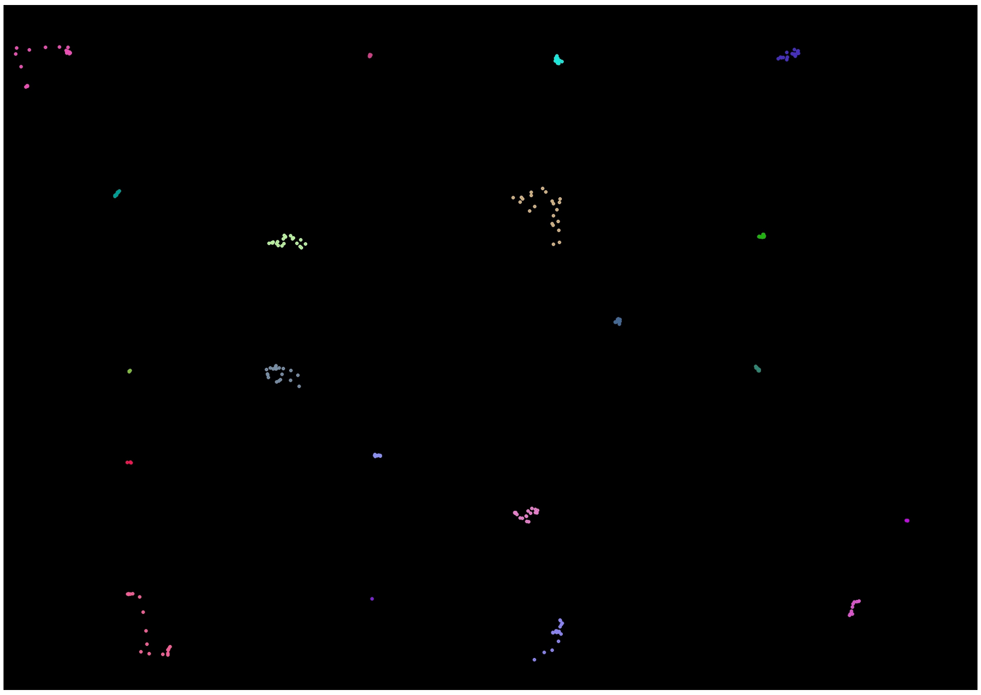


Figure 3.4: Merged trajectories from the arena setup

as an assignment problem and rely on high-quality object detectors. Although there are exceptions, this study chose the MOT method to solve the defined problem.

The MOT algorithms can be further divided into batch and online methods. Batch tracking algorithms utilize future information when detecting the identities of objects in related frames, which leads to better performance. On the other hand, online object tracking algorithms can only use present and past information to predict the objects in the frame. The lack of future information makes it difficult for online tracking algorithms to perform as well as batch tracking algorithms. As a result, online tracking algorithms do not have the opportunity to correct the process by reviewing past values.

3.2.2 Kalman Filter

In this study, the detection of objects, specifically honeybees, is carried out using the YOLOv3 detector. The position of the object is estimated based on the state of past objects in the current frame. To achieve this, the well-known Kalman Filter [57] has been implemented and configured to meet the requirements of the study.

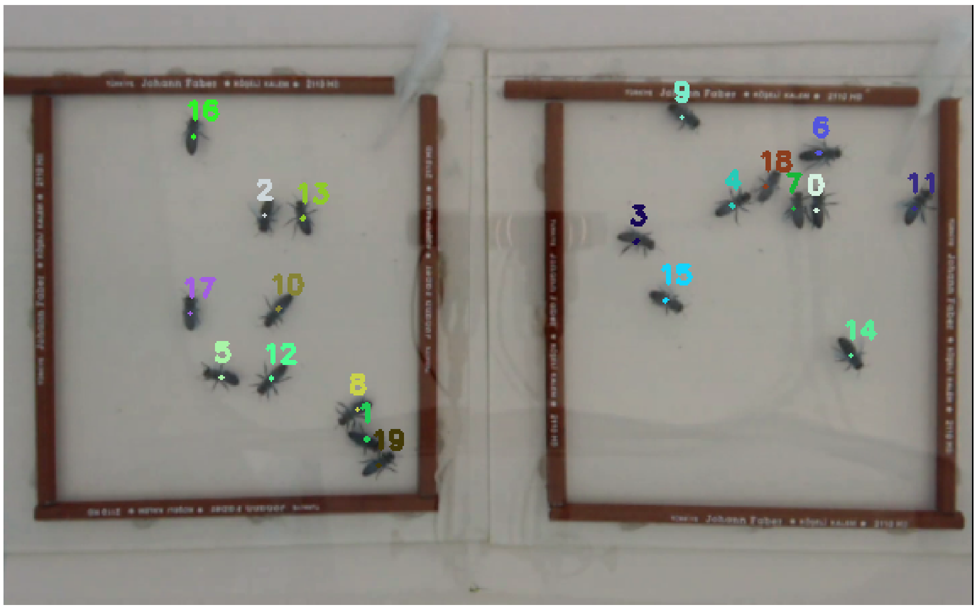


Figure 3.5: State of random frame on tracking state with the indicators printed

Despite the fact that the velocities of honeybees are not constant, which is evident from the generated data set, the constant velocity approach was used in this study. This was due to the limitations of the Arena Setup installation, where the true velocity of the control group and test group could not be accurately determined. The reference points, such as the frame borders of the setup, were constantly changing and the height of the camera was not recorded for each experiment. Despite these limitations, the constant velocity approach was deemed the most suitable assumption for this study.

To minimize uncertainty, the F matrix was used for the $x = [x, y, v_x, v_y]$ set. The YOLOv3 network was applied to the data set, producing results for each honeybee in the form of (x,y) coordinate pairs. The output was then fed into the Kalman Filter to smooth the results and provide the positions of the honeybees for three consecutive frames.

After predicting the positions of the honeybees, the estimated positions were matched with newly detected objects in frames to provide individual-based tracking. The Hungarian Algorithm was used to match these positions with the newly detected objects, as described in [18].

3.2.3 Hungarian Algorithm

We tackle the problem of object tracking by applying the Hungarian Algorithm to match tracklets with observed objects in sequential video frames. The Hungarian Algorithm allows us to assign the correct tracklets to the detected objects by computing the correlation between objects in two consecutive frames. In other words, we use the Hungarian Algorithm to solve the assignment problem of matching the objects in one frame to those in the next frame.

In this study, we employed the Hungarian Algorithm to determine the correct object tracking tracklets and match them with the observed objects. This involved constructing a cost matrix to measure the association between elements in one set with those in the other set.

We initialized a cost matrix by using the Euclidean Squared Distance metric instead of the Euclidean Distance metric. The reason for this choice is that it saves computation time by avoiding the calculation of square roots, making the method more computationally efficient. Moreover, using the squared distance allows us to use only the center points of the tracks, as all other features like size, shape, appearance, and sensor position and background are nearly identical in our experiment.

$$d_{i_k, j_{k+1}} = \sqrt{\sum_{m=1}^{n} (O_{i,k}^{(m)} - O_{j,k+1}^{(m)})^2} \tag{3.3}$$

We utilized the Hungarian Algorithm to iteratively link the detected objects in one video frame to the next. To do this, we considered the Euclidean Squared distance metric as the basis for assigning the cost of linking objects between adjacent frames k and k + 1. The algorithm iteratively carried out the linking process by optimizing the ultimate solution of the distance metric assignments.We used Equation 3.3 to determine the difference between the objects in which the variable O represents the center points of the tracks of objects detected in a video frame.

$$D_{k,k+1} = \begin{bmatrix} d_{1_k,1_{k+1}} & d_{1_k,2_{k+1}} & d_{1_k,3_{k+1}} \\ d_{2_k,1_{k+1}} & d_{2_k,2_{k+1}} & d_{2_k,3_{k+1}} \\ d_{3_k,1_{k+1}} & d_{3_k,2_{k+1}} & d_{3_k,3_{k+1}} \end{bmatrix}, \tag{3.4}$$

We calculated the difference between the identified objects in adjacent frames k and k+1, represented as $O_{i,k}$ and $O_{i,k+1}$, respectively. The assignment matrices generated for a sample three object scene can be seen in Equation 3.4.

$$\begin{gathered} f = Min(\{D_{k,k+1}\}) \\ \{f : \{(O_{1,k+1} \rightarrow O_{1,k}), (O_{2,k+1} \rightarrow O_{3,k})\} | O_{i,k}, O_{i,k+1} \in R^2\} \end{gathered} \tag{3.5}$$

The Hungarian Algorithm maps objects by finding the minimum cost solution from the assignment matrices. The process begins with the initialization of the cost matrix 3.5. We then identify the minimum element in each row of the cost matrix and subtract it from every element in that row to ensure that the matrix does not contain negative values, which would cause issues later on in the implementation.

