

國立臺灣大學生物資源暨農學院生物機電工程學系

碩士論文

Department of Biomechatronics Engineering

College of Bioresources and Agriculture

National Taiwan University

Master Thesis

AIoT 技術在設施栽培中精密監控小型害蟲與優化蜜蜂
授粉之應用

Applications of AIoT Technology for Precision Monitoring
of Small Pests and Optimization of Bee Pollination in
Protected Cultivation

張善程

Shan-Cheng Chang

指導教授：周呈霽 博士

Advisor: Cheng-Ying Chou, Ph.D.

中華民國 112 年 6 月

June, 2023





Acknowledgements

在兩年的碩士生涯中，其中遇到許多困難，但由於師長、家人以及同學的無私幫助及陪伴，最終也得以完成此本碩論。首先，衷心感謝我的指導教授周呈霖教授。感謝您在整個研究過程中的耐心指導和悉心指教。您的寶貴意見和建議對於我的研究工作至關重要，讓我受益匪淺。也感謝口試委員江昭皚教授、楊恩誠教授以及王人正老師的建議與鼓勵，讓我得以更加完善我的論文內容。

另外，我要特別感謝我的家人。感謝父母對我學業的支持和鼓勵，是您們給予我堅強的後盾，讓我有信心走過每一個學術挑戰。同時，我也要感謝我的實驗室夥伴們。感謝你們在我研究上的相互扶持和共同成長。有你們的陪伴，讓我在做實驗時不會太孤單，每一次的討論和交流都讓我重新獲得向前邁進的動力。

最後，我還想感謝所有實驗場地內的昆蟲。在害蟲實驗中，如果沒有粉蝨及薊馬的犧牲，我的黏蟲紙影像就沒有用處；而在蜜蜂實驗中，如果蜜蜂們無法忍受我的打擾，繼續專心採集花粉粒，我就無法收集到花粉粒影像，真心感謝牠們的配合。希望這份誌謝能夠表達出我內心深處的感激之情。我將永誌不忘，將來也會以更積極的態度回饋社會，繼續努力，報答所有支持和幫助過我的人。



摘要

在臺灣，為提高作物單位面積產量，通常採用易於管理的溫室種植方法。然而即便作物採用溫室種植，仍無法完全避免蟲害爆發。另一方面，蜜蜂不太適應於密閉空間內飛行，進而導致作物授粉效率不太穩定。因此，在蟲害防治方面，基於過去研究，本研究利用物聯網系統搭配 Faster R-CNN、YOLO 系列以及 Transformer 系列物件偵測模型進行大數據分析，我們提出更加輕量化且客製化的方法，以提供溫室害蟲防治之有效參考指標；另外授粉效率評估方面，過去研究多利用蜜蜂進出巢數量評估作物授粉效率，此法無法得知蜜蜂攜帶花粉粒大小，進而無法精確掌握與授粉效率節節相關的花粉量，因此本研究提出利用 Mask R-CNN、YOLACT 以及 Transformer 系列等實例分割模型的方法，針對花粉粒面積進行辨識以及計算，並進一步與實際花粉粒重量進行相關性分析，以優化蜜蜂授粉。基於以上研究成果，將對溫室中種植作物之蟲害防治以及蜜蜂授粉優化提供一個較智能的解決方案，進而提昇作物產量。

關鍵字：溫室種植、物聯網、物件偵測、實例分割



Abstract

In Taiwan, to increase the yield per unit area of cultivation, greenhouse cultivation methods that are easy to manage are usually adopted. However, even if crops are grown in greenhouses, pest outbreaks cannot be completely avoided, and bees are not well adapted to flying in confined space, resulting in unstable pollination efficiency of crops. Therefore, in terms of pest control, based on past research, this research uses an IoT system with Faster R-CNN, YOLO series, and Transformer series object detection models to conduct big data analysis. We propose a more lightweight and customized method to provide greenhouse pest control with effective reference indicators. In addition, in terms of pollination efficiency assessment, past studies mostly used the number of honey bees entering and leaving the hive to evaluate crop pollination efficiency. Such methods cannot know the size of pollen grains carried by honey bees and thus cannot accurately grasp the amount of pollen-related to pollination efficiency. The study proposes a method of instance segmentation models such as Mask R-CNN, YOLACT, and Transformer series to detect and

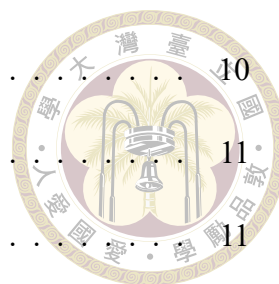
calculate the pollen grain area and conduct correlation analysis with crop pollination to provide a more reliable pollination efficiency evaluation index. Based on the above research results, a more intelligent solution will be provided for pest control and pollination efficiency assessment of crops grown in greenhouses, thereby maximizing crop yields.

Keywords: Greenhouse cultivation, IoT, Object detection, Instance segmentation



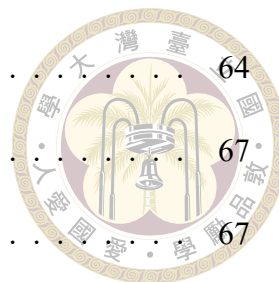
Contents

	Page
Acknowledgements	i
摘要	ii
Abstract	iii
Contents	v
List of Figures	viii
List of Tables	xii
Chapter 1 Introduction	1
1.1 Background	1
1.2 Research purpose	2
Chapter 2 Literature Review	5
2.1 Common insects in crop cultivation and management	5
2.1.1 Whiteflies & thrips	6
2.1.2 Honey bee	7
2.2 AIoT systems in crop cultivation and management	7
2.2.1 Environmental sensing	8
2.2.2 Monitoring of pests	8
2.2.3 Optimization of bee pollination	9



2.3	IoT technology	10
2.4	Deep learning algorithm	11
2.4.1	Object detection	11
2.4.2	Instance segmentation	26
2.4.3	Tracking algorithm	30
Chapter 3	Materials & Methods	35
3.1	Monitoring of pests	35
3.1.1	IoT system	36
3.1.2	Image preprocessing	38
3.1.3	Model training	41
3.1.4	Combining environmental monitoring and pest counting	43
3.2	Optimization of bee pollination	44
3.2.1	Image acquisition system	44
3.2.2	Image preprocessing	46
3.2.3	Counting of honey bees entering and leaving the hive	48
3.2.4	Calculation of pollen grain area	51
3.2.5	Correlation analysis between calculated area and actual weight of pollen grains	55
Chapter 4	Results & Discussion	56
4.1	Monitoring of pests	56
4.1.1	Model comparison	56
4.1.2	Results obtained by YOLOv5 model with “x6” backbone.	57
4.1.3	Combining environmental sensing and pest monitoring measures	63

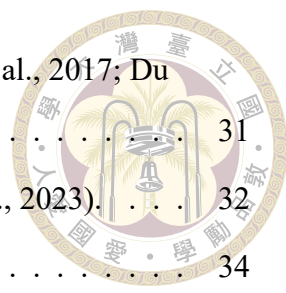
4.1.4	Practical applications	64
4.2	Optimization of bee pollination	67
4.2.1	Counting of honey bees entering and leaving the hive	67
4.2.2	Calculation of pollen grain area	73
4.2.3	Correlation analysis between pollen count and actual pollination status of crops	81
Chapter 5	Conclusion	85
Chapter 6	Future work	87
	References	88



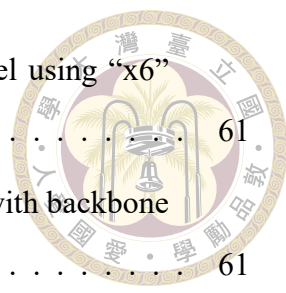


List of Figures

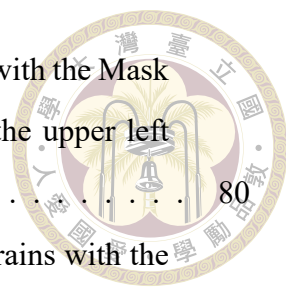
2.1	The architecture of the Faster R-CNN model (Ren et al., 2015).	13
2.2	The architecture of the YOLOv3 model, drawn according to (Redmon and Farhadi, 2018).	14
2.3	The architecture of the YOLOv5 (5.0) model, drawn according to (Jocher, 2020).	16
2.4	The architecture of the YOLOv5 (6.0) model, drawn according to (Jocher, 2020).	16
2.5	The architecture of the YOLOR model, drawn according to (Wang et al., 2021).	18
2.6	The architecture of the YOLOv6 model (Li et al., 2022).	19
2.7	The neck internal corresponding components of the YOLOv6 model version 3.0 (Li et al., 2023).	19
2.8	The architecture of the YOLOv7 model, drawn according to (Wang et al., 2023).	20
2.9	The architecture of the YOLOv8 model, drawn according to (Jocher et al., 2023).	21
2.10	The architecture of the RetinaNet model (Lin et al., 2017b).	24
2.11	The architecture of the swin transformer model (Liu et al., 2021).	25
2.12	The architecture of the swin transformer block (Liu et al., 2021).	25
2.13	The architecture of the Mask R-CNN model, drawn according to (He et al., 2017).	27
2.14	The architecture of the YOLACT model (Bolya et al., 2019, 2022).	28



2.15	The flow chart of DeepSORT tracking algorithm (Wojke et al., 2017; Du et al., 2023).	31
2.16	The flow chart of StrongSORT tracking algorithm (Du et al., 2023).	32
2.17	The flow chart of AFLink (Du et al., 2023).	34
3.1	Flow chart of pest monitoring.	35
3.2	Schematic of pest monitoring.	36
3.3	Locations of greenhouse nodes.	37
3.4	The sticky paper trap monitoring node.	38
3.5	Original images of sticky pest traps for (a) whiteflies and (b)-(c) thrips.	38
3.6	Optical wavelength test results of yellow sticky paper samples (IATP).	39
3.7	Optical wavelength test results of blue sticky paper samples (IATPBL).	39
3.8	Optical wavelength test results of white sticky paper samples (FATP).	39
3.9	(a) Original sticky pest trap image after division. (b) Image after HSV processing. (c) Image after CIELAB processing.	41
3.10	Flow chart of bee pollination optimization.	44
3.11	Schematic of bee pollination optimization.	45
3.12	The honey bee monitoring system.	47
3.13	Mechanisms for observation box version 2.	48
3.14	Images (a)-(c) show the observation aisle in version 1 of the observation box. Images (d)-(f) depict the observation aisle in version 2 of the observation box. The images are positioned below, to the left of, and to the right of the observation aisle.	49
3.15	(a) The pollen grains scraped by the mechanism. (b) The electronic scale (TP-214).	55
4.1	Results of pest detection with YOLOv5 model using “x6” backbone. The thrips sticky paper trap images are represented by blue and white rectangles.	58
4.2	Curve of precision versus recall for pest detection with YOLOv5 model using “x6” backbone.	60



4.3	Curve of F1 scores for pest detection with YOLOv5 model using “x6” backbone.	61
4.4	Confusion matrix of pests detected using YOLOv5 model with backbone “x6”.	61
4.5	Results of object classification and counting using YOLOv5 model with “x6” backbone	62
4.6	Combining sensing values and pest numbers. Boxplots of sensing values and line graphs of pest numbers from January 19, 2022, to February 25, 2022, for (a) air temperature and (b) air humidity.	65
4.7	Results showing variations in the whitefly population during the pest outbreak from January 01, 2020 to January 31, 2020.	67
4.8	The resulting bounding boxes of honey bee using the YOLOv8 model with “s” backbone.	69
4.9	Curve of precision vs. recall for honey bee with YOLOv8 model using “s” backbone.	70
4.10	Curve of F1 scores for honey bee with YOLOv8 model using “s” backbone.	70
4.11	Confusion matrix of honey bees detected with YOLOv8 model using “s” backbone.	71
4.12	The honey bee tracking results of version 1 (a) and version 2 (b) observation boxes using the YOLOv8 model with “s” backbone in conjunction with the StrongSORT (Du et al., 2023) tracking algorithm.	73
4.13	Scoring examples of segmentation score for version 1 and 2 observation boxes, where red numbers are segmentation scores.	76
4.14	The Mask R-CNN model with Swin-T backbone was used to obtain segmentation results for pollen grains in both version 1 and version 2 observation boxes. Images (a) and (b) show the segmentation results for pollen grains on the left and right sides of version 1. Images (c) and (d) display the segmentation results for pollen grains on the left and right sides of version 2.	79



4.15	The detection curve of precision vs. recall for pollen grains with the Mask R-CNN model using Swin-T backbone. The numbers in the upper left corner of the figure represent bounding box mAP.	80
4.16	The segmentation curve of precision vs. recall for pollen grains with the Mask R-CNN model using Swin-T backbone. The numbers in the upper left corner of the figure represent mask mAP.	80
4.17	Confusion matrix of pollen grains detected with the Mask R-CNN model using Swin-T backbone.	81
4.18	Images (a) and (c) show the segmentation results for pollen grains in version 1 and version 2 observation boxes, respectively. The Mask R-CNN model with Swin-T backbone was used for both. Images (b) and (d) depict the calculation results for version 1 and version 2, respectively, using the algorithm (white pixels indicate calculated pixels).	82
4.19	Regression curve of the experimental data on January 13, March 05, and March 06, 2023.	84



List of Tables

3.1	Training parameters of the models.	43
3.2	Training parameters of the models used by the tracking algorithm.	50
3.3	Training parameters of the instance segmentation models.	53
4.1	Comparison of the precision, recall, and mAP of the models.	57
4.2	Mean count accuracy per day for seven consecutive days.	63
4.3	Cross-correlation of environmental temperature and humidity with pest numbers on various days. d_0 , d_{-1} , d_{-2} , and d_{-3} represent the current day, 1 day prior, 2 days prior, and 3 days prior, respectively.	64
4.4	Comparison of the precision, recall, and mean average precision (mAP) of all models.	68
4.5	Comparison of the StrongSORT tracking accuracy of different model weight files.	72
4.6	Comparing the mAP of all instance segmentation models.	74
4.7	Comparing the segmentation result of all instance segmentation models.	77
4.8	The experimental data on January 13, 2023 ($1 \text{ mm}^2 = 256 \text{ pixels}$).	83
4.9	The experimental data on March 05, 2023 ($1 \text{ mm}^2 = 256 \text{ pixels}$).	83
4.10	The experimental data on March 06, 2023 ($1 \text{ mm}^2 = 256 \text{ pixels}$).	84

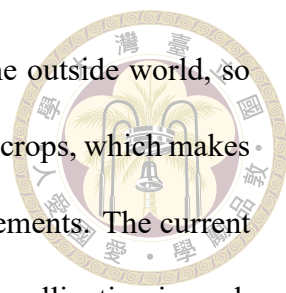


Chapter 1 Introduction

1.1 Background

In recent years, smart greenhouse agricultural technology has become increasingly mature. Farmers can automatically adjust the sunlight, temperature, water, and nutrient solution of planting in greenhouse facilities through the Internet of Things (IoT) technology and use smart cultivation technology to grow crops with high economic value (Liao et al., 2017). Although greenhouse cultivation can effectively control environmental factors and reduce the occurrence of pests and diseases, it still cannot completely isolate crop pests and diseases.

Asparagus is prone to infestation by Silverleaf whiteflies (*Bemisia tabaci*) (Guo et al., 2019) and thrips (*Thysanoptera*) (Hsieh et al., 2020; Guo et al., 2020). To enhance yields and reduce the pest population, growers employ passive pest control methods, classified into physical and chemical techniques. Yellow sticky paper traps are commonly utilized to observe and control pest activity. In cases where more pests are detected, asparagus is splashed with water, while severe cases call for the use of insecticides for chemical pest control (Guo et al., 2017, 2019). Asparagus' vulnerability to whitefly and thrip infestations can cause significant economic losses as it can weaken the plant, making it incapable of photosynthesizing and resulting in wilting of the stem and buckling of shoot tips.



In addition, greenhouse cultivation is also well isolated from the outside world, so pollinating insects such as honey bees and butterflies cannot pollinate crops, which makes greenhouse crops face great challenges in terms of pollination requirements. The current greenhouse pollination method is to allocate at least one beehive for pollination in each greenhouse, and each beehive must be equipped with an expensive queen bee, resulting in a very high cost in the pollination stage (Washitani et al., 1994). Although most farmers and beekeepers cooperate in pollination operations, whenever there is a need for honey production, the needs of beekeepers and farmers to use beehives will conflict, resulting in a gap in greenhouse crop pollination needs. If the number of beehives in the greenhouse is too few, pollination will be incomplete, and the yield will be affected. However, if the number of beehives is too large, the honey bee colony may hit the wall in the greenhouse.