Next, we repeat the same process of finding the minimum element in each column of the cost matrix until all columns have been completed. After this, we check for the presence of covering zeros in the cost matrix by trying to find a set of zeros that can be covered with the minimum number of lines, which can be either horizontal or vertical. We do this by marking the rows and columns that cover the zeros.

The following step is to make the assignments of the columns and rows. If a perfect matching is found, we assign each row to the corresponding column with a zero. If no more zeros are present, the algorithm returns the assignments made so far. However, if a perfect matching is not found, we find an augmenting path in the residual graph and increase the matching size by one. This process repeats until a perfect matching is found or all zeros have been covered.

When using the Hungarian Algorithm for object tracking, it is important that the number of predicted trajectories and the number of detected objects match. However, in real-world scenarios, there may be instances where the number of detected objects is less than the number of predicted trajectories, such as when an object is temporarily occluded or when the object detector fails to detect an object.

To handle this situation, there are several approaches that can be used. One approach is to use a simple greedy algorithm to assign the remaining detected objects to the nearest predicted trajectories. While this method is easy to implement, it may not be appropriate in all scenarios as it can increase the number of ID switch errors. Another approach is to incorporate additional information about the objects, such as velocity and acceleration, into the assignment process to reduce the impact of missing detections and improve the accuracy of the object tracking results.

A third approach is to use a more sophisticated optimization algorithm, such as the Particle Filter [58], to estimate the missing detections and perform the assignment. This method can provide more accurate results in challenging scenarios, but it may be more complex to implement.

In our case, we chose a low computational complexity solution with fewer ID switches by using the Hungarian Algorithm. In situations where the Hungarian Algorithm does not work, we considered a simple solution of keeping the lost detection track and re-assigning it when a new detection is made. This is accomplished through a reassigning strategy, where the lost track ID is pushed to a track ID stack and, in the event of a new detection, the first step is to match the lost track to the new candidate. This strategy was promising, as the total number of objects in the vision remained constant during the experiment. However, it still faced challenges with concurrent lost tracks and objects.

3.2.4 Object Tracking Evaluation

We evaluated the performance of object tracking algorithms by using various metrics that are standardized by works such as Wu and Nevatia [59], CLEAR MOT [60], and ID metrics [61]. Our aim was to understand the underlying process of the tracking and to project the performance of the object tracking models using these metrics. The "Classical Metrics" developed by Wu and Nevatia [59] measure different scores, including:

- Mostly tracked trajectories, which refer to objects that are correctly tracked in at least 80% of the total frames.
- Mostly lost trajectories, which refer to objects that are only correctly tracked in 20% of the frames.
- Fragments, which are trajectories that cover more than one fragment.
- False trajectories, which refer to trajectories that do not match the object in the scene.
- ID switches, which refer to the number of times that correctly tracked objects have changed incorrectly.

The CLEAR MOT metrics, developed for the Classification of Events, Activities, and Relationships (CLEAR) workshop [62], summarize the basic measurement metrics of that time. This evaluation system was systematically constructed based on large-scale annotated data and a wide variety of participating systems, to evaluate the usefulness of an MOT algorithm with metrics. The CLEAR workshop demonstrated that MOT metrics can be applied to a wide range of object tracking tasks, allowing us to identify the strengths and weaknesses of the algorithms [60]. The performance tests conducted in the study covered a range of tasks, including 3D visual tracking, 2D face tracking, 2D person and vehicle tracking, 3D acoustic and multi-modal person tracking.

As the database used in our study was generated from a single camera at one angle, the focus of the metrics was on 2D tracking with a single camera. Additionally, this method is widely used when a tracker corresponds to a detector that decides if an object and its related prediction with an IoU. Similar to the detection stage of object detection, the mapping between predictions and trajectories for detection was performed. If the ground truth object o_i and the prediction p_j matched on frame $f-1$ with $IoU(o_i, p_j) \geq \theta$, then the match was continued with other predictions until all were processed. Predictions that did not match the decided BB were considered FN and FP, while the fragmentation, or the time when a ground truth object is lost and then resumed, was counted as an ID switch. Finally, the MOTA is calculated as follows:

$$\text{MOTA} = 1 - \frac{\text{FN} + \text{FP} + \text{IDSW}}{\text{GT}} \in (-\infty, 1] \tag{3.6}$$

We calculate MOTP to overcome the chance of ending up with a negative value in MOTA, which focuses mainly on tracking performance. The general practice is to report MOTP instead of MOTA. The MOTP is defined as the ratio of the sum of the bounding box IoU with the ground truth on frame f to the sum of correct matches on frame f. The formula for MOTP is given by equation (3.7) as:

$$\text{MOTP} = \frac{\sum_f^i d_{f,i}}{\sum_f c_t} \tag{3.7}$$

where c_t is the number of correct matches on frame f and d_f is the BB IoU with the predicted i according to the ground truth. It is important to note that MOTP score primarily reflects the performance of the detector used and not necessarily the tracking performance of the model.

We propose the use of ID metrics to address the limitation of MOTA scores, which are influenced by the number of incorrect decisions made by a tracker such as IDSW. However, this may not be practical for all scenarios, particularly in cases where a long tracking time for an object is desired. To address this, we adopt a predicted trajectory that maximizes the number of correctly classified frames by matching the ground truth and detections frame by frame. This converts the performance into a global measure. As previously mentioned, the CLEAR MOT metrics are more appropriate for this study and, like most reported work, we use CLEAR MOTA and DetectA due to the database used and the specific needs of the project that the algorithm serves. [57]

3.3 Object Labeling Methodology

In our automated bee hive application, we utilize a pipeline that comprises three main phases: detection, tracking, and labeling, which we adopt in a recursive manner to develop a reliable and efficient system. We have previously discussed the first two phases of the pipeline, and in this section, we will introduce the third phase, labeling.

3.3.1 The Honeybee Labeling Tool

We developed the Honeybee Labeling Framework as a specific software for the Automated Bee Hive project, aimed at reducing the workload of manual labeling. Our labeling tool focuses on steps 3 to 7 in handling tracking errors, which were previously explained in the pipeline's detection and tracking phases.

The process of our Honeybee Labeling Framework is as follows: Image Input -> Step 1: Object Detection -> Step 2: Object Tracking -> Step 3: Provide ID and Locations of Honeybees (per 1 of 3 frames) -> Step 4: Extract Differences -> Step 5: Convert Video to Frame and Print IDs -> Step 6: Show User Suspected Detection -> Step 7: Human Annotation -> Update Ground Truth Data -> Ground Truth (Output)

By obtaining the ID and locations of the honeybees from the detection and tracking algorithm output, we can easily measure the detections and tracking records. As the total number of honeybees is known and stable, the unique ID assigned to each honeybee serves as a good indicator of the detection and tracking accuracy. The tool's user can easily identify changes in the number of honeybees, which could indicate errors in steps 1 and 2, such as loss of objects, new detections, or lost/found objects. To handle these changes, we compare the locations comma-separated values file with an iterative approach and consider the max bee ID, which ranges from 0 to 19 when all objects are detected in the first complete detection frame. If the max bee ID changes in the next frame, it indicates that the human annotator needs to check the change, as the number would not change during the experiment if the detection and tracking algorithm were 100% accurate. Our Honeybee Labeling Tool, with the augmentation of a human annotator, corrected these errors in the tracking algorithm.

3.3.2 Object Label Changes

We faced one challenge during the experiments, which involved losing an object and then re-finding it in more than 15 consecutive frames (1 frame out of every 3 frames). This scenario is depicted in Figure 3.6. To handle this issue, we employed different methods. For case (a) in Figure 3.6, the detector identified and tracked a honeybee with a track id, referred to as bee id. However, the detected and tracked object could be lost, and if the lost state of the object exceeded 0.5 seconds (15/30 fps), which corresponds to 15 frames in the video, a new id was assigned when a new object was found. To address this, we pushed the lost object onto a stack and waited for re-detection, then assigned the previously lost object track to the re-detected object. However, if the threshold was exceeded or more than one object was lost in the same frame, causing the problem to become more complex, we flagged the state as "change detected". Thus, the annotator could check the label switch case and determine if the newly detected object was a new object or just a lost and re-detected object, by examining the frames 30 frames before and after the change detected flag assigned frame.

For case (b), the detector assigned a new bee id to a newly detected object. The most likely cause of this issue was the detection of a false positive (FP) that was very close to the current object, leading to the generation of a bounding box (BB) very near to the correctly tracked object. If the objectiveness score degraded and the FP detection caused the object to be lost, the tracking id held in the stack for re-assignment was reassigned to the new object instead of the first tracked bee id, resulting in the loss of track of the previous object. To address this, we directly flagged the frame and showed the frame set to the annotator, allowing them to eliminate the wrong detection and reassign the new id number to the old id number by referencing the frame number where the change occurred.