Hence, there is a need to establish a comprehensive system that not only can track real-time pest populations but also can effectively optimize the pollination of bees. By doing so, this system can help mitigate economic losses for farmers in pest infestation while enhancing bees' pollination, thereby increasing overall profits.

1.2 Research purpose

Starting from the emergence of precision agriculture during the 1990s (Heuvel, 1996), modern agricultural technology has made significant strides towards enhancing the efficiency of agricultural production and resolving the labor shortage challenges faced by the industry. The practical implementation of the IoT and Artificial Intelligence (AI) in smart agriculture (Li et al., 2020a) has enabled farmers to monitor crop growth remotely and receive recommendations for planting. This development marks a noteworthy milestone

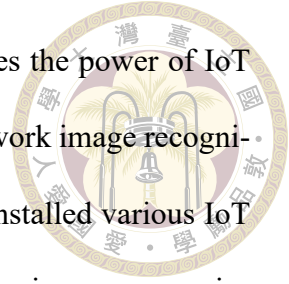
in the agricultural sector's quest for advanced solutions.



Over the past few years, Convolutional Neural Networks (CNN) has become a ubiquitous AI technique for detecting objects. The development of deep learning algorithms has enabled the detection of object features, facilitating various detection tasks (Liu and Wang, 2021). Compared to traditional image processing techniques, these methods have become more prominent. Object detection, a widely-researched field, utilizes bounding boxes to select object boxes (Zhao et al., 2019; Zou et al., 2023; Xu et al., 2022). For instance, the two-stage model, Faster R-CNN (Ren et al., 2015), was developed to achieve high detection accuracy, but it has a lengthy inference time. Currently, the YOLO model family (Redmon and Farhadi, 2018; Jocher, 2020; Wang et al., 2023; Jocher et al., 2023; Li et al., 2023), the most commonly used one-stage model, accelerates model inference while maintaining comparable accuracy. The Transformer model family (Liu et al., 2021; Xu et al., 2022), known for its exceptional performance in natural language processing, has recently garnered significant attention in computer vision. These models have all established themselves as leading object detection techniques.

Instance segmentation is a relatively new research topic (Tian et al., 2022). At present, there is still a problem of excessive computation for real-time image recognition. It is to first select objects through the bounding box, and then mark the object area in the bounding box. An example of an early two-stage model, Mask R-CNN (He et al., 2017), has been improved by leveraging the architecture of Faster R-CNN (Ren et al., 2015). The one-stage models currently available for instant application are the YOLACT model (Bolya et al., 2019, 2022) and the YOLO model (Jocher, 2020; Jocher et al., 2023). The Transformer models are among the relatively new models (Liu et al., 2021; Xu et al., 2022). These models are all instance segmentation models.

The goal of this study is to create an AIoT system that harnesses the power of IoT and AI technology to optimize crop insect management as neural network image recognition technology continues to progress. To achieve this goal, I have installed various IoT sensors that can monitor the crop-growing environment while leveraging more precise deep-learning models to monitor insect populations. The primary objective is to enhance the pollination of bees and minimize the risk of pest outbreaks. Through the analysis of big data, this study aims to uncover potential correlations between ambient changes and insect activity, leading to improved insect control in crop fields.





Chapter 2 Literature Review

2.1 Common insects in crop cultivation and management

In recent years, the intensification of climate change worldwide has underscored the significance of protected cultivation as a vital solution for safeguarding crops against extreme weather conditions. As the area dedicated to facility planting continues to expand rapidly, the industry has recognized the pressing challenges posed by the high instability and cost associated with facility pest monitoring and pollination. In light of the climate change impact, facility planting has emerged as the most effective response. Notably, the area of protected cultivation in Taiwan has seen significant growth in the past decade, accompanied by an increase in the number of protected cultivation farmers who now enjoy considerably higher incomes compared to traditional farmers. Given Taiwan's limited farmland, farmers have prioritized maximizing crop yield per unit area through the use of indoor greenhouses. These greenhouses provide precise control over crop growth by allowing farmers to regulate environmental conditions and minimize pest infestations using protective nets. However, despite the comprehensive protection offered by four-sided nets, pest outbreaks can still inflict significant damage on crops in the absence of effective pest control strategies. Furthermore, as the greenhouse environment is isolated from the outdoors, farmers must manually introduce beehives to ensure proper pollination. To

address these challenges, it is imperative to monitor pests and optimize pollination within the greenhouse setting.



2.1.1 Whiteflies & thrips

The common small pests of facility asparagus are Silverleaf whitefly and thrips. Silverleaf whitefly is small in size, has strong migration ability, and high reproductive potential. When the population density is high, its secretions often induce soot mold, hinder the photosynthesis of asparagus leaves, and affect the yield (Guo et al., 2017, 2019).

The thrips are small in size and highly concealed. The nymphs often hide in the scales of the tender stems after hatching, and suck the sap in the tissues through the rasping-mouthparts, which not only causes the shoot tips of the tender stems to bend and shrink, but also causes the scales to brown. This directly affects the value of the commodity and causes damage to the mother stem (Hsieh et al., 2020; Guo et al., 2020).

In outdoor cultivation environments, the severity of infestations from these two small pests tends to increase during dry and warm periods. However, in the controlled environment of a greenhouse, they can occur annually and persist at high population densities over extended periods. Therefore, it becomes crucial to implement effective monitoring measures, such as utilizing sticky paper traps and employing image recognition technology, to accurately assess and count the pest population within the greenhouse. These monitoring techniques are essential for timely intervention and efficient pest management strategies.



2.1.2 Honey bee

In natural conditions, approximately 80% of human food relies on pollination by insects (Solomon, 1971; Kevan, 1999; Delaplane et al., 2000; Klein et al., 2006). However, the adoption of facility cultivation methods hinders the natural pollination process. Consequently, the use of chemical agents or labor-intensive manual pollination practices becomes necessary, resulting in reduced fruit quality, lower production efficiency, and increased costs (Chautá-Mellizo et al., 1999; Sáez et al., 2019). To address these challenges, some facility farmers have introduced bumblebees or honey bee colonies into greenhouses to aid in pollination (Fisher and Pomeroy, 1989; Sabara and Winston, 2003; Sun et al., 2021). Among these methods, honey bee colonies are the most commonly employed for pollination assistance. However, the confinement of the greenhouse environment causes field bees, essential for crop pollination, to attempt to escape, leading to a significant number of field bee deaths and hindering efficient pollination (Nicodemo et al., 2018). Thus, it becomes necessary for experts to enhance the pollination performance of honey bee hives and utilize image recognition technology to optimize honey bee pollination in the greenhouse setting. These advancements will contribute to improving pollination efficiency and overall crop productivity.

2.2 AIoT systems in crop cultivation and management

In greenhouses, environmental controls are frequently established based on farmers' heuristics rather than objective quantitative analysis. As a result, crop growers must devise effective pest management strategies and pollination optimization methods to avoid pest infestation and monitor honey bee activity in the greenhouse while enhancing the

pollination of bees.

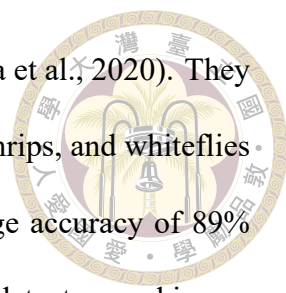


2.2.1 Environmental sensing

Integrating IoT (Ajao et al., 2017; Maraveas and Bartzanas, 2021) and AI technology (Rupnik et al., 2017; Tripathy et al., 2021; Chang et al., 2021) with the environmental monitoring system of crop greenhouses can greatly enhance the precision of crop growth control. By utilizing various sensors, the greenhouse environment can be monitored and relevant data wirelessly transmitted for remote monitoring and subsequent analysis. Studies have demonstrated the practical application of IoT-based systems for environmental monitoring in farmland (Ajao et al., 2017; Maraveas and Bartzanas, 2021). Rupnik et al. developed a cloud-based decision system that utilized random forest algorithm (Rupnik et al., 2017). Tripathy et al. presented an IoT system that provides decision support for the challenges encountered in growing roses in a greenhouse (Tripathy et al., 2021). Chang et al. proposed an AIoT approach that combines IoT and AI technology to predict leaf lettuce (*Lactuca sativa L.*) growth-related physiological parameters. The data collected through IoT technology and imaging systems is analyzed using fuzzy logic and integrated neural networks (Chang et al., 2021). These studies illustrated the effectiveness and feasibility of the AIoT system in providing decisive management information to farmers.

2.2.2 Monitoring of pests

Different deep learning algorithms have been suggested for detecting various pest types on agricultural fields, including whiteflies and thrips. Rustia et al. employed image processing techniques and support vector machines (SVM) to classify and count insects on

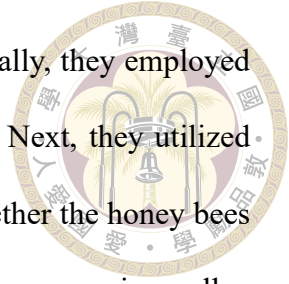


tomato seedling farms, achieving an average accuracy of 93% (Rustia et al., 2020). They also used cascaded neural networks to detect and count flies, gnats, thrips, and whiteflies on tomato seedling and *Lisianthus (Eustoma)* farms, with an average accuracy of 89% (Rustia et al., 2018). Furthermore, Rustia et al. utilized CNN object detectors and image classifiers to detect and count gnats, mothflies, shoreflies, and whiteflies in vegetable seedlings and cabbage farms, achieving an average accuracy of 91% and 90%, respectively (Rustia et al., 2021). They also analyzed a correlation between the pest count results and environmental perception values, enabling farmers to predict pest trends based on ambient conditions (Rustia et al., 2020). In another study, Chen et al. utilized the deep learning model YOLOv3 for object detection to locate *Tessaratoma papillosa* achieving a detection accuracy of 90%. In addition, Long Short-Term Memory (LSTM) was used to analyze environmental information from weather stations and predict the occurrence of pests (Chen et al., 2020).

2.2.3 Optimization of bee pollination

The typical method for evaluating the pollination effectiveness of honey bees is by counting the number of bees entering and exiting their hive. Ngo et al. introduced an image processing technique utilizing a Kalman Filter and Hungarian algorithm for tracking to automatically count honey bees entering and leaving their hive. Their method achieved an automatic counting accuracy of $93.9 \pm 1.1\%$ compared to manual counting (Ngo et al., 2019). Additionally, Ngo et al. employed a deep learning model that used the same tracking algorithm to count bees entering and leaving the hive. The average percent errors for the count of pollen-bearing honey bees and total incoming honey bees were $8.45 \pm 2.72\%$ and $10.55 \pm 2.10\%$, respectively (Ngo et al., 2021). Yang et al. implemented a multi-step

approach for their study, achieving a measurement error of 7%. Initially, they employed image processing techniques and a Kalman filter to track the bees. Next, they utilized an object detection model to detect pollen grains and distinguish whether the honey bees were carrying them. Finally, a bee count specifically focusing on those carrying pollen grains was conducted (Yang and Collins, 2019).



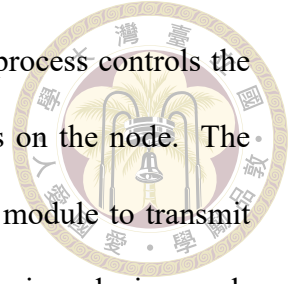
2.3 IoT technology

Wireless Sensor Networks (WSNs) play a crucial role in the realm of IoT technology (Ouni and Saleem, 2022). WSNs consist of front-end data detection and back-end data analysis components. The gateway serves as the intermediary, transmitting data collected by wireless sensor nodes to the back-end database for analysis. These nodes are equipped with sensing, computing, and wireless transmission capabilities to gather environmental data. The collected data is then organized and categorized through back-end analysis, catering to various purposes in different domains.

To enable seamless communication within WSNs, multiple protocols have been proposed, including the IEEE 802.11 standard for Wi-Fi and Zigbee. Wi-Fi offers a transmission distance of 100 meters and a speed of 54 Mbps. However, its higher power consumption and costs restrict its suitability for certain applications. In contrast, Zigbee provides a low-power alternative with a shorter transmission distance of 50-100 meters, a rate of 250 Kbps, and support for diverse network topologies (Baronti et al., 2007). Zigbee has gained wide adoption in the industry due to its cost-effectiveness and user-friendly nature.

A typical wireless sensor node comprises essential components such as a microprocessor control unit, memory, power supply module, sensor module, analog-to-digital con-

verter (ADC), and wireless communication module. The back-end process controls the system program, allowing for the management of hardware devices on the node. The node's operating system interacts with the wireless communication module to transmit data to other nodes or gateways. The gateway, implemented using various devices, collects sensor information and facilitates its transmission to the external network.



WSNs have been specifically developed to meet the demands of wide-area monitoring in the IoT landscape. They provide real-time information and exhibit high adaptability, allowing for the integration of different sensors based on specific user requirements. WSNs find applications in diverse fields, including ecological monitoring, marine ecological monitoring, medical care, and smart grid systems (Caicedo-Ortiz et al., 2018). These applications harness the capabilities of wireless sensor nodes, encompassing communication chips, sensing elements, and computing processors.

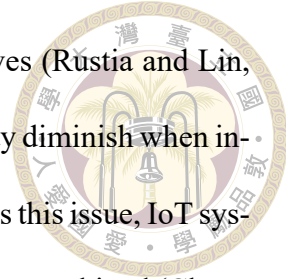
In summary, WSNs serve as a robust solution for collecting, transmitting, and analyzing data from sensor nodes in the context of IoT. They offer a wide range of applications across various domains, presenting a powerful tool for addressing monitoring needs and facilitating efficient data utilization in the ever-expanding IoT ecosystem.

2.4 Deep learning algorithm

2.4.1 Object detection

In order to respond promptly to pest infestations and enhance bee pollination, IoT sticky paper trap monitoring nodes are commonly used to obtain trap images and beehive entrances. Previously, image processing algorithms were utilized to count pests captured

on yellow sticky paper traps and honey bees entering and exiting hives (Rustia and Lin, 2017; Ngo et al., 2019). However, the accuracy of these algorithms may diminish when insects of various kinds have comparable appearance features. To address this issue, IoT systems and AI algorithms including convolutional neural networks can be combined (Chang et al., 2022; Chou et al., 2023). Furthermore, since the counting of pests and honey bees is focused on the number, using an object detection model is suitable.



2.4.1.1 Faster R-CNN

The model architecture of Faster R-CNN is shown in Fig. 2.1 (Ren et al., 2015). Faster R-CNN is a two-stage model that divides the detection step into two stages. In the first stage, Faster R-CNN builds Feature Pyramid Network (Lin et al., 2017a) through multi-layer convolutional layers, and uses ReLU activation function (Agarap, 2018) and pooling layer. Anchors then grab features on the generated feature maps. Region Proposal Networks judges that the features captured by anchors belong to positive or negative through softmax, and then uses bounding box regression to correct anchors to obtain accurate region proposals. The ROI Pooling layer collects input feature maps and proposals, and extracts proposal feature maps after synthesizing this information. In the second stage, the proposal feature maps are used to calculate the proposal category through the fully connected layer, and the bounding box regression is used again to obtain the final precise position of the detection frame.