Case (c) differs from case (b) in that it involved the detection of a true positive (TP) that was very close to another object, particularly when honeybees form clusters. For the sake of simplicity, the remaining honeybees were not represented in the figure. When the detector assigned the center point of the BB in the middle of two or more honeybees, a single BB was generated for multiple honeybees. This resulted in the mis-detection of some scene objects, as the total number of objects was known on the current frame. In subsequent frames, the detections were separated as the honeybees were re-found, particularly after the clusters increased in inner distance. This interchange process impacted MOTA values by increasing the switch number. The only solution used here was to show the frame set to the annotator for correcting the label switches.

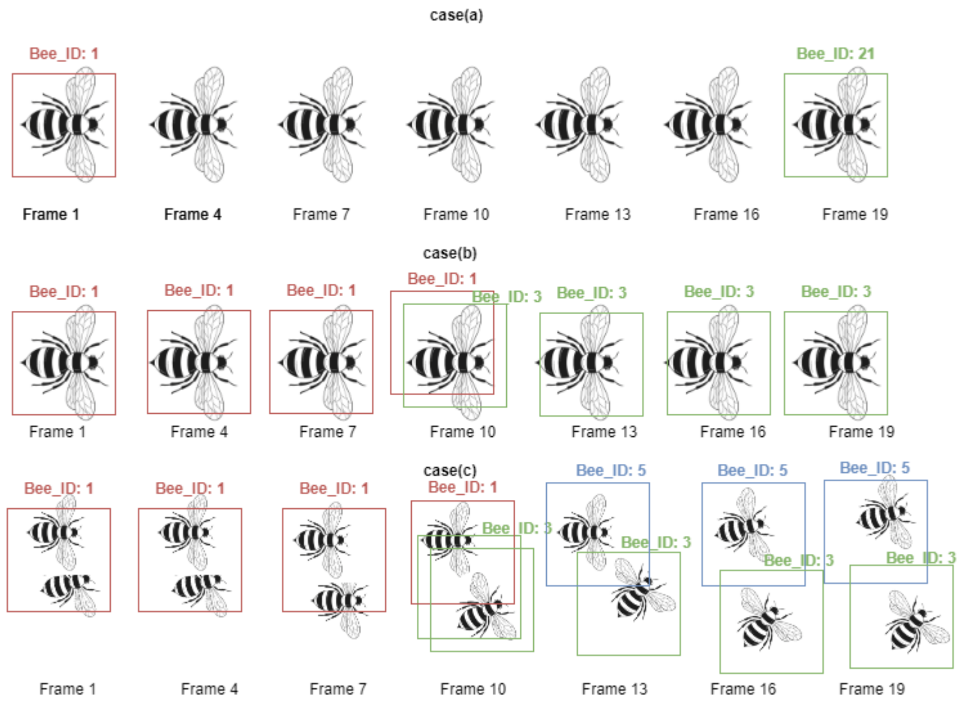


Figure 3.6: The Bee ID (label) change cases

3.3.3 The False Positives Trackings

We encountered occasional false positive tracking errors due to the presence of dots on the background of the setup, which are similar to honeybees, as seen in Figure 3.7. In these cases, the detector detects these dots as newly detected honeybees and assigns them a new bee ID. We easily detected this issue by analyzing a sudden increase in the number of maximum bee IDs on the frame. We then presented the frame to the annotator, who easily flagged this change as a FP and discarded it from the locations file. The background of the setup is clean and white, which makes it easy to subtract, so the FP tracking error was not frequently encountered in our used data set.

3.3.4 Re-assigning Method

We improved the tracking of honeybees by using the reassigning method to handle lost tracks. The performance of the detector plays a crucial role in tracking success and annotating workload, as it relies on the detection success for tracking. Thanks to the white static background, the detector performed well on honeybees, except for

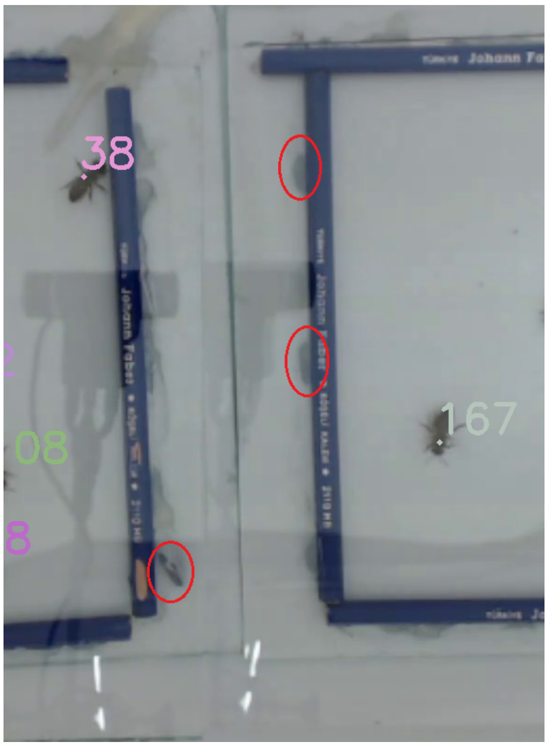


Figure 3.7: A sample of illumination and the dots on a random frame.

bee clusters, allowing lost objects to be re-found in subsequent frames.

However, even small changes in the detector caused dramatic increases in bee IDs assigned. As the basic approach in annotating honeybees is to give them IDs that correspond to their tracking IDs, any change in the tracking IDs due to a new object being found results in an increase in the total number of honeybee IDs. This is a problem in our study, as the total number of honeybees is known and does not change during the experiment, affecting the precision of the experiments that biologists will carry out. If the number of honeybees is not kept steady during the experiment, the effect of pesticides on honeybees cannot be clearly measured.

To reduce the number of false annotated frames, we added a threshold to the algorithm to delete a track after a series of lost track frames. Additionally, we used only one image from three consecutive frames to decrease the total number of processed images by a ratio of one to three.

The reassigning method also benefits from the limited movement areas provided by the arena setup for honeybee groups, as seen in Figure 3.8. The control and test groups