2.4.1.2 YOLOv3

The model architecture of YOLOv3 is shown in Fig. 2.2. The three basic components of YOLOv3 are CBL (Conv+BN+Leakyrelu), Bottleneck, and Bottleneck_X (Redmon

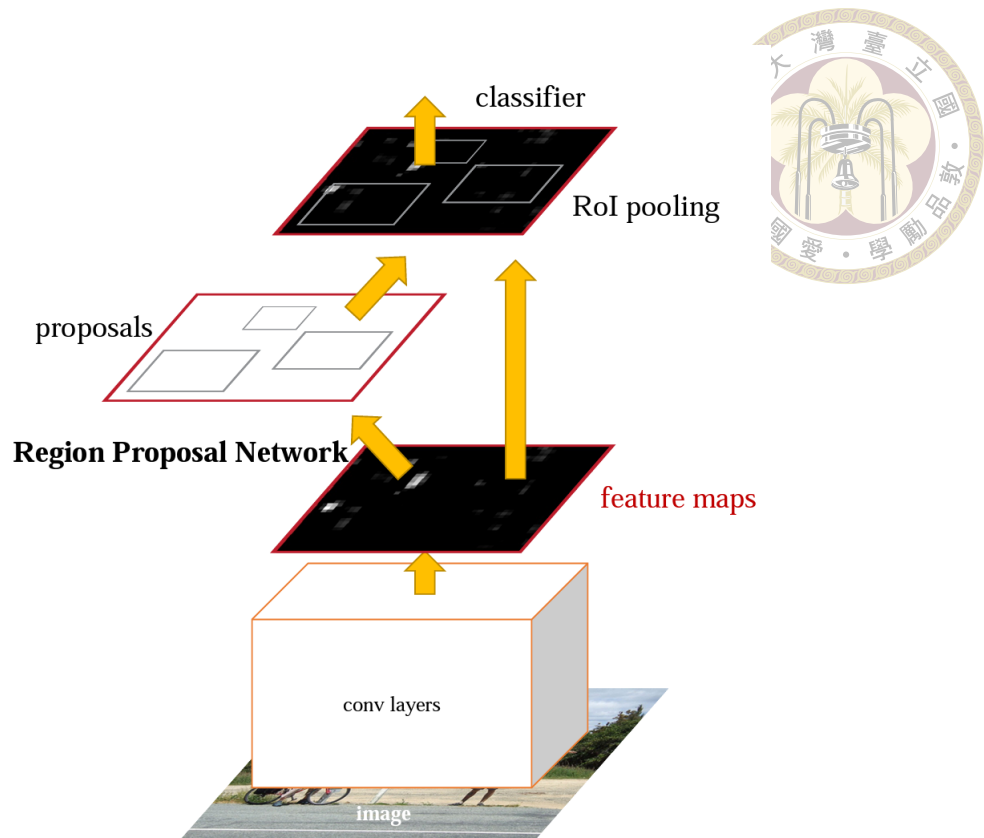


Figure 2.1: The architecture of the Faster R-CNN model (Ren et al., 2015).

and Farhadi, 2018). CBL is the smallest component in the YOLOv3 network structure, consisting of the convolution layer, batch normalization (BN), and LeakyReLU activation function. Bottleneck is based on the residual structure in the Resnet network, so that the network can be built deeper. Bottleneck_X consists of a CBL and X Bottleneck components, where the CBL in front of each Bottleneck plays the role of downsampling. In the YOLOv3 architecture, after a CBL and 5 Bottlenecks, a Darknet53 without fully connected layer can be formed.

2.4.1.3 YOLOv5

Building upon the research achievements of YOLOv3 (Redmon and Farhadi, 2018), the YOLOv5 model was introduced in 2020 (Jocher, 2020). However, due to ongoing improvements, YOLOv5 has undergone different versions. In the subsequent discussion,

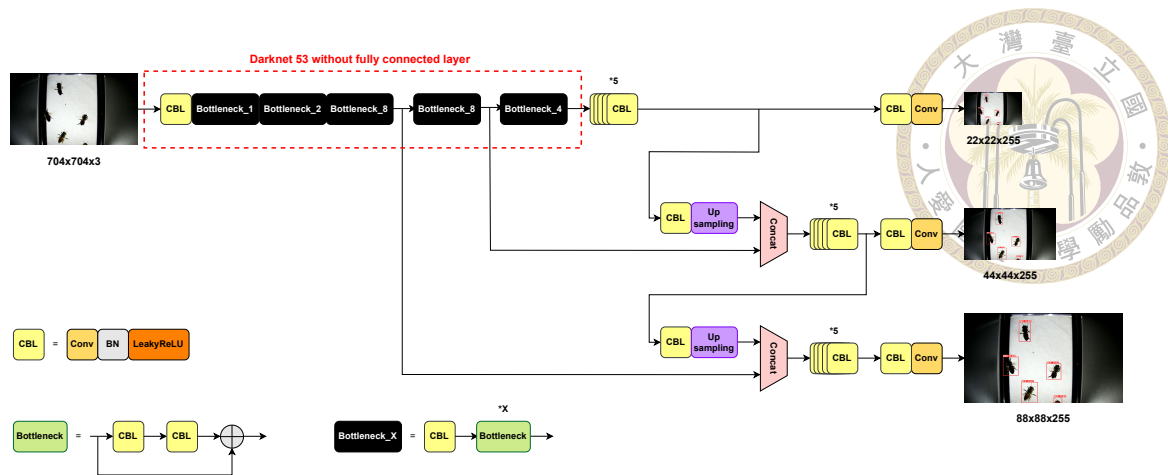
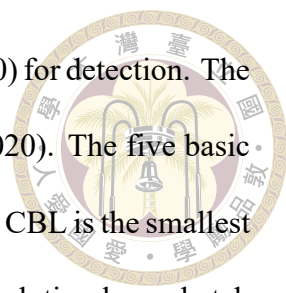


Figure 2.2: The architecture of the YOLOv3 model, drawn according to (Redmon and Farhadi, 2018).

YOLOv5 (5.0) refers to the 5.0 version of YOLOv5, while YOLOv5 (6.0) represents the 6.0 version of YOLOv5. The YOLOv5 (5.0) model architecture can be mainly divided into four parts: input layer, backbone layer, neck layer and detection layer. In the input layer, Mosaic data augmentation (Bochkovskiy et al., 2020), adaptive anchor box calculation and adaptive image scaling are mainly added based on YOLOv3. Mosaic data augmentation can increase dataset diversity, model robustness, and reduce GPU usage. The backbone layer adds Focus architecture and Cross Stage Partial Network (Wang et al., 2020a). The Focus architecture increases the amount of model computation and parameters, so that there is no information loss during downsampling. The Cross Stage Partial Network reduces the computational bottleneck of the model, the memory cost required for computation, and enhances the learning ability of the model, making it lightweight while maintaining accuracy. The neck layer is changed to an Feature Pyramid Networks (FPN) (Lin et al., 2017a) + Path Aggregation Network (PAN) (Liu et al., 2018) architecture, which conveys semantic features through upsampling, and conveys localization features through downsampling, which improves the ability of parameter aggregation and feature extraction.



The detection layer uses the CIoU loss function (Zheng et al., 2020) for detection. The model architecture of YOLOv5 (5.0) is shown in Fig. 2.3 (Jocher, 2020). The five basic components of YOLOv5 are CBL, Bottleneck, C3_X, Focus, and SPP. CBL is the smallest component in the YOLOv5 network structure, consisting of the convolution layer, batch normalization (BN), and SiLU activation function. Bottleneck is divided into two types, one is based on the residual structure in the Resnet network, so that the network can be built deeper, and the other is composed of two CBLs alone. C3_X is also divided into two types by matching X different Bottlenecks. Both are composed of CBL and Bottleneck. One is applied to the backbone network, and the other is mainly applied to Neck. Focus utilizes slicing operations and increases the amount of model computation and parameters, so that there is no information loss during downsampling. SPP uses the maximum pooling method to perform multi-scale fusion. The model architecture of YOLOv5 (6.0) is shown in Fig. 2.4 (Jocher, 2020). The YOLOv5 (6.0) mainly makes five changes based on the YOLOv5 (5.0): replacement of Focus with an equivalent layer for improved exportability, new SPPF replacement for SPP layer for reduced ops, placement of SPPF at the end of backbone, reintroduction of shortcut in the last C3 backbone layer, and increase of mixup and copy-paste augmentation.

YOLOv5 (5.0) and YOLOv5 (6.0) employ four main types of loss functions: box loss, object loss, class loss (Redmon et al., 2016), and segment loss (Jocher, 2020). The CIoU loss (Zheng et al., 2020) serves as the box loss, quantifying the disparity between the predicted and actual bounding box as:

$$\text{CIoU loss} = 1 - \text{CIoU} = 1 - \left[\text{IoU} - \frac{\rho^2(b, b^{gt})}{c^2} - \alpha v \right], \quad (2.1)$$

where $\rho^2(b, b^{gt})$ denotes the distance between the centers of the predicted and ground-truth

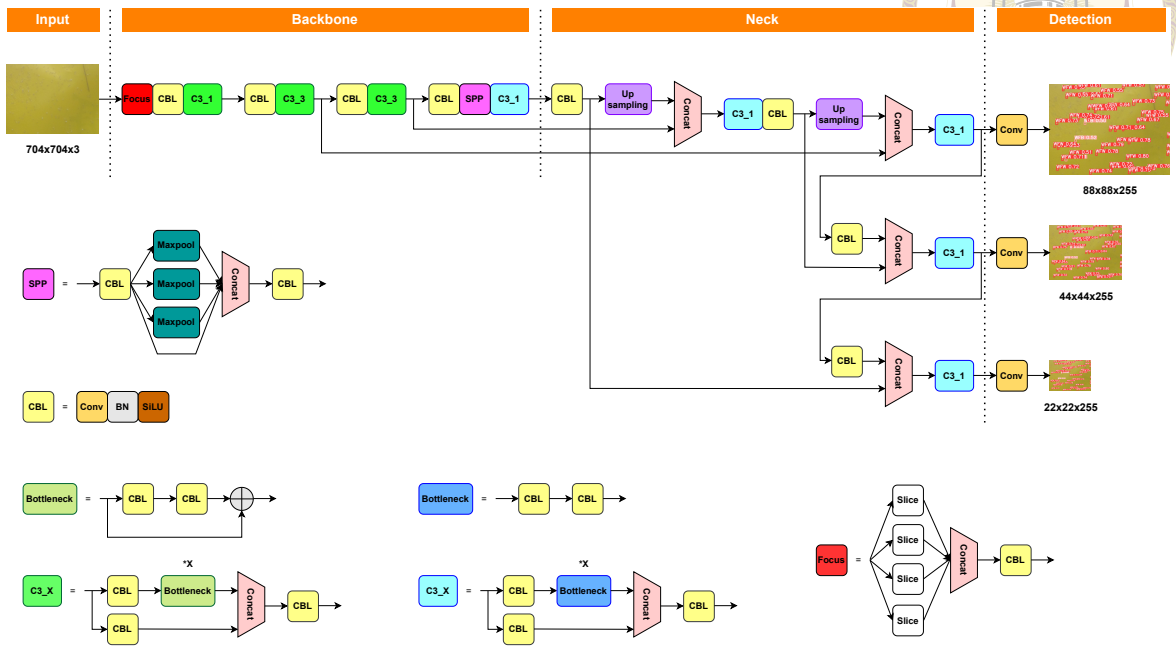


Figure 2.3: The architecture of the YOLOv5 (5.0) model, drawn according to (Jocher, 2020).

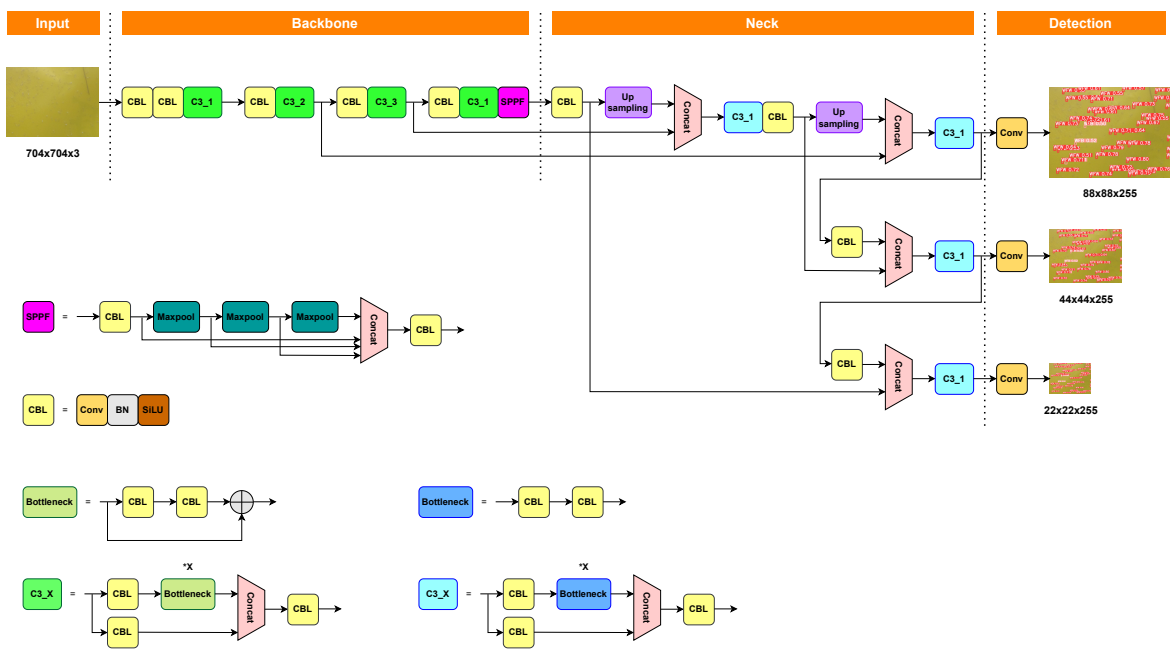


Figure 2.4: The architecture of the YOLOv5 (6.0) model, drawn according to (Jocher, 2020).

bounding boxes, calculated using the Euclidean distance. b and b^{gt} denote the center point of the predicted and ground-truth bounding boxes, respectively. c indicates the diagonal distance of the minimum area that is able to fully enclose both the predicted and ground-truth bounding boxes. α and v represent a positive trade-off and a measurement of aspect ratio consistency, respectively.

The object loss initially verifies whether there are objects present within each bounding box, and then utilizes the CIoU value to calculate the BCEWithLogitsLoss, represented as:

$$\text{BCEWithLogitsLoss} = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log \sigma(x_i) + (1 - y_i) \cdot \log(1 - \sigma(x_i)), \quad (2.2)$$

where N denotes the batch size. y_i is a binary variable that determines whether an object is present in the bounding box. x_i represents the CIoU value of the predicted and ground-truth bounding box calculated in Eq. (2.1), and σ is the sigmoid function. To calculate the class loss, BCEWithLogitsLoss is applied to the predicted and ground truth categories, following the same format as Eq. (2.2). In this equation, y_i represents the ground truth class, x_i represents the predicted class, and N denotes the batch size. The segment loss is computed by calculating BCEWithLogitsLoss on the predicted mask and the ground truth mask using the same form as Eq. (2.2). N , y_i , and x_i represent the batch size, the ground truth mask, and the predicted mask, respectively.

2.4.1.4 YOLOR

The YOLOR (You Only Learn One Representation) model was introduced in 2021 (Wang et al., 2021), featuring its architecture which is depicted in Fig. 2.5. The basic

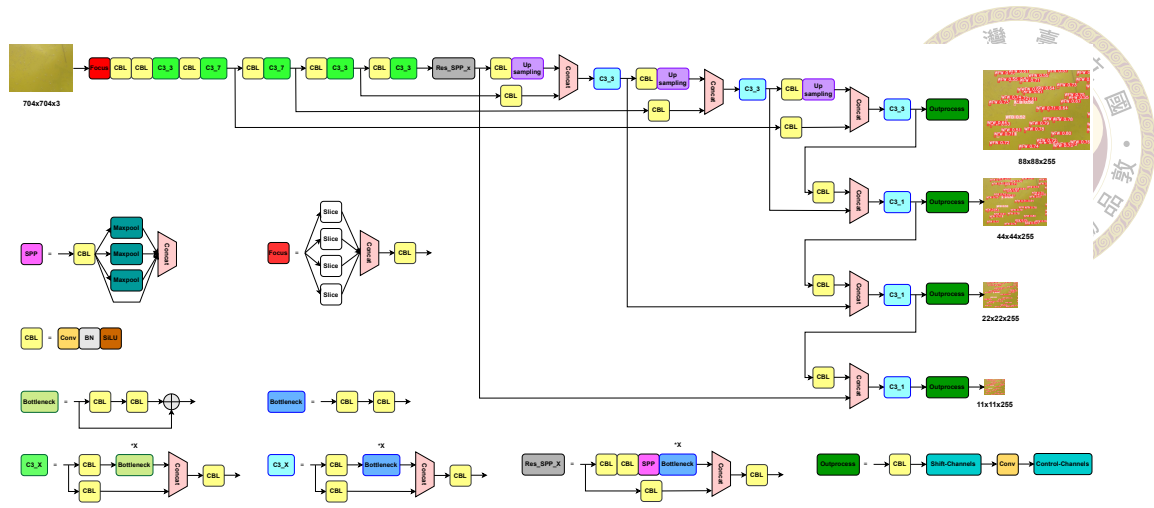


Figure 2.5: The architecture of the YOLOR model, drawn according to (Wang et al., 2021).

components of YOLOR are similar to YOLOv5, Implicit knowledge is introduced, and two main components are added: Res_SPP_X and outprocess. The Res_SPP_X module is mainly composed of CBL and SPP, combined with residual structure. The Outprocess module is composed of CBL, shift-channels, convolution layer and control-channels. Shift-channels and control-channels extract the Implicit knowledge vector, which is similar to the self-attention in the transformer model (Xu et al., 2022).

2.4.1.5 YOLOv6

The YOLOv6 model was officially released in 2022 (Li et al., 2022), and its model architecture can be observed in Fig. 2.6. YOLOv6, built upon the architecture of the YOLOv5 model, introduced three major enhancements. Firstly, there was a uniform design for a more efficient Backbone and Neck. Drawing inspiration from the hardware-aware neural network concept, an EfficientRep Backbone and Rep-PAN Neck were devised based on the RepVGG style (Ding et al., 2021). These components offer reparameterizability and enhanced efficiency. Secondly, a concise and effective Efficient Decoupled Head was optimized and designed to minimize additional delays typically caused by general decoupled heads while maintaining accuracy. Lastly, the training strategy in-

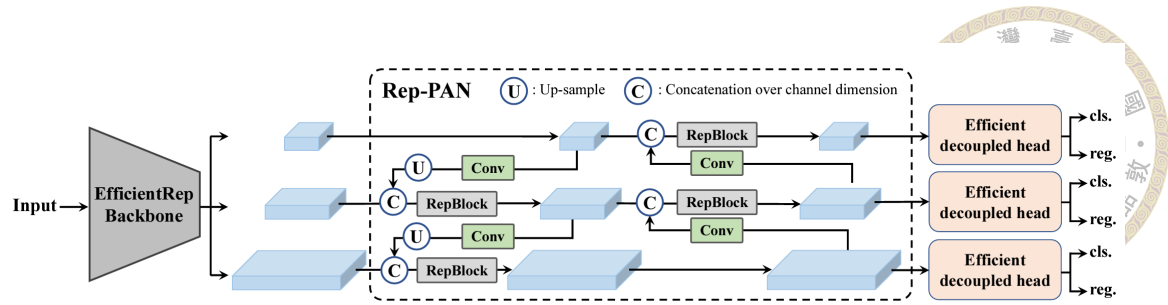


Figure 2.6: The architecture of the YOLOv6 model (Li et al., 2022).

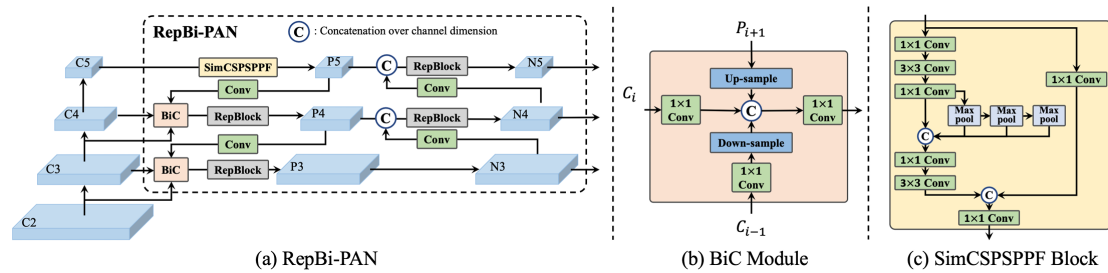


Figure 2.7: The neck internal corresponding components of the YOLOv6 model version 3.0 (Li et al., 2023).

volved adopting an anchor-free paradigm, complemented by the SimOTA label assignment strategy (Ge et al., 2021) and the SIoU bounding box regression loss (Gevorgyan, 2022). This combination aimed further to enhance the detection accuracy of the YOLOv6 model.

In 2023, YOLOv6 received an update to version 3.0 (Li et al., 2023), and the research team asserted that its accuracy surpassed that of YOLOv7 and YOLOv8. This enhanced version is built upon the previous architecture while incorporating three key modifications. Firstly, a reparameterizable bidirectional fused PAN (RepBi-PAN) Neck network with enhanced representation capability was designed, as shown in Fig. 2.7. Secondly, a new Anchor-Aided Training strategy was introduced. Lastly, a Decoupled Location Distillation strategy was proposed to elevate the performance of smaller models. These updates collectively contributed to the improved accuracy and capabilities of YOLOv6 model version 3.0.

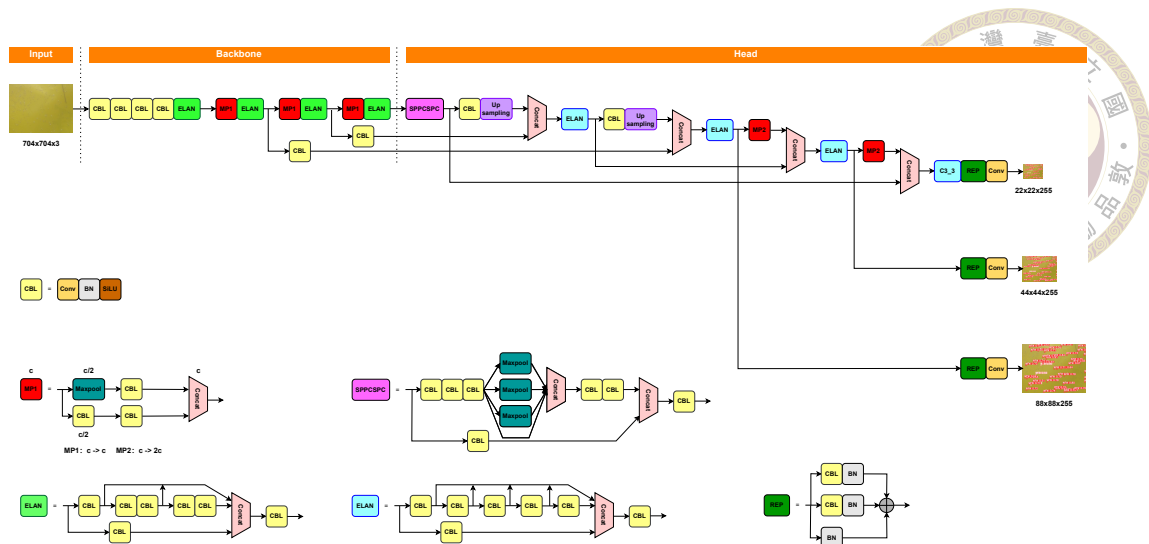


Figure 2.8: The architecture of the YOLOv7 model, drawn according to (Wang et al., 2023).

2.4.1.6 YOLOv7

In 2022, the YOLOv7 model was introduced (Wang et al., 2023). YOLOv7 first resizes the input image and inputs it into the backbone network, then outputs three layers of feature maps of different sizes through the head layer network, and finally outputs the prediction results through the Rep and convolution layers. The model architecture of YOLOv7 is shown in Fig. 2.8. The ELAN module is divided into two types, both of which are composed of multiple CBLs. The main difference between the two is that the number of concatenations is different and they are applied to the backbone and head networks respectively. The MP module is mainly composed of Maxpool and CBL, among which MP1 and MP2 mainly change the ratio of the number of input and output channels. Rep consists of CBL and BN. It adjusts the number of channels of P3, P4 and P5 feature maps output by PAFPN (Liu et al., 2018), and uses 1x1 convolution to predict objectness, class, and bbox.

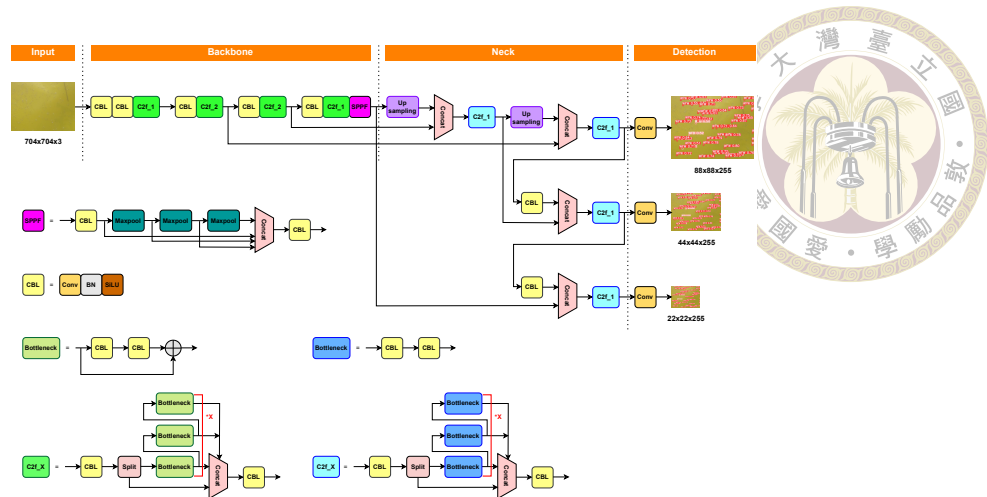


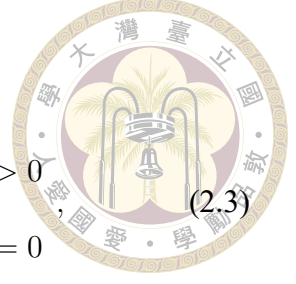
Figure 2.9: The architecture of the YOLOv8 model, drawn according to (Jocher et al., 2023).

2.4.1.7 YOLOv8

The YOLOv8 model was introduced in early 2023, featuring its architecture depicted in Fig. 2.9. Its idea is based on YOLOv5, and it is mainly improved for modules, loss functions and anchors (Jocher et al., 2023). First, the C3_X module that is distributed throughout the YOLOv5 model is replaced by the C2f_X module. The main idea of the module is still CSPNet (Wang et al., 2020a), but instead of vertical operations, parallel operations are used to reduce the amount of model computation without losing feature information. In addition, YOLOv8 employs four main types of loss functions: box loss, class loss, DFL, and segment loss. The box loss and segment loss are the same as YOLOv5, using CIoU loss and BCEWithLogitsLoss respectively, as shown by Eq. (2.1) and Eq. (2.2). The class loss is changed from BCEWithLogitsLoss to Varifocal Loss (VFL) (Zhang et al., 2021). When the classification is correct, it is similar to BCEWithLogitsLoss and weighted to highlight the object classification; when it is wrongly predicted, it reduces the impact on

loss value, as shown in Eq. (2.3):

$$\text{VFL}(p, q) = \begin{cases} -q(q \log(p) + (1 - q) \log(1 - p)) & q > 0 \\ -\alpha p^\gamma \log(1 - p) & q = 0 \end{cases} \quad (2.3)$$



where p represents the classification score (predicted classification probability). q is the Intersection over Union (IoU) of the predicted bounding box and ground truth bounding box, otherwise it is 0. α is the constant of proportionality. γ is a hyperparameter.

DFL represents Distribution Focal Loss (Li et al., 2020b), which models the possible bounding box location information as a general distribution, allowing the model to predict the bounding box to quickly focus on the target location, as shown in Eq. (2.4):

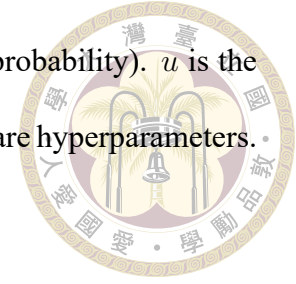
$$\text{DFL}(S_i, S_{i+1}) = -((y_{i+1} - y) \log(S_i) + (y - y_i) \log(S_{i+1})), \quad (2.4)$$

where S_i and S_{i+1} are the softmax results of the confidence score (predicted positioning probability) of two predicted bounding boxes closest to the ground truth bounding box. y_i and y_{i+1} are two predicted bounding box coordinate information. y is the coordinate information of the ground truth bounding box.

Finally, YOLOv8 replaces the anchor pairing method from anchor-based to anchor-free, which can directly grab the center point of the object and predict its distance to the four sides to build anchor box. And the Task-Aligned Assigner guides the model to focus on anchors with better positioning and classification results through the metric and the aforementioned loss function. The metric is shown in Eq. (2.5):

$$t = s^\alpha + u^\beta, \quad (2.5)$$

where s represents the classification score (predicted classification probability). u is the IoU between the anchor box and ground truth bounding box. α and β are hyperparameters.



2.4.1.8 RetinaNet

To assess the performance between the transformer series model and other models, the RetinaNet model with the Swin-T backbone was utilized for identifying pests. Therefore, the subsequent discussion also includes an introduction to the RetinaNet model. The model backbone is Resnet-based FPN (Lin et al., 2017b). FPN is a bottom-up, top-down, and horizontally connected network structure. Through horizontal connections, features at different levels can be integrated to enhance the feature extraction capabilities of the network (Lin et al., 2017a). From Fig. 2.10, we can know that each layer of FPN connects subnets, and the weights of these subnets are shared within themselves. The class subnet is used to predict the prediction probability of K categories for each anchor. The network model uses a total of five layers, four of which are Fully Convolutional Network (FCN) networks using the ReLU activation function, and the last layer is FCN networks using the Sigmoid activation function. The dimension of the last layer is $3 \times 3 \times K A$ because there are K -dimensional one-hot vectors for A anchors, representing the prediction probability of each category. The box subnet is the same as the class subnet. The difference is the dimension of the last layer because the box subnet is used to predict the offset from the ground truth position (x, y, w, h) .

2.4.1.9 Swin transformer

Unlike the previous Convolutional Neural Network (CNN), which uses the convolutional layer to extract features, the transformer series model uses self-attention to calculate

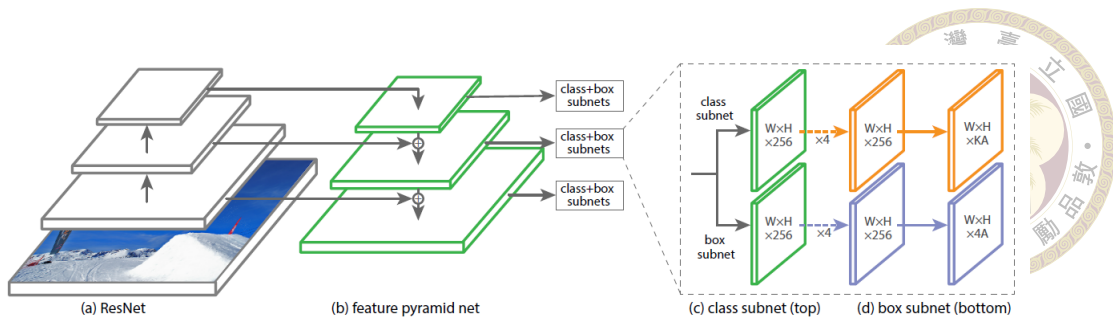


Figure 2.10: The architecture of the RetinaNet model (Lin et al., 2017b).

the current pixel features, and calculates the relationship with other pixels at the same time, similar to combining the functions of CNN and Recurrent Neural Network (RNN). However, compared with the shorter audio length in the field of Natural Language Processing (NLP), the application of transformers to high-resolution images requires a large amount of computation, which leads to higher training difficulties and longer time-consuming (Xu et al., 2022).

Swin transformer is similar to a backbone and can be applied to most imaging tasks, such as object detection or instance segmentation (Liu et al., 2021). It mainly divides an image into multiple windows for self-attention, and allows the model to see a larger field of view through a multi-layer structure, similar to CNN. The model architecture of swin transformer is shown in Fig. 2.11. The operation process of the swin transformer mainly includes four stages, but the operation mode of each stage is almost the same, only the size of the data processed is different. Each stage includes patch merging and swin transformer block, and patch merging includes patch partition and linear embedding. Patch merging captures features by setting the window size, similar to CNN pooling, but without losing information. The swin transformer block is shown in Fig. 2.12, which contains window multi-head self-attention (W-MSA) and shifted-window multi-head self-attention (SW-MSA). W-MSA performs self-attention calculation on each pixel in the window compared with calculating the entire image, the computational complexity is reduced to linear

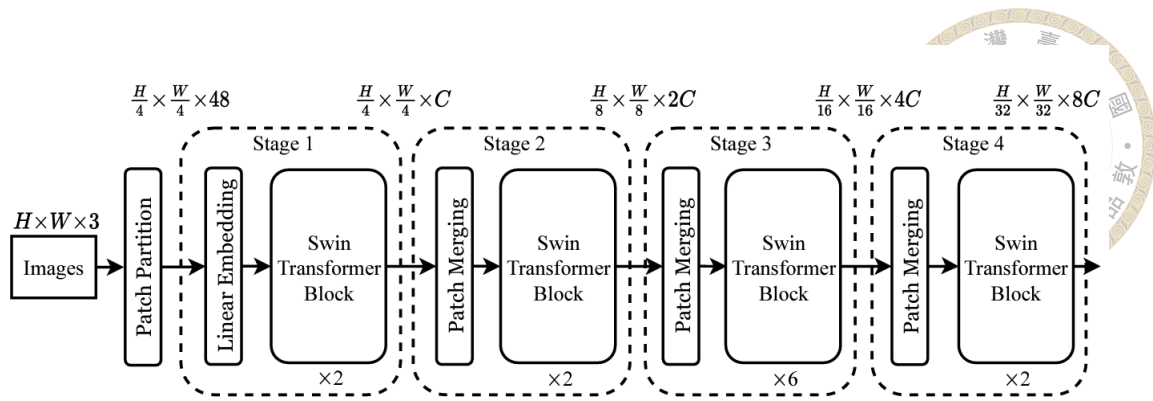


Figure 2.11: The architecture of the swin transformer model (Liu et al., 2021).