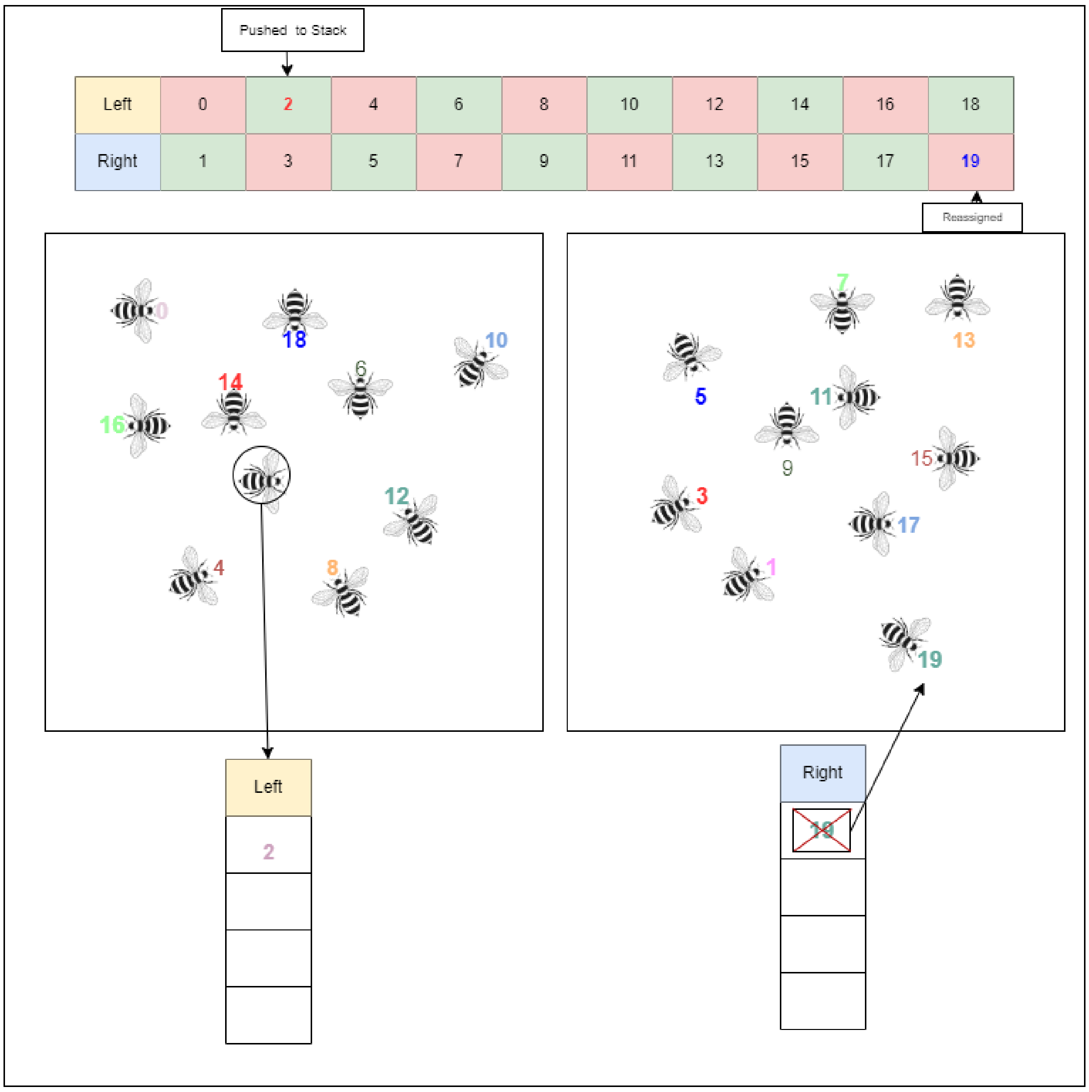


Figure 3.8: Reassigning of the honeybees

are separated into distinct areas, with odd-numbered tracks assigned to one side and even-numbered tracks assigned to the other. This decreases the risk of losing track of the honeybees, compared to labeling all 20 honeybees with a possibility of 10-10 for each group. By keeping two different stacks, lost and re-found objects can be assigned the same track, as the latest lost track is not deleted and kept for re-finding.

However, the method also has drawbacks. If the number of lost tracks for each group exceeds two at the same frame, or if the latest lost track cannot be assigned to any detection due to exceeding the threshold, the track may be assigned to the wrong honeybee or be completely lost due to the deletion of the corresponding track.

3.3.5 The Honeybee Labeler Tool GUI

We developed a user interface that streamlines the process of correcting tracking errors and annotating data. Our hybrid model reduces workload and saves time. Figure 3.9 provides an overview of the Honeybee Labeler Tool GUI we developed.

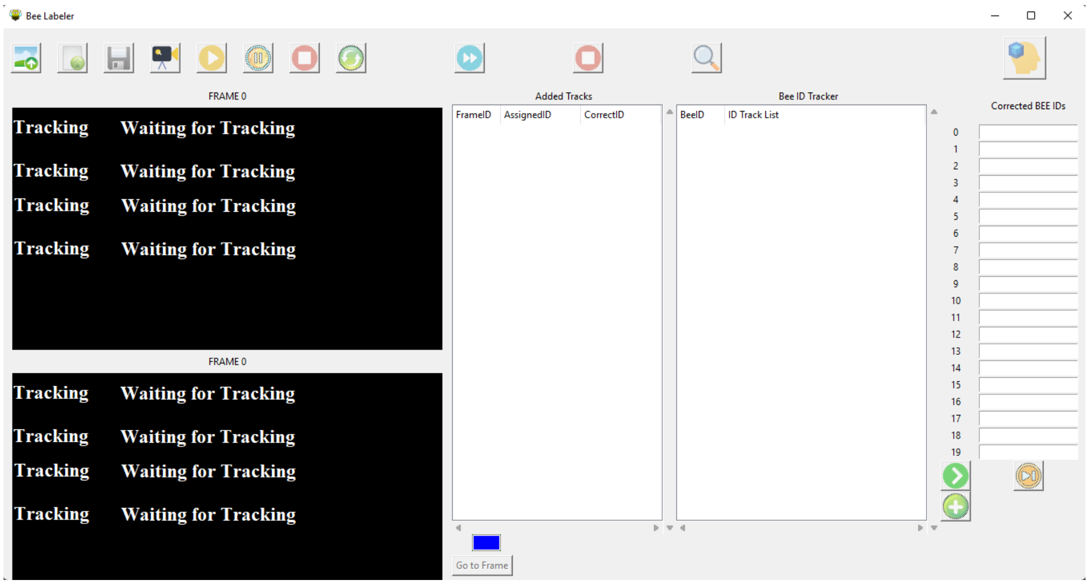


Figure 3.9: The Bee Labeler Tool GUI

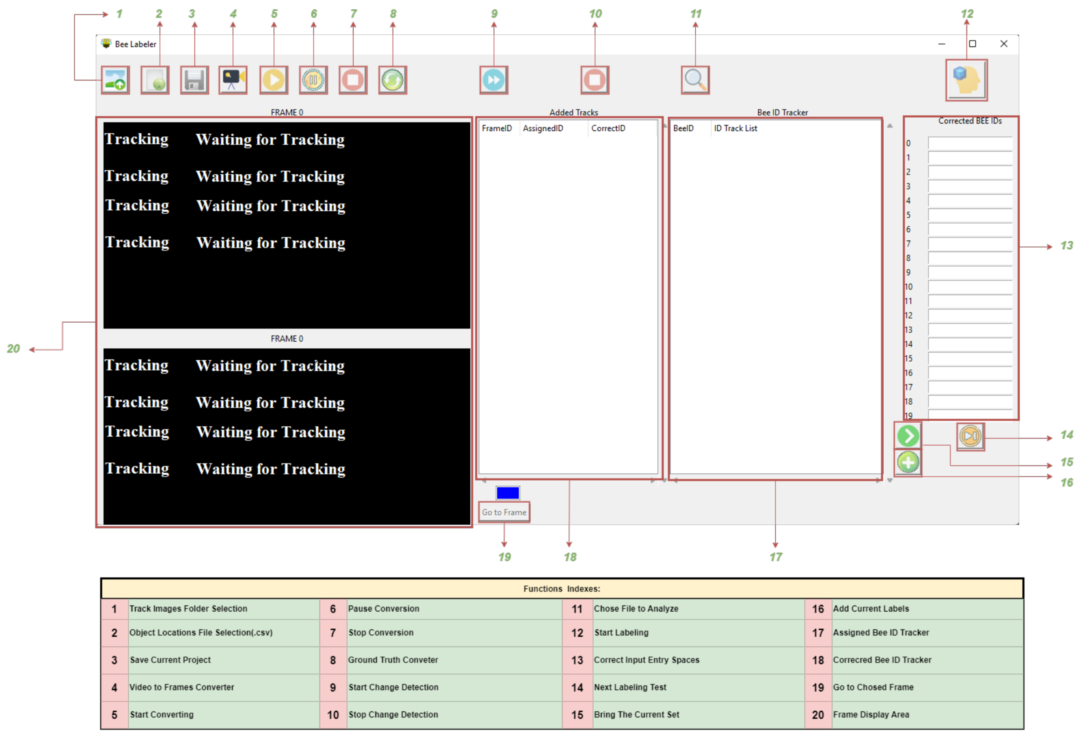
3.3.5.1 Requirements and Specifications

We developed the Honeybee labeling tool using Tkinter version 8.5 with python version 3.9. Tkinter, also known as TK interface, is a standard Python interface to the Tc/Tk GUI toolkit. Both Tk and Tkinter are readily available for Unix, macOS, and Windows.

By utilizing Tkinter, we were able to take advantage of its features, including widgets, geometry management, and event handling, to create a modern GUI for our project.

3.3.5.2 Algorithms and Functions

The Figure 3.10 demonstrates the functions and introduction of the labeling tool utilized in the Bee Hive 4.0 project. The index numbers displayed in the figure summarize the functions and their details are outlined in the subsequent sections.



Functions Indexes:							
1	Track Images Folder Selection	6	Pause Conversion	11	Chose File to Analyze	16	Add Current Labels
2	Object Locations File Selection(.csv)	7	Stop Conversion	12	Start Labeling	17	Assigned Bee ID Tracker
3	Save Current Project	8	Ground Truth Conveter	13	Correct Input Entry Spaces	18	Correcred Bee ID Tracker
4	Video to Frames Converter	9	Start Change Detection	14	Next Labeling Test	19	Go to Chosed Frame
5	Start Converting	10	Stop Change Detection	15	Bring The Current Set	20	Frame Display Area

Figure 3.10: The Bee Labeler Tool Functions

As we examine the figure, the following functions are present:

1. The Track Image Folder Selection Button enables us to select the folder location where the selected video frames are saved, displaying the screen locations from the locations file. This makes it possible for us to view sequential frames for labeling and separates work on the same video by selecting different folders. Note that the generated images are deleted for storage and memory efficiency.

2. The Object Locations File Selection Button imports the locations file (.csv) generated by the tracking algorithm. This file consists of the Frame ID, Bee ID, and Locations (x,y), which we use to compare the frames, detect changes, and decide which frames will be included in a set. It also prints the locations information to the selected set of frames.

3. The Save Current Project Button saves the project to avoid losing any completed labeling work. It also has a background check and will not allow us to quit the application without saving the current file. Additionally, it triggers

an auto-save function after every hundred frames labels to prevent the loss of progress.