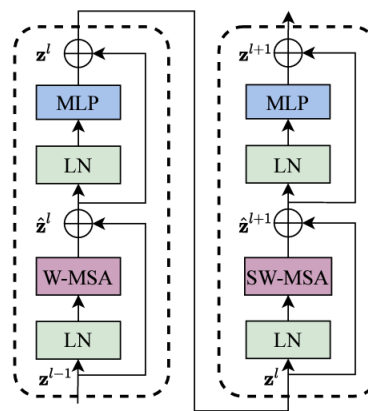
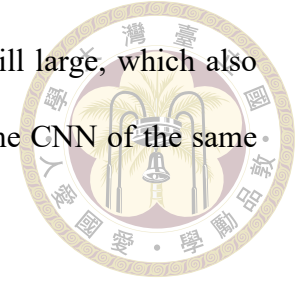


Figure 2.12: The architecture of the swin transformer block (Liu et al., 2021).

complexity. In addition, SW-MSA moves the window to the upper left by half the size of the window, so that the middle window can obtain the information of all windows in the upper layer, and then learn the correlation with other windows. Finally, after the Swin Transformer completes the calculation in all stages, it first obtains a feature vector with a length of 768 through a Global Average Pooling, and then obtains the final prediction result through a LayerNorm (LN) and a fully connected layer.

Currently, the transformer is a newer image recognition method, and the swin transformer can also be used as a general purpose backbone through many designs similar to CNN. In addition, compared with the general transformer, the swin transformer only calculates self-attention within the window. When facing the visual task of large images, the input can be divided into more windows to achieve linear computational complexity.

However, the computing resources required for self attention are still large, which also causes a large gap between the speed of the swin transformer and the CNN of the same level.



2.4.2 Instance segmentation

In order to more accurately optimize the pollination of honey bees, I intend to detect the area using images of pollen grains on honey bees to estimate the amount of pollen collected by honey bees. Because the pollen grain area needs to be accurately marked, it needs to be detected using the instance segmentation model, instead of object detection models, which simply select the bounding box.

2.4.2.1 Mask R-CNN

The model architecture of Mask R-CNN is shown in Fig. 2.13 (He et al., 2017). Mask R-CNN is mainly based on the architecture of Faster R-CNN, and adds a mask prediction branch, mask loss, and corrects the problem of mis-alignment between the Faster R-CNN feature map and the original image. Mask loss mainly calculates binary cross entropy for the category whose predicted category is the same as the ground truth category. Mask R-CNN uses RoiAlign to replace RoiPooling, so that the position of proposals can be represented by floating point numbers, which can more effectively integrate the input feature map and proposal information and extract the proposal feature map. Bounding box regression is to select the best bounding box for the object from the bounding box candidates.

Mask R-CNN's loss function is mainly divided into five types: RPN classification loss, RPN regression loss, mrcnn classification loss, mrcnn regression loss, and mask loss.

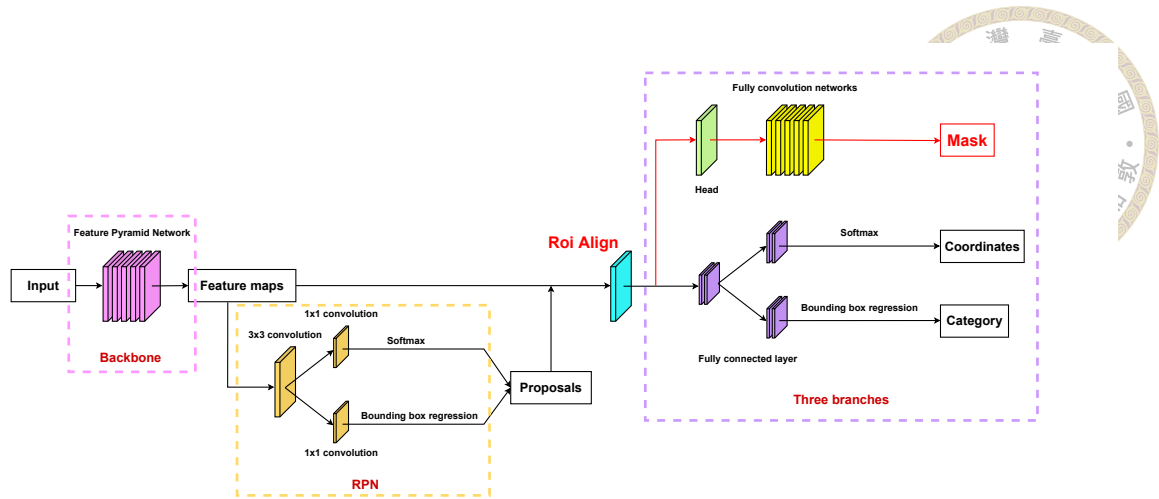


Figure 2.13: The architecture of the Mask R-CNN model, drawn according to (He et al., 2017).

The RPN classification loss is a cross entropy loss between the predicted class (x_i) of RPN and the ground truth class (y_i) in the batch size (N), as shown in Eq. (2.6):

$$\text{Cross Entropy Loss} = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(x_i) + (1 - y_i) \cdot \log(1 - x_i). \quad (2.6)$$

The RPN regression loss is a smooth L1 loss between the parameters of the predicted box of RPN (x_i) and the ground truth box (y_i) by regressing to offsets for the center of the default bounding box and for its width and height, as shown in Eq. (2.7):

$$\text{Smooth}_{L_1}(x_i, y_i) = \begin{cases} 0.5|x_i - y_i|^2 & \text{if } |x_i - y_i| < 1 \\ |x_i - y_i| - 0.5 & \text{otherwise} \end{cases}. \quad (2.7)$$

The classification loss and the regression loss of mrcnn follow the same format as Eq. (2.6) and Eq. (2.7), respectively. The mask loss is computed using the same formula as Eq. (2.2), which involves calculating BCEWithLogitsLoss between the predicted mask and the ground truth mask. N represents the batch size, y_i represents the ground truth mask, and x_i represents the predicted mask.

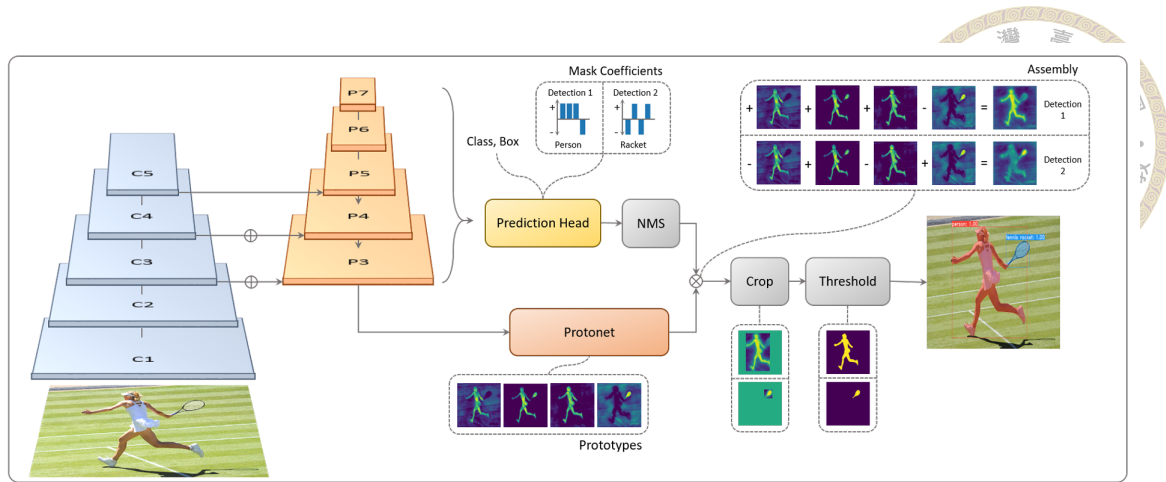


Figure 2.14: The architecture of the YOLACT model (Bolya et al., 2019, 2022).

2.4.2.2 YOLACT

YOLACT is one of the few instance segmentation models that can be applied in real-time at present. Despite its fast speed, it also comes with a price that its mAP is slightly lower than other two-stage models (Bolya et al., 2019, 2022). The model architecture of YOLACT is shown in Fig. 2.14. The YOLACT model first generates the prototype of the entire input image through the prototype constructed by multiple convolutional layers, and the prototype is similar to the feature map. In addition, anchor-based detectors are used to predict class, box, mask coefficients and pass Fast Non-Maximum Suppression (Fast NMS). There is information on how to combine prototypes in mask coefficients. Finally, the prototype and mask coefficients are directly multiplied by the matrix and combined with a sigmoid function as:

$$M = \sigma(PC^T), \quad (2.8)$$

where M is the assembled mask, σ is the sigmoid function, P represents the matrix of prototypes, C represents the matrix of mask coefficients, and T represents the transpose operation.

YOLACT employs four main types of loss functions: box regression loss, classifica-

tion loss, mask loss, and semantic loss. Let x_{ij}^p be an indicator for matching the i -th default box to the j -th ground truth box of category p . The box regression loss is a Smooth L1 loss between the predicted box (l) and the ground truth box (g) parameters by regressing to offsets for the center (c_x, c_y) of the default bounding box (d) and for its width (w) and height (h) as (Liu et al., 2016):

$$L_{box}(x, l, g) = \sum_{i \in Pos}^N \sum_{m \in c_x, c_y, w, h} x_{ij}^p \text{smooth}_{L1}(l_i^m - \hat{g}_j^m). \quad (2.9)$$

The classification loss is the softmax loss over multiple classes confidences (c) as (Liu et al., 2016):

$$L_{cls}(x, c) = - \sum_{i \in Pos}^N x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg}^N \log(\hat{c}_i^0), \quad (2.10)$$

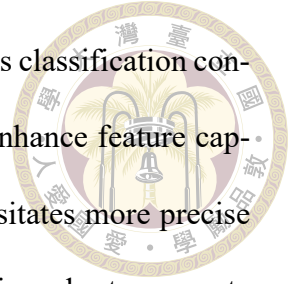
where $\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$.

The mask loss is the pixel-wise binary cross entropy between assembled masks M and the ground truth masks M_{gt} as:

$$L_{mask} = \text{BCE}(M, M_{gt}). \quad (2.11)$$

Based on the YOLACT architecture, YOLACT++ was proposed (Bolya et al., 2022). YOLACT++ increases the detection accuracy of the model without reducing the speed as much as possible. There are three main improvements: adding semantic segmentation Loss, adding fast mask re-scoring network, and changing to a deformable convolution layer. Semantic segmentation loss is applied to our feature space for layers that are only evaluated during training, which effectively increases feature richness with no speed penalty. Fast mask re-scoring network calculates the IoU of each object's prediction mask

and ground truth mask through an FCN, and then modifies the previous classification confidence. The addition of the deformable convolution layer aims to enhance feature capturing capabilities in YOLACT. This is essential as YOLACT necessitates more precise convolutional operation information compared to two-stage models in order to generate highly accurate prototypes.



2.4.3 Tracking algorithm

In the bee pollination optimization, because honey bees entering and leaving the hive were counted using videos, I tracked each honey bee through a tracking algorithm to avoid counting the same bee repeatedly in adjacent frames. There are two main categories of algorithms for Multi-Object Tracking (MOT): tracking-by-detection and joint-detection-association, with similar accuracy. Tracking-by-detection is based on the detection results of other models, and then uses the tracking algorithm to connect the adjacent frame relationships (features, positions) to achieve the tracking effect. The advantage is that the model dataset is easy to handle, but the disadvantage is that the processing speed is slower than joint-detection association (similar to two-stage detection and tracking). Joint-detection association directly uses a special model to detect objects and the relationship between adjacent frames. The advantage is that the processing speed is fast (one-stage direct detection and tracking), but the disadvantage is that the model dataset needs pre-processing to let the model know the relationship between adjacent frames. In the optimization of bee pollination, the object detection model employed for honey bee detection offers a combination of high accuracy and fast computation. Therefore, I primarily utilized the tracking-by-detection tracking algorithm to monitor the entry and exit of honey bees from the hive for bee counting purposes.

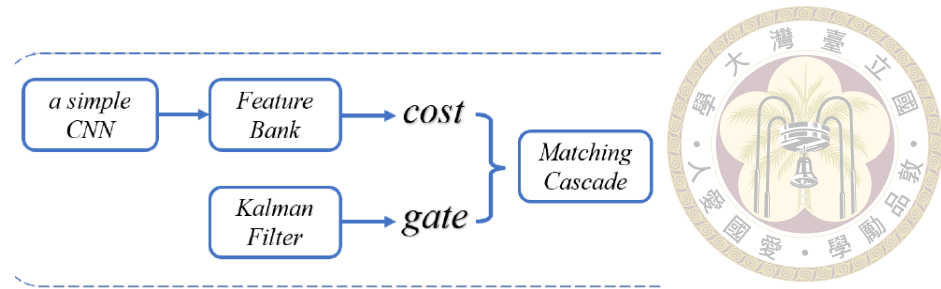


Figure 2.15: The flow chart of DeepSORT tracking algorithm (Wojke et al., 2017; Du et al., 2023).

2.4.3.1 DeepSORT

DeepSORT is a tracking-by-detection type multi-object tracking algorithm (Wojke et al., 2017; Du et al., 2023). The flow chart of DeepSORT is shown in Fig. 2.15. DeepSORT tracking algorithm is mainly divided into two branches: appearance and motion. The appearance branch first extracts the object features in the trajectory through a simple CNN, and puts the object features of the first 100 frames of the same trajectory into the feature bank. The cosine distance of the object features of the current frame and the object features in the feature bank was calculated as the cost. The motion branch uses the Kalman filter (Kalman, 1960) to predict the object's position in the trajectory in the current frame from the previous frame, and uses the Mahalanobis distance to measure the spatio-temporal dissimilarity between the predicted object in the trajectory and the actual object as a gate. Finally, through the matching cascade algorithm, the objects that frequently appear in the first few frames are prioritized, and the Hungarian algorithm (Kuhn, 1955) is combined with cost and gate to track each object in sequence.

2.4.3.2 StrongSORT

StrongSORT is a tracking-by-detection type multi-object tracking algorithm (Du et al., 2023), which is improved based on DeepSORT to obtain better tracking accuracy. The

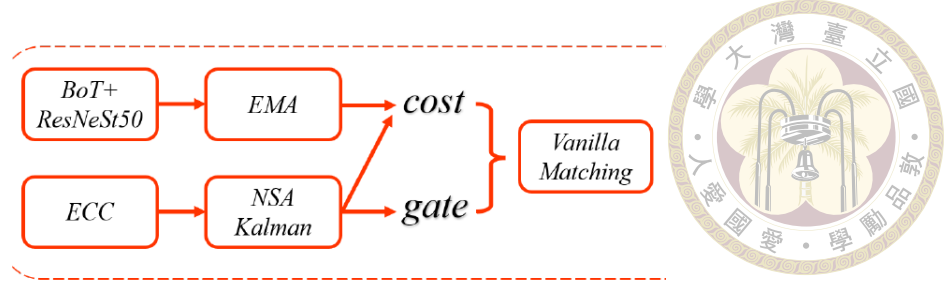


Figure 2.16: The flow chart of StrongSORT tracking algorithm (Du et al., 2023).

flow chart of StrongSORT is shown in Fig. 2.16. StrongSORT tracking algorithm is also divided into two branches: appearance and motion. In the appearance branch, the new feature extractor, BoT (Luo et al., 2019), replaces the past simple CNN and utilizes the ResNeSt50 backbone (Zhang et al., 2022) for object feature extraction in the trajectory. In addition, the feature bank has also been changed to filter object features through a feature updating strategy (Wang et al., 2020b). In the feature updating strategy, the cosine similarity between the object feature of the current frame and each track feature is calculated to confirm the trajectory category to which the object feature of the current frame belongs. Each trajectory feature is updated through Exponential Moving Average (EMA). The formula is as follows:

$$e_i^t = \alpha e_i^{t-1} + (1 - \alpha) f_i^t, \quad (2.12)$$

where e_i^t is the feature embedding of the i -th trajectory at time t (updated trajectory feature). e_i^{t-1} is the feature embedding of the i -th trajectory at time $t - 1$ (the feature of the trajectory before the update). f_i^t is the feature embedding (feature of the current object) of the object corresponding to the i -th trajectory at time t , and α is the proportionality constant (usually set to 0.9).

In the motion branch, an ECC image alignment algorithm (Evangelidis and Psarakis, 2008) is added for image alignment. In addition, the vanilla Kalman filter (Stadler and Beyerer, 2022) is replaced by the NSA Kalman filter (Du et al., 2021) to reduce the in-

fluence of noise and predict the position of the trajectory in the current frame, and then the Mahalanobis distance is calculated as the gate. Different from DeepSort, which only calculates the appearance branch information, the cost function of StrongSort combines both the appearance and motion branch information (Wang et al., 2020b). The formula is as follows:

$$C = \lambda A_a + (1 - \lambda) A_m, \quad (2.13)$$

where C is the total cost, λ is the weight factor (set to 0.98), A_a is the appearance cost (cosine similarity), and A_m is the motion cost (Mahalanobis distance). To avoid amplifying the priority of wrong object features, the final solution is performed by vanilla global linear assignment instead of the matching cascade algorithm. Finally, the Hungarian algorithm (Kuhn, 1955) is combined with cost and gate to track each object in sequence.

Based on the StrongSORT architecture, StrongSORT++ was proposed (Du et al., 2023). StrongSORT++ presents two lightweight tracking algorithms, Appearance-Free Link (AFLink) and Gaussian-smoothed interpolation (GSI), which do not require appearance branch information. AFLink simply uses time and position information for connectivity confidence calculations. The flow chart is shown in Fig. 2.17. First, two trajectories (tracklet in Fig. 2.17) are used as input, and the information of the first 30 frames is taken (if it is insufficient, 0 is added). Then, the features of different dimensions are extracted, squeezed, and pooled into the feature vector rich in time and position information. Ultimately, the multilayer perceptron (MLP) is used to determine connectivity confidence. Furthermore, GSI uses the interpolation algorithm with Gaussian process regression (Williams and Rasmussen, 1995) to fill in the blank frame identification results. Compared with linear interpolation (average of front and rear frames), GSI adds position information to make the results closer to ground truth without increasing the calculation

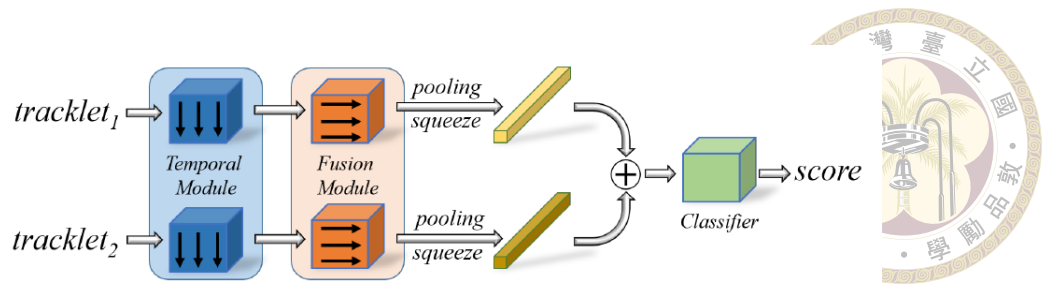


Figure 2.17: The flow chart of AFLink (Du et al., 2023).

time. The replacement of appearance branch information by AFLink and GSI proves beneficial in enhancing tracking performance due to their sophisticated operations.



Chapter 3 Materials & Methods

The following will be explained in two parts: monitoring of pests and optimization of bee pollination.

3.1 Monitoring of pests

This research aims to count the number of pests on sticky paper traps, and to measure the greenhouse environmental data through IoT sensors. Based on the correlation analysis results of the number of pests and environmental data, relevant indicators of pest control were established. The flow chart and schematic of pest monitoring are shown in Fig. 3.1 and Fig. 3.2, respectively.

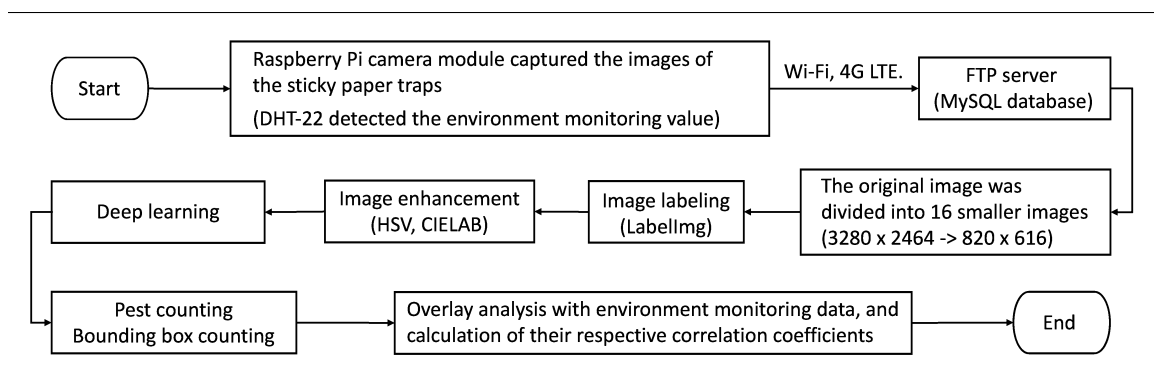


Figure 3.1: Flow chart of pest monitoring.

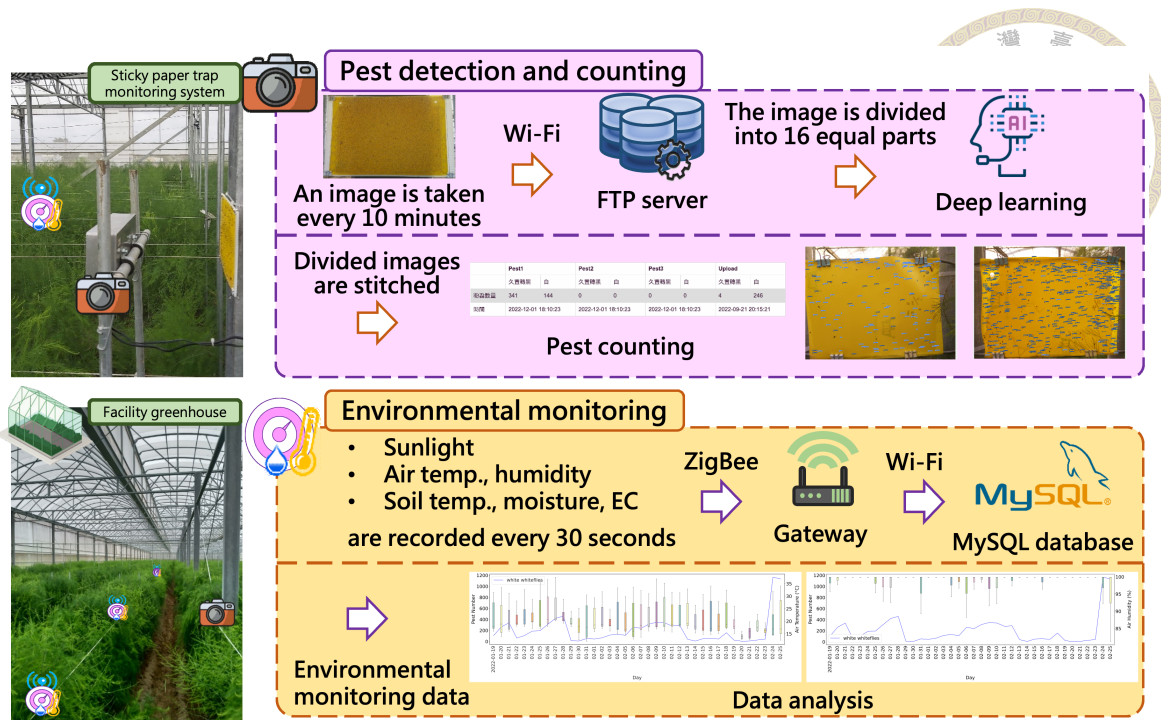


Figure 3.2: Schematic of pest monitoring.

3.1.1 IoT system

The location of the study, Yichu Branch Station at Tainan District Agricultural Research and Extension Station in Chiayi County of Taiwan, is depicted in Fig. 3.2. The field is enclosed with four-sided nets to prevent initial pest infestation. In order to establish an IoT system, environmental sensing nodes and sticky paper trap monitoring nodes were placed in the greenhouse. The network transmitted the data and images collected from these nodes back to our cloud database. Each environmental sensing node consisted of a gateway and a wireless temperature and humidity sensor. Six nodes were placed in different locations (A to F in Fig. 3.3) throughout the greenhouse. 7fb39b-87ca2c in Fig. 3.3 were the nodes' respective IDs. Air temperature and humidity within the greenhouse were recorded by the DHT-22 sensor every 30 seconds. The gateway, integrated with a Raspberry Pi 3B, initiated the wireless sensors and confirmed the transmission paths by sending a command packet. Subsequently, the gateway commanded the wireless sensors to gather

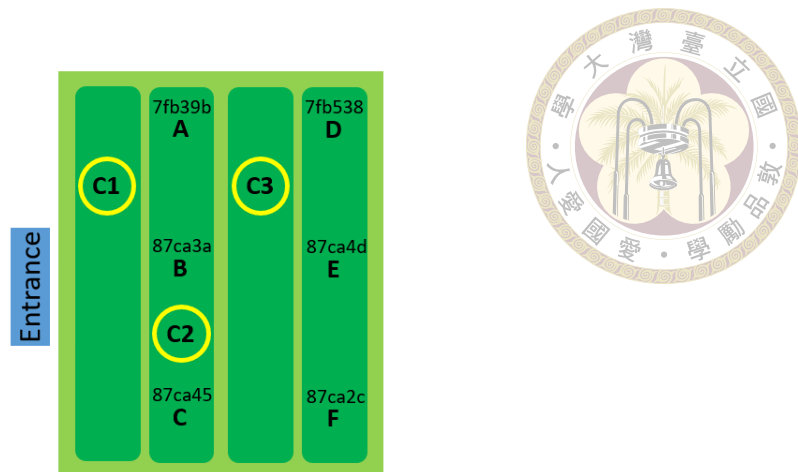


Figure 3.3: Locations of greenhouse nodes.

sensing data and transmit it back. Finally, the collected environmental sensing data was transmitted through Wi-Fi or 4G LTE. to the MySQL database.

The greenhouse was equipped with sticky paper trap monitoring nodes, consisting of a Raspberry Pi camera module and a yellow sticky paper trap placed at a height of 1 meter above the soil to capture pests in flight (as illustrated in Fig. 3.4). The Raspberry Pi 3B controlled the module, which had a static image resolution of up to 3280×2464 . Three nodes were placed at different positions (C1 to C3 in Fig. 3.3) to capture images of the sticky paper traps every 10 minutes, starting from 6:00 am until 8:00 pm. The Raspberry Pi 3B is a single-board computer that features a central processing unit (CPU), graphics processing unit (GPU), and system-on-chip (SoC). It also includes GPIO pins and a USB hub for enhanced connectivity and expandability. To optimize the efficacy of sticky paper traps in capturing small pests like silverleaf whitefly and thrips, the selection of an appropriate light wavelength played a crucial role (Fig. 3.5). Previous studies have demonstrated that silverleaf whitefly was highly attracted to optical wavelengths within the range of 500 nm to 600 nm (Kim et al., 2012). Based on this finding, the study specifically opted for IATP sticky paper trap with a light wavelength of around 600 nm. Similarly, previous studies have demonstrated that thrips exhibit a strong attraction to optical wavelengths between

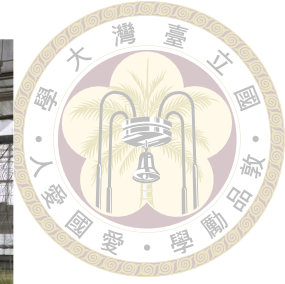
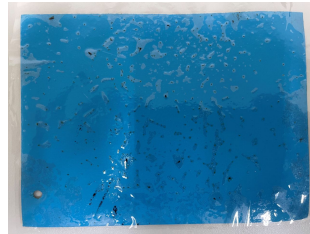


Figure 3.4: The sticky paper trap monitoring node.



(a) Whiteflies



(b) Thrips



(c) Thrips

Figure 3.5: Original images of sticky pest traps for (a) whiteflies and (b)-(c) thrips.

420 nm and 470 nm (Lopez-Reyes et al., 2022). Consequently, the study selected IATPBL and FATP sticky paper traps with light wavelengths of around 450 nm. The experimental results of the adhesive trap sample are presented in Figures 3.6-3.8, illustrating the “relative sensitivity” of the sticky paper trap to different light wavelengths. The dimensions of the sticky paper traps were 15 cm × 11 cm, manufactured by KK Enterprise Co., Ltd. The images of sticky paper traps were transmitted through Wi-Fi or 4G LTE. to the FTP server for further analysis.

3.1.2 Image preprocessing

In Fig. 3.5, whiteflies and thrips are pests commonly found in areas where asparagus is grown. When whiteflies get caught in sticky paper traps, their appearance changes over

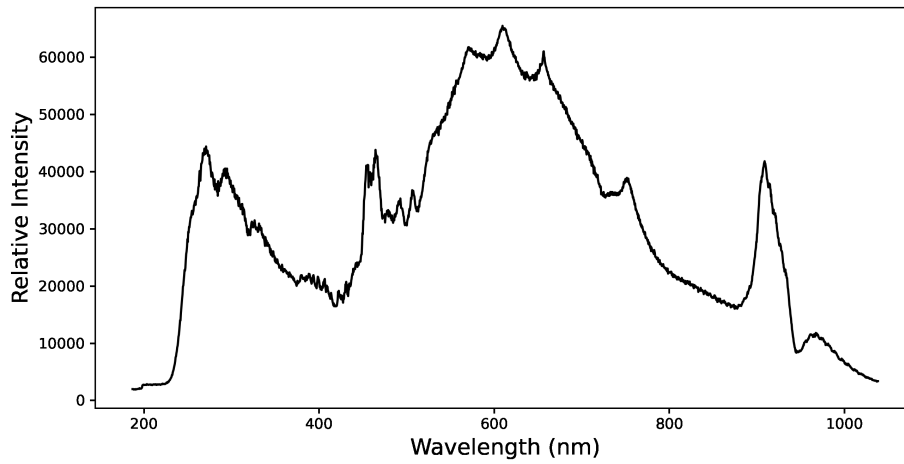


Figure 3.6: Optical wavelength test results of yellow sticky paper samples (IATP).

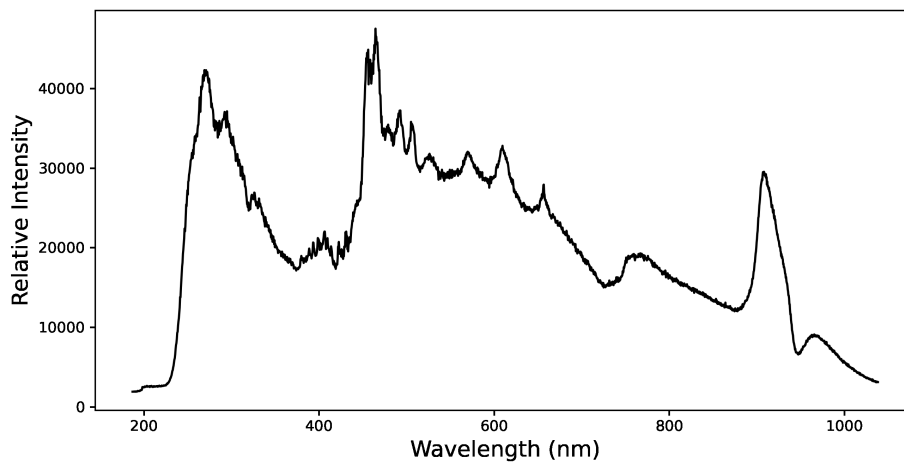


Figure 3.7: Optical wavelength test results of blue sticky paper samples (IATPBL).