4. The Video Frames Converter Button allows us to choose and convert a subject video from the dataset to Tracking Images and save them to the selected location.
5. The Start Converting Button resumes the conversion process if it was paused. This is a less frequently used option as the process is typically handled in sets.
6. The Pause Conversion Button pauses the conversion process.
7. The Stop Conversion Button completely stops the conversion process and requires a restart of all processes if necessary. The current process will be overwritten by a new process.
8. The Ground Truth Converter (Merger) Button matches the locations file and the annotation final file, merging them into the ground truth.
9. The Start Change Detection Button checks for the maximum bee id increase and compares the locations of honeybees to identify changes to be analyzed by the annotator.
10. The Stop Change Detection Button stops the current process and saves the change.csv file with the current detections.
11. The Choose Analyze File Button helps us choose the file to analyze and bring the change sets for annotation.
12. The Start Labeling Button initiates the labeling process by checking and creating all necessary files and folders. Once everything is in place, it enables the labeling functions to start.
13. The Correct Input Entry Space is designed for annotators' inputs. When annotators provide inputs and click on the 16 numbered button, their inputs are saved, and the honeybee Id indexes on the left-hand side are updated.
14. The Next Labeling Set Button moves on to the next set of track images for labeling if all ids have been correctly assigned in the current set.

15. The Bring Labeling Set Button calls the first set of track images if they exist.

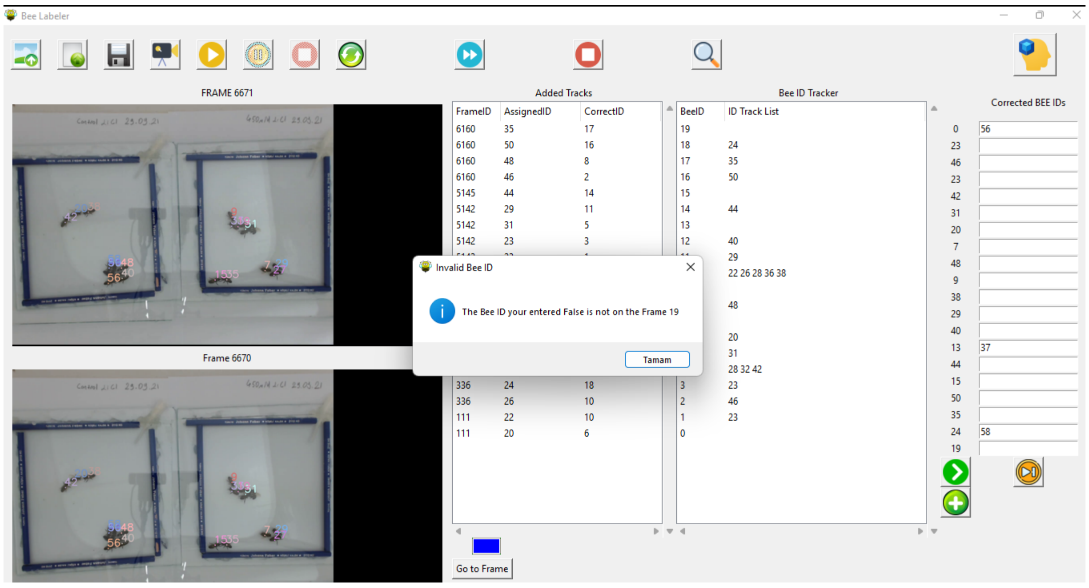
16. The Add Current Labels Button adds the input labels to the assigned Bee ID tracker space.

17. The Assigned Bee ID Tracker Space assists us in identifying the previously assigned id before the correct labels were given by the annotator.

18. The Corrected Bee ID tracker Space allows us to follow the ids on the frames and their corresponding corrections on the indexes side.

19. The Go to Frame Button helps us navigate to specific frames to check the object positions if there is a need for clarification.

20. The Frame Display Area provides users the ability to check the differences in the current track image set, allowing them to easily understand the reason for changes and identify the honeybees.



Added Tracks

FrameID	AssignedID	CorrectID
6160	35	17
6160	50	16
6160	48	8
6160	46	2
5145	44	14
5142	29	11
5142	31	5
5142	23	3
336	24	18
336	26	10
111	22	10
111	20	6

Bee ID Tracker

BeeID	ID Track List
19	
18	24
17	35
16	50
15	
14	44
13	
12	40
	29
	22 26 28 36 38
	48
	20
	31
	28 32 42
3	23
2	46
1	23
0	

Corrected BEE IDs

0	56
23	
46	
23	
42	
31	
20	
7	
48	
9	
38	
29	
40	
13	37
44	
15	
50	
35	
24	58
19	

Invalid Bee ID

The Bee ID your entered False is not on the Frame 19

Tamam

Figure 3.11: The Bee Labeler operation

A photo of the GUI during operation with a false label entry is shown in Figure 3.11. Our aim in designing the functions mentioned above is to reduce the time required for labeling while improving its quality.

3.3.5.3 Order of Flow

In this thesis, we describe the procedure followed by an annotator when using the tool for labeling honeybees in a video. To begin, we first choose a folder to generate the tracked frames. Next, we import the location files by selecting the "locations.csv" file in the tool. Then, we upload the video file to apply the tracking labels to the frames.

Once the uploading is complete, we detect the change set and start the conversion process by clicking the "start converting" button. This initiates the printing of the labels of the honeybees to the relevant images, starting with the first set of images. When we switch to analyzing mode by clicking "start analyzing", all inputs provided in the right-hand side input area are saved.

When we have completed the labeling or need to close the current work for later, we save the project. This stores the session information, including the labeling, tracking, and change set values, in a session file, enabling us to restore it at a later time.

CHAPTER 4

EXPERIMENTAL EVALUATION

We present the usage of our labeling tool and the results in this chapter. In Section 4.1, we describe the experimental methodology we employed. Section 4.2 presents our findings on object detection, tracking and labeling. In Section 4.3, we evaluate the results, with a particular focus on our labeling tool and its impact on detection and tracking performance.

4.1 Experiments

In our thesis, we conduct experiments in a linked manner. First, we select a target video as the starting point. Next, we feed the selected video to the object-detecting and tracking algorithm pipeline proposed for this study. This produces an output file that includes the algorithm's predictions for the video's coordinates and labels. After that, we label the video manually to determine the time it takes. Then, we provide the algorithm's output to the semi-automated labeling tool to create a ground truth file, measuring the time as well. Finally, we evaluate the performance and gain of both the algorithm and labeling tool.

In the following subsection, we share the details of the experiment we carried out.

4.1.1 Experimental Procedure

In our study, we conducted an experiment on a dataset containing hundreds of videos of experiments observing the effects of pesticides on honeybees. The behavior of honeybees differed based on the number of chemical substances used. To ensure proper observation of honeybee behavior, we chose a video that contained more clusters of honeybees, which posed a challenge for the object detection and tracking algorithm. Our chosen video was 15 hours long and had 1.6 million frames.

To reduce the cost and human workload of labeling the entire video, we used a specific duration of 20.004 frames. The chosen duration was the most affordable option in terms of time and money.

We fed the chosen video to the detection and tracking algorithms and completed the test. However, creating a ground truth file for 1.6 million frames was deemed too costly. To reduce the cost, we filtered the results to generate one frame for every three consecutive frames, reducing the total number of frames to 546.000. We focused on the part of the video that contained 20.004 frames to avoid losing perspective. The output of the experiment was a list of honeybee IDs and their positions.

We initiated the labeling process with semi-automated labeling and compared the similarity of the ground truth values generated by hand and by the tool. After completing the labeling, three expert annotators evaluated the labeled video to measure the time-saving potential of our tool.

Evaluating the results of the experiment, we used the detection accuracy metric to assess the detection and MOTA and precision values to measure the tracker performance. However, MOTP was not used as it mainly focuses on the localization of the detector and not the performance of the tracker. Our study aimed to decrease the workload of the ground truth file creation process and focused on reducing the time spent during the labeling process. The gain of our proposed method was calculated by checking the sustainability of a label, the recovery of mismatched or lost labels, and the time spent during the labeling process.

4.2 Results

The object detector and tracker work together to accurately detect and track objects in a video sequence. The object detector is responsible for detecting objects in individual frames, while the tracker is responsible for linking detections across frames to maintain object identities. In our system, we used YOLOv3 as the object detector and Hungarian and Kalman Filter as the tracker.

The output was evaluated by comparing the ground truth values of the subject video. The parameters are calculated in order to measure the performance of the detector and tracker. Because the ground truth-generating performance depends on the performance of the detector and tracker.

The TP corresponds to the times when the algorithm detects the right object and kept the given IDs (labels) by touching the honeybees on a specific frame.

The FP corresponds to the times when the algorithm detects and tracks the wrong object by pointing to a coordinate that has no actual bee on it.