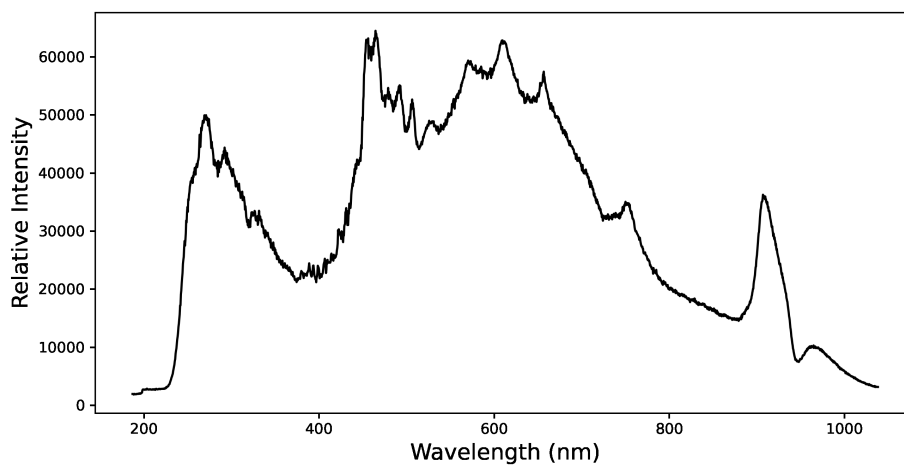


Figure 3.8: Optical wavelength test results of white sticky paper samples (FATP).

time. Initially, they are white and stay that way for around two to three weeks. After three weeks, they gradually turn gray and then eventually darken after about four weeks. To avoid confusion in the model, whiteflies that were recently caught on sticky paper traps were classified as white whiteflies (WWF), while those that stayed on the traps for a few weeks were classified as black whiteflies (BWF). The experimental asparagus greenhouse had no thrips present, but images of thrips on sticky paper traps from other greenhouses provided by Yichu Branch Station were also utilized to make pest detection more versatile. An extra class was allocated for thrips, making the models capable of detecting three categories: white whitefly, black whitefly, and thrips.

In Fig. 3.9(a), the sticky paper trap monitoring node captured images with dimensions of 3280×2464 . However, the pests' small size in the image posed a challenge in manually labeling them and extracting relevant features for model training. To address this issue, the original image was cropped into 16 smaller images, each having a resolution of 820×616 , and subsequently labeled using Labelling software (Tzutalin, 2015).

Our objective is to accurately count pests, so image enhancement techniques such as rotation and inversion have minimal impact on improving accuracy. Instead, the primary challenge is changes in sunlight brightness. To address this challenge, two methods were used. Firstly, The hue (H) in the HSV color space was modified to replicate color differences caused by varying lighting conditions. This modification changed the color of the sticky paper to an almost orange hue, as seen in Fig. 3.9(b). Secondly, the average brightness (L^*) of all images was computed in the CIELAB color space and mapped onto all images to ensure consistent brightness data, thereby enhancing the model's consistency during training. The CIELAB processing resulted in a darkening of the original image, as illustrated in Fig. 3.9(c).

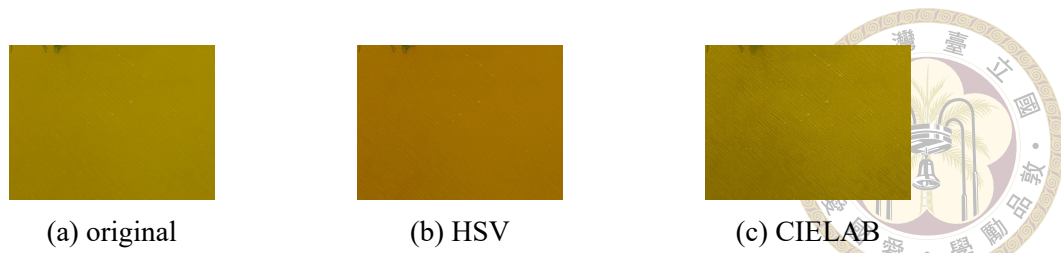


Figure 3.9: (a) Original sticky pest trap image after division. (b) Image after HSV processing. (c) Image after CIELAB processing.

3.1.3 Model training

The main goal of pest monitoring was to determine the number of pests, so it was more feasible to train a deep learning model for precise pest detection and counting rather than accurate pest segmentation. Therefore, instead of using an instance segmentation model, an object detection model was selected for pest counting, which utilizes bounding boxes to detect the position of pests and provide an accurate count of their numbers. In this study, various models, including YOLOv5 models with backbones “x” and “x6” (Jocher, 2020), the YOLOR model (Wang et al., 2021), the YOLOv7 model (Wang et al., 2023), the YOLOv8 model (Jocher et al., 2023), the YOLOv6 model (Li et al., 2023), the Faster R-CNN model (Ren et al., 2015), and the RetinaNet model with Swin-T backbone (Liu et al., 2021), were trained to identify white whiteflies, black whiteflies, and thrips.

After manually eliminating out images of sticky paper traps obstructed by crops, a total of 1151 images were chosen for further analysis. Of those, 111 images of whitefly underwent image augmentation of HSV and CIELAB, resulting in an extra 222 images. There were 1040 images left, with 560 images depicting whiteflies and 480 images depicting thrips. A dataset of 1373 images, with dimensions of 820×616 , was used to train the models. The dataset included 893 images of whitefly and 480 images of thrips, which were split into training and validation sets, with a ratio of 80% and 20%, respectively. The training set consisted of 714 images of whitefly and 384 images of thrips, while the

validation set consisted of 179 images of whitefly and 96 images of thrips. To prevent any duplication between the original and enhanced images, 333 images of whitefly were exclusively included in the training set.



Parameters for model training are listed in Table 3.1, but there are variations in the input size, batch size, and training epochs between the models. As each model had a distinct architectural design, the corresponding input size required fine-tuning. For instance, the YOLO series model mandated a multiple of 32 for compatibility. By finishing fine-tuning the input size, the optimal dimensions for each model were selected to facilitate accurate bounding box localization, considering the initial goal of a sticky paper trap image size of 820×616 . The Faster R-CNN and RetinaNet models, for instance, were determined with an input size of 820×616 . In addition, the choice of batch size was influenced by the need to optimize GPU memory utilization for efficient calculations while ensuring consistent resource allocation across models. Hence, the batch size varied with the number of parameters of each model. In terms of training epochs, the objective was to ensure that each model reaches convergence, guaranteeing the presence of the most favorable results for comparison. Consequently, the training process continued until confirmation of convergence was obtained, resulting in varying values for the number of training epochs across the models.

In this study, the YOLOv5, YOLOR, YOLOv7, YOLOv8, and YOLOv6 models required input sizes that were multiples of 32 to match the kernel and stride of the models. Therefore, the dimensions were chosen to be 704×704 . The process of resizing the images involved first downsampling their dimensions to approximately 704×529 , then padding to the shorter side to achieve a final size of 704×704 . The input size for the Faster R-CNN and RetinaNet models was kept the same as the original image size. In addition, the

learning rates were set to increase linearly for 1,000 iterations in the YOLOv5, YOLOR, and YOLOv7 models, and for 100 iterations in the YOLOv8 model, before being decayed by the one-cycle learning rate policy (Smith and Topin, 2019). The learning rates in the YOLOv6, Faster R-CNN, and RetinaNet were warmed up linearly for 1,000 iterations before being decayed using the cosine annealing method (Loshchilov and Hutter, 2016). The training was conducted using Python 3.7.10 and PyTorch 1.7.1 on a CentOS Linux 7 (Core) system with Xeon(R) Silver 4110 for CPU, NVIDIA GeForce RTX 3090 for GPU, and CUDA 11.2 for Computation Unified DEVICE Architecture.

Table 3.1: Training parameters of the models.

Model	Backbone	Input size	Batch size	Learning rate	Training epoch
YOLOv5	YOLOv5x	704x704	24	0.001	500
YOLOv5	YOLOv5x6	704x704	24	0.001	500
YOLOR	YOLOR-D6	704x704	24	0.001	500
YOLOv6	YOLOv6-L6	704x704	24	0.001	500
YOLOv7	YOLOv7x	704x704	24	0.001	500
YOLOv8	YOLOv8x6	704x704	24	0.001	500
Faster R-CNN	ResNet101	820x616	12	0.001	1500
RetinaNet	Swin-T	820x616	12	0.001	1500

3.1.4 Combining environmental monitoring and pest counting

This study aims to establish a correlation between environmental changes and pest infestation using AIoT data. The pest populations, measured from January 19, 2022, to February 25, 2022, along with the environmental temperature and humidity data collected using IoT technology, were analyzed for daily correlations. However, variations in environmental sensing data may not necessarily mirror the pest populations on that particular day. Hence, correlation coefficients were computed between the daily change in pest populations and sensing data from the past three days, two days, and one day. Given that whiteflies were the primary pests in the field of asparagus, the count of whiteflies was

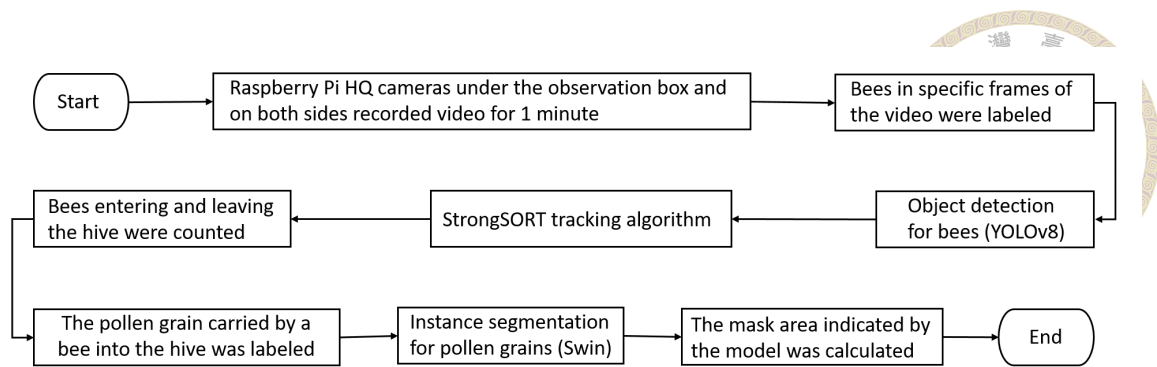


Figure 3.10: Flow chart of bee pollination optimization.

employed as the pest population.

3.2 Optimization of bee pollination

This study aimed to count the number of bees entering and leaving the hive and estimate the amount of pollen carried by bees. Pollen grains could be estimated based on the pollen image pixels captured by the camera. Ultimately, this study combined bee entry and exit counts and pollen grain detection to enhance bee pollination. The flow chart and schematic of bee pollination optimization are shown in Fig. 3.10 and Fig. 3.11, respectively.

3.2.1 Image acquisition system

The honey bee monitoring system based on IoT technology was regularly monitored in the greenhouse of the oriental melon (*Cucumis melo Linn.*) depicted in Fig. 3.11. The system mainly consisted of an acrylic observation box, three Raspberry Pi HQ cameras with a video resolution of 1920×1080 and 25 frames per second, an LED light bar, and a relay module. The observation box was installed at the entrance of the beehive, and the honey bees must pass through the observation aisle to enter or exit the beehive. Three

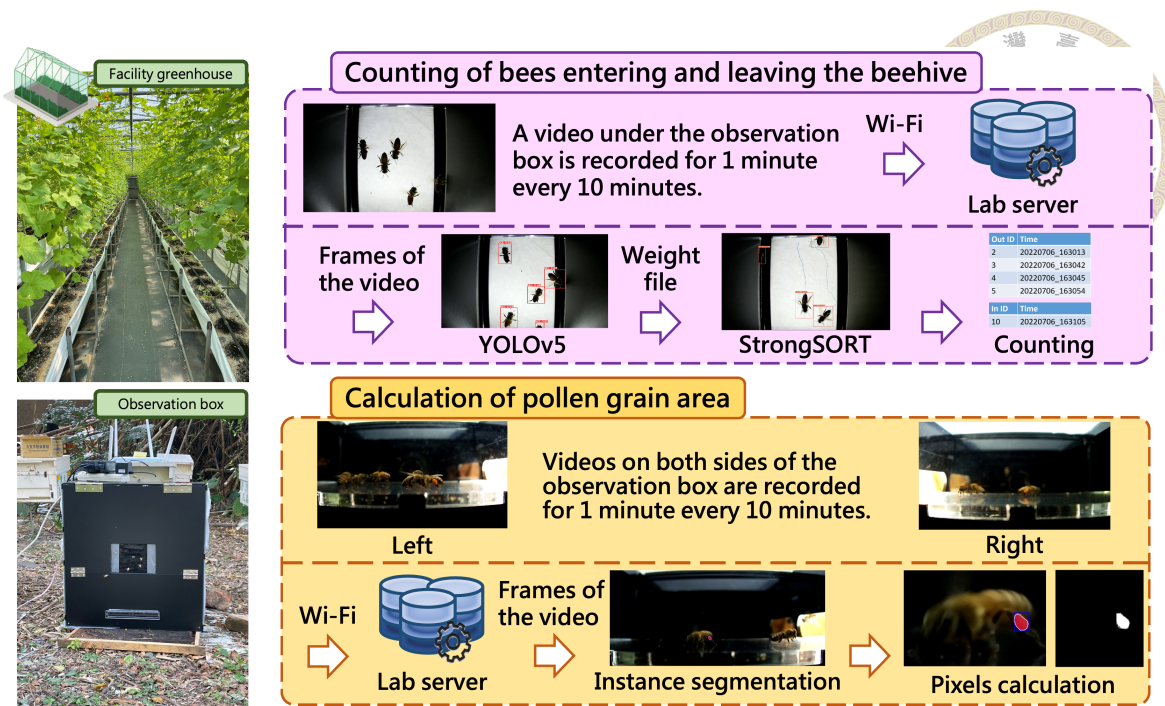
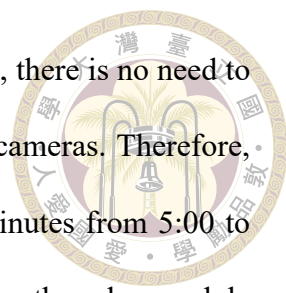


Figure 3.11: Schematic of bee pollination optimization.

cameras controlled by corresponding Raspberry Pi 3B were installed under, to the left, and to the right of the observation aisle to record the videos of the bee entering and leaving the hive. The SMD 5730 white LED light bar, manufactured by JIN HUA ELECTRONIC Co., Ltd., was mounted at a height of 20cm above the observation aisle. The relay module was installed under the observation aisle. Please note that there were two versions of the observation box, as it was undergoing modifications throughout the study. This can be observed in Fig. 3.12. The second iteration of the observation box boasted several upgrades that provided a better observation experience and enhanced its ability to resist water. In the observation aisle part, four mechanisms could be replaced to accommodate different situations, as highlighted in Fig. 3.13. Furthermore, the waterproof part of the box incorporated most of the circuits found in the junction box in Fig. 3.12(a), and the opening method of the observation box had been altered to prevent water droplets from seeping into the box from the top down.

Because this study only aims to find the correlation between the area and weight of

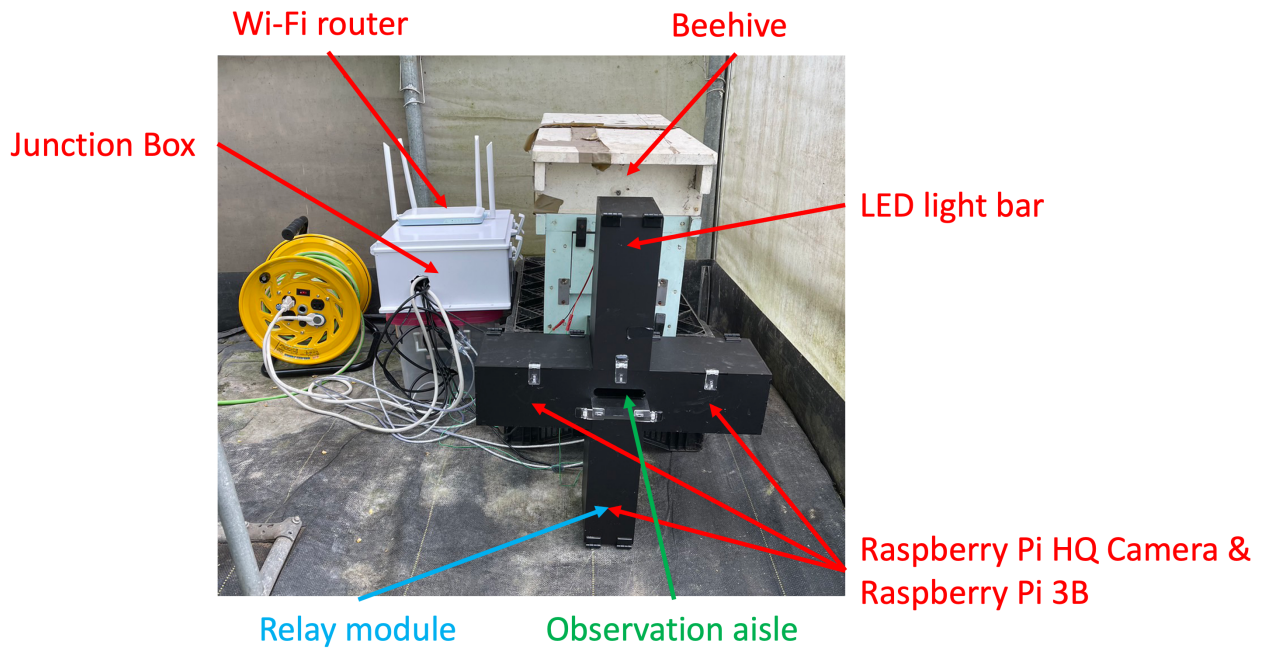


pollen collected by bees rather than recording complete bee activities, there is no need to implement continuous recording, which tends to result in overheating cameras. Therefore, the system was set to record videos for the first minute every 10 minutes from 5:00 to 19:00 every day. At the same time, when the cameras were turned on, the relay module controlled by the Raspberry Pi 3B turned on the LED light bar. The videos were saved to a hard disk and sent back to the server simultaneously through Wi-Fi or 4G LTE. Each video was named according to the time that the camera began to record.