The FN corresponds to the times when the algorithm did not detect and track the honeybee actually on the frame.

The IDSW value corresponds to the ID switching that occurs when the honeybees get too close to each other. The correct IDs switch in between and in the future, it keeps changing with other IDs. The value counts of every id switched frame by switch times.

The Deviation value corresponds to times when the algorithm starts a new track on detection that exists like a new object. These values were not counted on the frame base since the labeling handled used only a time approach for every id label.

The parameters are used to calculate the MOTA, precision, recall, and DetectA scores. Since the MOTP is not telling so much about the tracking performance the MOTA is preferred for the tracking performance and detection performance is measured with DetectA and precision scores.

The MOTA is calculated as follows:

$$\mathrm{MOTA} = 1 - \frac{\mathrm{IDFN} + \mathrm{IDFP} + \mathrm{IDSW}}{\mathrm{GT}} \tag{4.1}$$

The ID-Precision is calculated as follows:

$$\mathrm{ID-Precision} = \frac{\mathrm{IDTP}}{\mathrm{IDTP}+\mathrm{IDFP}} \tag{4.2}$$

The ID-Recall is calculated as follows:

$$\mathrm{ID-Recall} = \frac{\mathrm{IDTP}}{\mathrm{IDTP}+\mathrm{IDFN}} \tag{4.3}$$

The IDF1 is calculated as follows:

$$\mathrm{IDF1} = \frac{\mathrm{IDTP}}{\mathrm{IDTP}+0,5\times(\mathrm{IDFN}+\mathrm{IDFP})} \tag{4.4}$$

The DetectA is calculated as follows:

$$\mathrm{DetectA} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}+\mathrm{FP}} \tag{4.5}$$

Table 4.1: Object detection results

True Positives (TP)	False Positives (FP)	False Negatives (FN)
387.184	12.816	5.736

Evaluating the performance of the object detector according to the results in Table 4.1, we achieved a Precision of %97, Recall of %98, and F1 score of %97 in Table 4.2. Precision is a measure of how many of the detections are true objects, while Recall is a measure of how many objects are detected.

Table 4.2: Object detection performance

Precision	Recall	F1 Score
0,967	0,98	0,97

The F1 score is a weighted average of Precision and Recall and provides a comprehensive evaluation of the performance of the object detector. These results demonstrate that our object detector is able to accurately detect objects in the video sequence, with a low number of False Positives and False Negatives.

Table 4.3: Object tracking results

IDTP	IDFP	IDFN	IDSW	Deviation
384.360	15.720	31.440	2.850	77

When evaluating the overall performance of the ID tracking system according to Table 4.3, we achieved IDTP 384.360, IDFP 15.720, IDFN 31.440, IDSW 2.850, Deviation 77, and MOTA %89, ID-Precision %96, ID-Recall %89, IDF1 %94, and DetectA %96 which can be seen in Table 4.4. These results indicate the algorithms struggle with continuously tracking.

Table 4.4: Object tracking performance

MOTA	ID-Precision	ID-Recall	IDF1	DetectA
0,89	0,96	0,89	0,94	0,96

We observe that despite the high scores achieved by our implementation, the detection and tracking algorithm struggles to continuously track a detected object throughout the video. This hinders its ability to be used directly to create a ground truth from the prediction file. To ensure the production of accurate ground truth outputs, we need to perform the labeling process and retrain the algorithm or measure its performance.

Table 4.5: The ID error details

Source	False Positives	False Negatives	IDSW	Deviation
Object Detection	12.816	5.736	-	-
Object Tracking	15.720	31.440	2.850	77

When the details of the ID errors are considered according to Table 4.5 in detail, it indicates the false positives in object detections are triggering the false id assignments to the nonexisting objects which increases the false positives in object tracking. The difference between the false positives of detection and tracking algorithms is related to the tracking performance of the framework. Because we are using tracking by detection, the increase in the false positives of the object detection linearly affects the false positives of the object tracking. The false positives of the object detection are considerably lower than the object tracking false positives. The reason for this is the tracking algorithm suffers from the IDSWs and deviations, especially in the honeybee clusters. When a given ID changes with a false positive object that increase the false positive numbers and causes neglecting of an object in a cluster which increase the false negative counts.

Table 4.6: Recovered bee label result snippet

FrameID	AssignedID	CorrectID	Recovered
110	20	6	26.583
111	22	10	219
334	24	18	49
335	26	10	77
389	28	4	19
419	32	10	5.710
425	36	18	6.229
914	38	4	52.068
931	40	8	6.069
932	42	10	51.550

We present our labeling tool, designed to increase accuracy while decreasing time and cost in the labeling process. We introduce the relevant parameters of the labeling action in Table 4.6 which features a frame ID column for the tracked frames where the first loss of the bee occurred, an assigned ID column to indicate whether the object is new or an IDSW, and a corrected ID column to show the label correction made by the annotator. Additionally, the recovered column displays the frame number of the labeled series recovered from the prediction file during the ground truth creation process. Although only the first 10.000 frames (out of 20.004) are considered in this study, it is evident from the table that the values reached much higher numbers.

To visually present the results of the recovered bee IDs, we created a plot as seen in Figure 4.1. The blue bar depicts the successfully tracked honeybees until the first loss of the track or the IDSW, while the red bars indicate the lost and recovered bees through the labeling process. The 10.000 frame number represents the duration of interest in the selected video, while the rest of the video has been omitted for the purpose of simplification

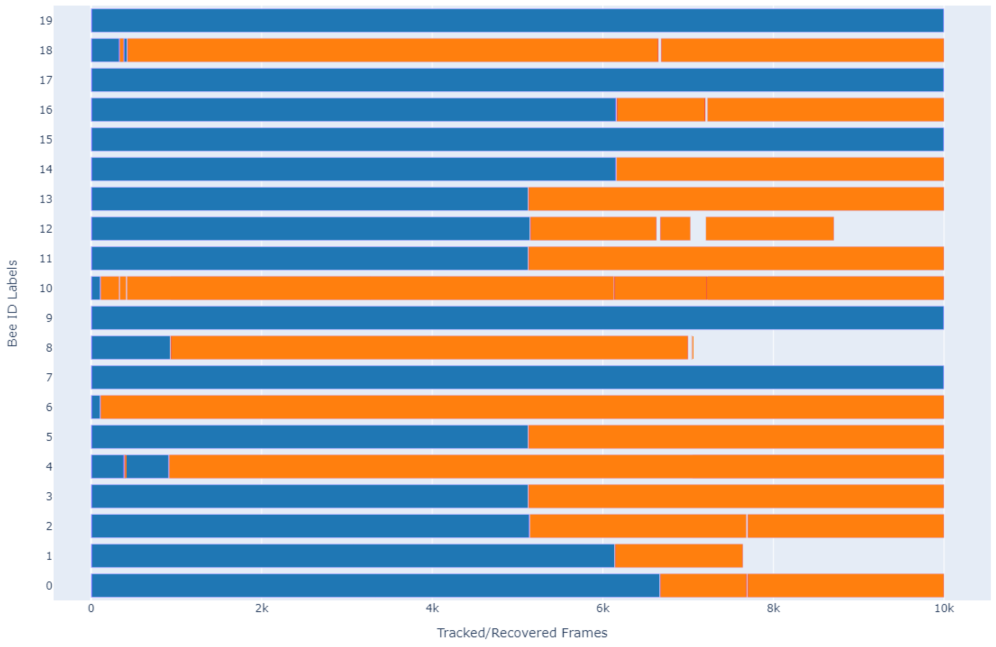


Figure 4.1: The tracked and recovered frames performance

We have successfully reduced the manual labeling effort by recovering IDs, which enabled us to decrease the time consumed. Out of the 20 objects on the frame, we were able to recover approximately 70% of the whole frame. Furthermore, all the 17 objects were recovered across all the frames of the video.

$$\text{Gain} = \frac{RO + TO}{GT} \tag{4.6}$$

We decreased the manual labeling effort by recovering objects. We evaluate this reduction by calculating the True Positive (TP) rate of each recovered object divided by the total number of ground truth objects. Applying this calculation to the recovered version demonstrates the gain in efficiency achieved for the entire video. The recovered gain of the algorithm is calculated in the

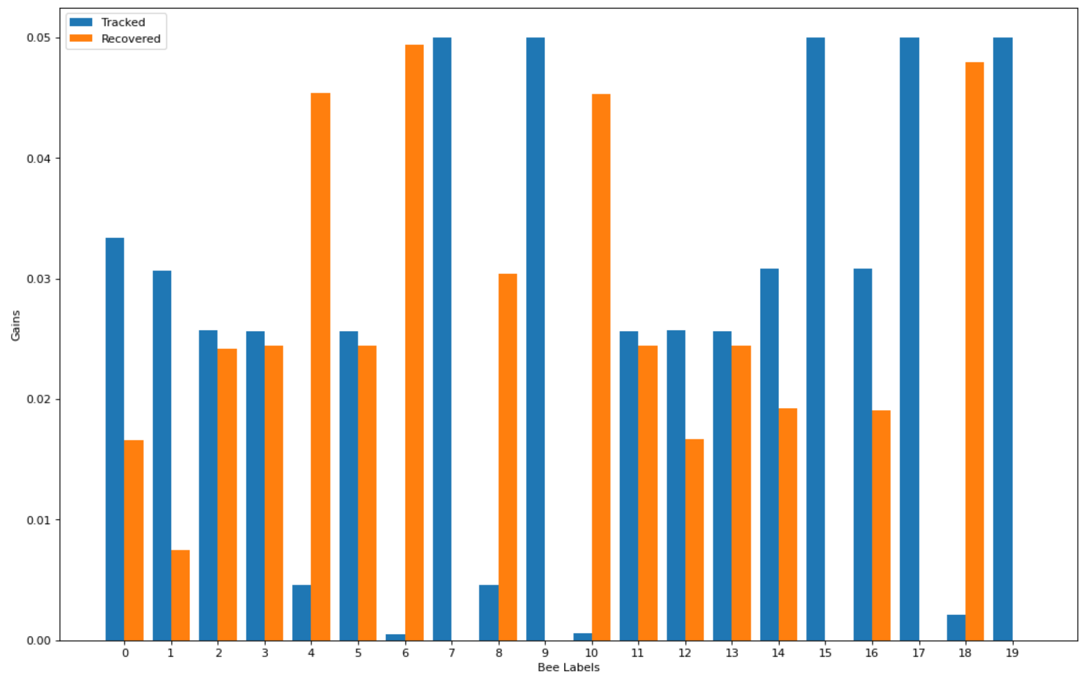


Figure 4.2: The tracked and recovered gains

The graph demonstrating the gain is presented in Figure 4.2. By comparing the tracked precision with and without the use of the labeling tool, we observe a significant increase in performance. Specifically, the tracked precision was 54% without the use of the labeling tool, but with its implementation, we achieved a 42% increase, resulting in a total of 96%.

We calculate the workload of labeling as shown in Figure 4.3. Our analysis reveals

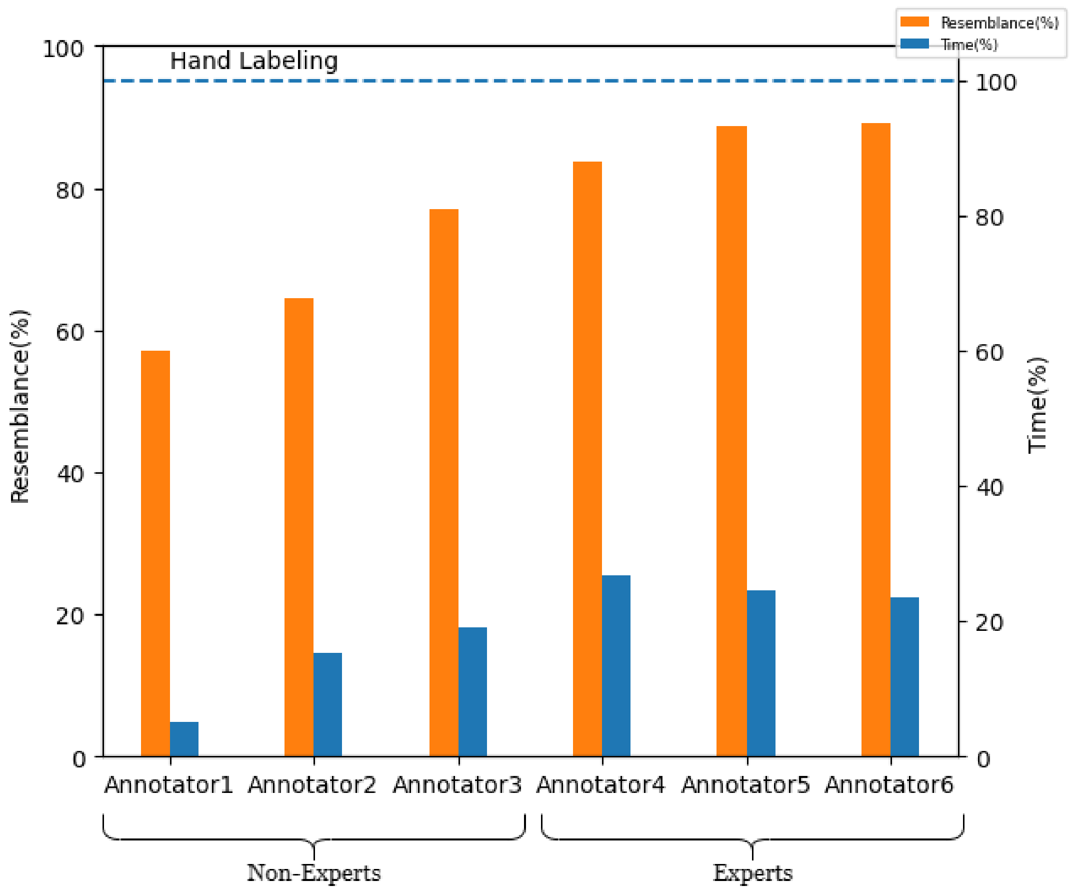


Figure 4.3: The workload and resemblance trade-off

that the performance of the labeling tool heavily relies on the detection and tracking algorithm performance. As our approach to the problem involves tracking through detection, any inefficiencies in the detection performance will inevitably impact the tracking performance, leading to a higher need for correction and additional labeling in the prediction output.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In our study, we introduced a semi-automated object labeling framework for the "Beehive 4.0: Artificial Intelligence-based Approach to Honeybee (Apis Mellifera) Observation" Project. Our framework enables human annotators to correct the errors generated by existing detector and tracker algorithms, instead of manually labeling the entire video, thereby reducing the labor-intensive process. We designed the framework to alleviate the time-consuming task of manual labeling through the integration of human-computer interaction output.

We empowered annotators to handle various types of errors such as IDSW and mismatch, produced by detectors and trackers, by incorporating human revision. Our results showed an improvement in the quality of the generated ground truth values compared to those obtained through manual labeling. However, a trade-off was evident, where more time spent using the labeling framework led to higher resemblance results and vice versa. Furthermore, we found that the results of expert annotators were consistent, regardless of the time spent, while non-expert annotators' results were more time-dependent.

We evaluated the overall performance of the annotators, tracked, and recovered frame numbers under the assumption of equal annotators' concentration and motivation. Our findings indicate that the workload can be reduced by an average of 70% in times with an average 80% resemblance rate. The high resemblance scores of 90% achieved by expert annotators demonstrate the effectiveness of our developed ILS in reducing the burden of labeling while maintaining the quality of the labels.

In this study, we observed that the workload for automated bee hive systems depends on various factors, particularly the performance of trackers and detectors. By improving the performance of these tools and implementing problem-specific solutions to optimize the algorithm, we can decrease the need for manual labeling. Additionally, using an ILS can further reduce the workload by automating the labeling process. Our findings suggest that the semi-automated approach, in which labels are not defined for every frame by a human annotator, has the potential to save both time and human effort. Overall, our research indicates that implementing a semi-automated human-computer interaction processed labeling approach can greatly benefit automated bee hive systems.

In our study, we acknowledged that the workload for automated bee hive systems depends on various factors, particularly the performance of trackers and detectors. By improving the performance of these tools and implementing problem-specific solutions to optimize the algorithms, we can reduce the need for manual labeling. Additionally, the use of an ILS can further minimize the workload by automating the labeling process. Our research suggests that the semi-automated approach, where labels are not assigned for every frame by human annotators, has the potential to save time and human effort. Overall, our study highlights the benefits of implementing a semi-automated human-computer interaction-based labeling approach for automated bee hive systems.

In this thesis, we propose that the labeling pipeline can be improved by exploring three categories of future work: object detection, object tracking, and object labeling. To improve the object detection part, we suggest investigating multiple object detectors with varying settings, purposes, and capabilities to gain deeper insight into the detection performance of the labeling tool. Additionally, we propose increasing the training samples from the actual environment to account for the limitations of the current system, which uses only 40 images of honeybees with a white background and known number of objects. We recommend conducting more tests on the NMS value and comparing the performance of the NMS and clustering algorithms to obtain more analyzable results.

To improve the tracking part, we suggest studying more complex computer vision techniques to fix the trajectories of honeybees, thus reducing the matching error. A reassigning strategy supported by scores of loss rate cases could further improve the accuracy of the tracking algorithm. After labeling the predictions, the algorithm can be re-run to re-calculate and compare the tracking accuracy.

For the labeling part, we recommend using multiple object detectors to generate different predictions, which could be combined using a voting scheme to improve the precision of the labels. The positional precision of the labels has not been considered in this study. So, we suggest implementing positional correction of the labels by fixing the center point to decrease ID switching between objects and increase the overall performance.

REFERENCES

[1] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial intelligence review*, vol. 53, no. 8, pp. 5455–5516, 2020.

[2] R. Padilla, S. Netto, and E. da Silva, "A survey on performance metrics for object-detection algorithms," 07 2020.

[3] G. Welch, G. Bishop, *et al.*, "An introduction to the kalman filter," 1995.

[4] A. Torralba, B. C. Russell, and J. Yuen, "Labelme: Online image annotation and applications," *Proceedings of the IEEE*, vol. 98, no. 8, pp. 1467–1484, 2010.

[5] D. K. Iakovidis, T. Goudas, C. Smailis, and I. Maglogiannis, "Ratsnake: a versatile image annotation tool with application to computer-aided diagnosis," *The Scientific World Journal*, vol. 2014, 2014.

[6] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently scaling up crowdsourced video annotation," *International journal of computer vision*, vol. 101, no. 1, pp. 184–204, 2013.

[7] D. Mihalcik and D. Doermann, "The design and implementation of viper," *University of Maryland*, vol. 21, no. 22, p. 3, 2003.

[8] I. Kavasidis, S. Palazzo, R. Di Salvo, D. Giordano, and C. Spampinato, "A semi-automatic tool for detection and tracking ground truth generation in videos," in *Proceedings of the 1st international workshop on visual interfaces for ground truth collection in computer vision applications*, pp. 1–5, 2012.

[9] A. Ambardekar, M. Nicolescu, and S. Dascalu, "Ground truth verification tool (gtvt) for video surveillance systems," in *2009 Second International Conferences on Advances in Computer-Human Interactions*, pp. 354–359, IEEE, 2009.

[10] F. Comaschi, S. Stuijk, T. Basten, and H. Corporaal, “A tool for fast ground truth generation for object detection and tracking from video,” in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 368–372, IEEE, 2014.

[11] M. Pizenberg, A. Carlier, E. Faure, and V. Charvillat, “Web-based configurable image annotations,” in *Proceedings of the 26th ACM international conference on Multimedia*, pp. 1368–1371, 2018.

[12] N. Fiedler, M. Bestmann, and N. Hendrich, “Imagetagger: An open source online platform for collaborative image labeling,” in *Robot World Cup*, pp. 162–169, Springer, 2018.

[13] R. Hallman, J. a. Setubal, and A. da Conceicão, “(old) finding minimum congestion spanning trees,” *J. Exp. Algorithmics*, vol. 5, p. 11, 2017.

[14] G. M. Almond, R.E.A. and T. E. Petersen, *Bending the curve of biodiversity loss.* WWF, Gland, Switzerland: Bending the curve of biodiversity loss., 1st ed., 2020.

[15] G. Ceballos, P. R. Ehrlich, and R. Dirzo, “Biological annihilation via the ongoing sixth mass extinction signaled by vertebrate population losses and declines,” *Proceedings of the National Academy of Sciences*, vol. 114, no. 30, pp. E6089–E6096, 2017.

[16] R. M. Johnson, L. Dahlgren, B. D. Siegfried, and M. D. Ellis, “Acaricide, fungicide and drug interactions in honey bees (apis mellifera),” *PloS one*, vol. 8, no. 1, p. e54092, 2013.

[17] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[18] Kuhn, “The hungarian method for the assignment problem,” *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[19] Y. LeCun, K. Kavukcuoglu, and C. Farabet, “Convolutional networks and applications in vision,” in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 253–256, 2010.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[21] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Light-head r-cnn: In defense of two-stage object detector," 2017.

[22] Z. Zhang, J. Warrell, and P. H. Torr, "Proposal generation for object detection using cascaded ranking svms," in *CVPR 2011*, pp. 1497–1504, IEEE, 2011.

[23] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[26] R. D. Singh, A. Mittal, and R. K. Bhatia, "3d convolutional neural network for object recognition: a review," *Multimedia Tools and Applications*, vol. 78, no. 12, pp. 15951–15995, 2019.

[27] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European conference on computer vision*, pp. 345–360, Springer, 2014.

[28] H. R. Roth, L. Lu, A. Seff, K. M. Cherry, J. Hoffman, S. Wang, J. Liu, E. Turkbey, and R. M. Summers, "A new 2.5 d representation for lymph node detection using random sets of deep convolutional neural network observations," in *International conference on medical image computing and computer-assisted intervention*, pp. 520–527, Springer, 2014.

[29] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015.

[30] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 80–91, 2012.

[31] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, pp. 945–953, 2015.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learningfor image recognition," *ComputerScience*, 2015.

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[35] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[36] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018. cite arxiv:1804.02767Comment: Tech Report.

[37] S. Fletcher, M. Z. Islam, *et al.*, "Comparing sets of patterns with the jaccard index," *Australasian Journal of Information Systems*, vol. 22, 2018.

[38] B. Qi, M. Ghazal, and A. Amer, "Robust global motion estimation oriented to video object segmentation," *IEEE Transactions on Image Processing*, vol. 17, no. 6, pp. 958–967, 2008.

[39] S. K. A. Prakash and C. S. Tucker, "Bounded kalman filter method for motion-robust, non-contact heart rate estimation," *Biomedical optics express*, vol. 9, no. 2, pp. 873–897, 2018.

[40] K. Saho, “Kalman filter for moving object tracking: Performance analysis and filter design,” *Kalman Filters-Theory for Advanced Applications*, pp. 233–252, 2017.

[41] C. J. Rapson, B.-C. Seet, M. A. Naeem, J. E. Lee, M. Al-Sarayreh, and R. Klette, “Reducing the pain: A novel tool for efficient ground-truth labelling in images,” in *2018 international conference on image and vision computing New Zealand (IVCNZ)*, pp. 1–9, IEEE, 2018.

[42] C. Sager, C. Janiesch, and P. Zschech, “A survey of image labelling for computer vision applications,” *Journal of Business Analytics*, vol. 4, no. 2, pp. 91–110, 2021.

[43] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, “Robustness and reliability evaluations of image annotation,” *The Imaging Science Journal*, vol. 64, no. 2, pp. 94–99, 2016.

[44] J. Yuen, B. Russell, C. Liu, and A. Torralba, “Labelme video: Building a video database with human annotations,” in *2009 IEEE 12th International Conference on Computer Vision*, pp. 1451–1458, IEEE, 2009.

[45] A. Dutta and A. Zisserman, “The via annotation software for images, audio and video,” in *Proceedings of the 27th ACM international conference on multimedia*, pp. 2276–2279, 2019.

[46] R. G. J. S. Shaoqing Ren, Kaiming He, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *arXiv preprint arXiv:1506.01497*, 2015.

[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[48] 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," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[49] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve.," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[50] R. Padilla, S. L. Netto, and E. A. Da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 international conference on systems, signals and image processing (IWSSIP)*, pp. 237–242, IEEE, 2020.

[51] B. Yang and R. Nevatia, "Online learned discriminative part-based appearance models for multi-human tracking," in *Computer Vision – ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), (Berlin, Heidelberg), pp. 484–498, Springer Berlin Heidelberg, 2012.

[52] Z. Sun, J. Chen, L. Chao, W. Ruan, and M. Mukherjee, "A survey of multiple pedestrian tracking based on tracking-by-detection framework," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 5, pp. 1819–1833, 2020.

[53] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[54] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

[55] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[56] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," *Advances in neural information processing systems*, vol. 29, 2016.

[57] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, “Deep learning in video multi-object tracking: A survey,” *Neurocomputing*, vol. 381, pp. 61–88, 2020.

[58] M. Isard and A. Blake, “A mixed-state condensation tracker with automatic model-switching,” in *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pp. 107–112, IEEE, 1998.

[59] B. Wu and R. Nevatia, “Tracking of multiple, partially occluded humans based on static body part detection,” in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR’06)*, vol. 1, pp. 951–958, IEEE, 2006.

[60] K. Bernardin and R. Stiefelhagen, “Evaluating multiple object tracking performance: the clear mot metrics,” *EURASIP Journal on Image and Video Processing*, vol. 2008, pp. 1–10, 2008.

[61] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, “Performance measures and a data set for multi-target, multi-camera tracking,” in *European conference on computer vision*, pp. 17–35, Springer, 2016.

[62] R. Stiefelhagen and J. Garofolo, *Multimodal Technologies for Perception of Humans: First International Evaluation Workshop on Classification of Events, Activities and Relationships, CLEAR 2006, Southampton, UK, April 6-7, 2006, Revised Selected Papers*, vol. 4122. Springer, 2007.