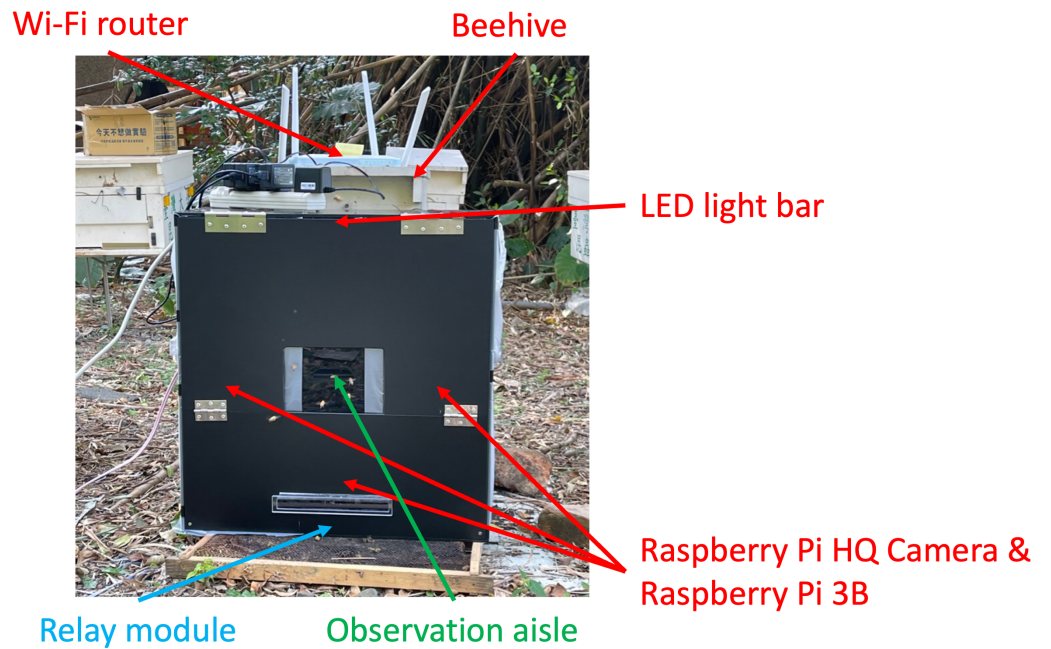
3.2.2 Image preprocessing

Crop pollination is largely dependent on the effectiveness of honey bees, and the amount of pollen they carry serves as a crucial indicator for evaluating pollination efficiency (Bernauer et al., 2022). To accurately calculate the pollen grain area, it is pivotal to train a deep learning model that not only can identify and count pollen grains but also precisely segment each one. Thus, an instance segmentation model was opted for detecting pollen grains rather than using an object detection model. In addition, the current instance segmentation model in real-time applications still consumes a lot of computing resources and the accuracy is slightly lower, so I first used the object detection model to detect honey bees, and then used the tracking algorithm to calculate the number of honey bees entering and leaving the hive, and finally used the instance segmentation model to segment pollen grains on the images of honey bees entering the hive.

In the object detection part, although the camera images under the observation aisle were in poor lighting, the color of honey bees can be ignored in the bee detection. The dataset without color features provided better accuracy and was suitable for tracking. Hence, I chose the images under the observation aisle, i.e., Fig. 3.14(a) and Fig. 3.14(d),



(a) Version 1



(b) Version 2

Figure 3.12: The honey bee monitoring system.

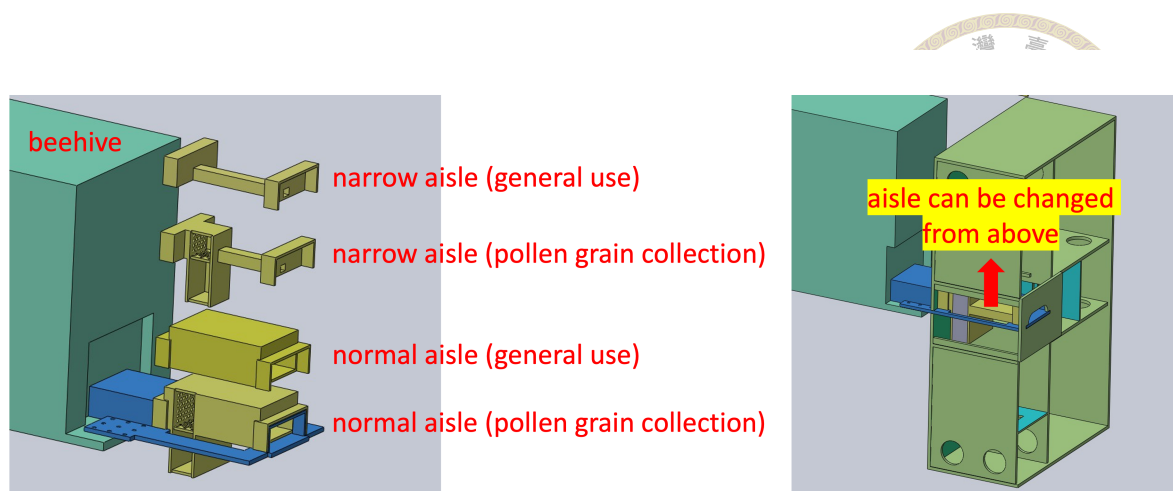


Figure 3.13: Mechanisms for observation box version 2.

as the dataset. Furthermore, I manually filtered the recorded videos and took screenshots of the number of frames where honey bees appeared. There was only one category of honey bees (bee). Bounding box labeling is performed using LabelImg software (Tzutalin, 2015).

In the instance segmentation part, because the pollen grain detection needs to consider the pollen color to obtain better accuracy, I chose the camera images on both sides of the observation aisle as the dataset, as shown in Fig. 3.14(b)-(c) and Fig. 3.14(e)-(f). I manually filtered the recorded videos and took screenshots of the number of frames where pollen grains appeared. There was only one category of pollen grains (pollen). Mask labeling was performed using LabelMe software (Russell et al., 2008).

3.2.3 Counting of honey bees entering and leaving the hive

3.2.3.1 Object detection model training

The object detection model employs bounding boxes to indicate the precise position of honey bees, and the tracking algorithm tracks the honey bees to count the number of

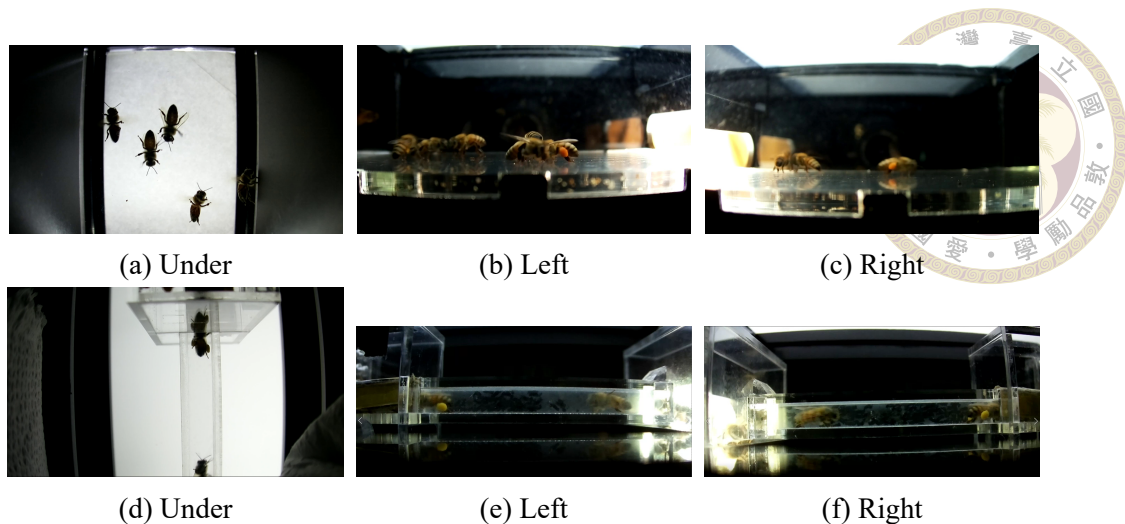


Figure 3.14: Images (a)-(c) show the observation aisle in version 1 of the observation box. Images (d)-(f) depict the observation aisle in version 2 of the observation box. The images are positioned below, to the left of, and to the right of the observation aisle.

honey bees entering and leaving the hive. Since the honey bee detection was relatively simple, the models with simpler structures and faster inference speed were trained to detect honey bees. In this study, because the high compatibility of YOLO models with tracking algorithms, the YOLOv3 model with backbone “tiny” (Redmon and Farhadi, 2018), the YOLOv5 model with backbones “n6” and “s6” (Jocher, 2020), the YOLOv7 model with backbone “tiny” (Wang et al., 2023), and the YOLOv8 model with backbones “n” and “s” (Jocher et al., 2023) were trained to detect honey bees.

After the manual screening, the model was trained with 686 honey bee images captured from the video under the observation aisle. The dataset was split 4:1 into training and validation sets, with 549 images in the training set and 137 images in the validation set. The image resolution was 1920×1080 , the same as the video captured by the honey bee monitoring system. Considering the minimal variations in lighting and color among the images captured under the observation channel, including screenshots from the same video in both the training and validation sets simultaneously does not introduce any opportunity for the model to gain an unfair advantage. Therefore, in this task, there was no

specific avoidance of allocating images from the same video screenshot to the training and validation sets simultaneously. Table 3.2 displays the training parameters. These parameters were adjusted to fully utilize GPU memory and maximize computing resources.

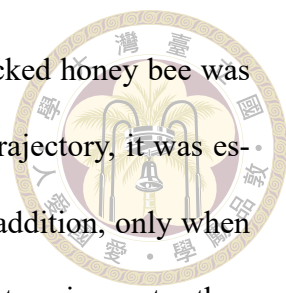
Table 3.2: Training parameters of the models used by the tracking algorithm.

Model	Backbone	Input size	Batch size	Learning rate	Training epoch
YOLOv3	YOLOv3-tiny	704x704	24	0.001	500
YOLOv5	YOLOv5n6	704x704	24	0.001	500
YOLOv5	YOLOv5s6	704x704	24	0.001	500
YOLOv7	YOLOv7-tiny	704x704	24	0.001	500
YOLOv8	YOLOv8n	704x704	24	0.001	500
YOLOv8	YOLOv8s	704x704	24	0.001	500

In this study, the YOLOv3, YOLOv5, YOLOv7, and YOLOv8 models required input sizes that were multiples of 32 to match the kernel and stride of the models. Therefore, the dimensions were chosen to 704×704 . The process of resizing the images involved first downsampling their dimensions to approximately 704×529 , then padding to the shorter side to achieve a final size of 704×704 . In addition, the learning rates were set to increase linearly for 1,000 iterations in the YOLOv3, YOLOv5, and YOLOv7 models, and for 100 iterations in the YOLOv8 model, before being decayed by the one-cycle learning rate policy (Smith and Topin, 2019). The training was conducted using Python 3.8.15 and PyTorch 1.10.0 on a CentOS Linux 7 (Core) system with Xeon(R) Silver 4110 for CPU, NVIDIA GeForce RTX 3090 for GPU, and CUDA 11.2 for Computation Unified DEVICE Architecture.

3.2.3.2 Tracking algorithm

To avoid double counting, I concatenated the YOLO detection results to the the DeepSORT (Wojke et al., 2017) and StrongSORT (Du et al., 2023) tracking algorithm. The counting rules for entering and leaving the hive was set by tracking the starting point,



ending point, and length of the trajectory. The trajectory of each tracked honey bee was drawn, and then by setting the starting and ending positions of the trajectory, it was estimated whether the honey bee was entering or leaving the hive. In addition, only when the Euclidean distance between the start and end points of the trajectory is greater than 480 pixels, will the trajectory be counted to avoid misjudgment. Through the shooting time recorded by the honey bee monitoring system (video name) and the frame recorded by DeepSORT and StrongSORT, the duration of the honey bee appearance was obtained. Based on the above honey bee detection, tracking, and counting, the entry and exit time of each honey bee in the video could be accurately found.

3.2.4 Calculation of pollen grain area

3.2.4.1 Instance segmentation models training

To find out the pollen grain area, an instance segmentation model was used for detection. Because the images under the observation box faced the light source, the color of the honey bees was all black and lacked color features. Therefore, the images on both sides of the observation box were used for pollen grain detection. In this study, the YOLACT model (Bolya et al., 2019, 2022), YOLOv5 model (Jocher, 2020), YOLOv7 model (Wang et al., 2023), YOLOv8 model (Jocher et al., 2023), Mask R-CNN model with ResNet50 backbone (He et al., 2017), and the Mask R-CNN model with Swin-T backbone (Liu et al., 2021) were trained to segment pollen grains carried by bees.

There were 1284 honey bee images on both sides of the observation box in the total dataset. The dataset was split in the ratio of 4:1 into training and validation sets, with 1025 images in the training set and 259 images in the validation set. All images contained honey

bees carrying pollen grains. The image resolution was also 1920×1080 , the same as the video captured by the honey bee monitoring system. Furthermore, to avoid misjudging the model result, images from the same video screenshot would not be allocated to the training set and validation simultaneously.

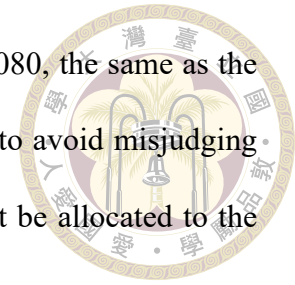


Table 3.3 lists the training parameters, but there are variations in the input size, batch size, and training epochs between the models. As each model had a distinct architectural design, the corresponding input size required fine-tuning. For instance, the YOLACT model necessitated a multiple of 550, whereas the YOLO series model mandated a multiple of 32 for compatibility. By finishing fine-tuning the input size, the optimal dimensions for each model were selected to facilitate accurate bounding box localization and mask generation, considering the initial goal of an approximate image size of 1000×1000 . Through rigorous testing, the Mask R-CNN model, for instance, was determined to perform optimally with an input size of 1333×800 . In addition, the choice of batch size was influenced by the need to optimize GPU memory utilization for efficient calculations while ensuring consistent resource allocation across models. Hence, the batch size varied with the number of parameters of each model. In terms of training epochs, the objective was to ensure that each model reaches convergence, guaranteeing the presence of the most favorable results for comparison. Consequently, the training process continued until confirmation of convergence was obtained, resulting in varying values for the number of training epochs across the models.

The YOLACT model requires the input width and height to be the same due to its design. Therefore, in this study, after balancing detection accuracy and computation time, the input image dimensions were set to 1100×1100 pixels. The image was downsampled using bilinear interpolation. On the other hand, the YOLOv5, YOLOv7, and YOLOv8

Table 3.3: Training parameters of the instance segmentation models.

Model	Backbone	Input size	Batch size	Learning rate	Training epoch
YOLACT	ResNet50	1100x1100	16	0.001	1994
YOLOv5	YOLOv5x-seg	960x960	24	0.001	500
YOLOv7	YOLOv7-seg	960x960	24	0.001	500
YOLOv8	YOLOv8x-seg	960x960	12	0.001	500
Mask R-CNN	ResNet50	1333x800	12	0.001	500
Mask R-CNN	Swin-T	1333x800	6	0.001	500

models required input sizes to be multiples of 32 to match the kernel and stride of the models. Therefore, the dimensions were chosen to 960×960 . The process of resizing the images involved first downsampling their dimensions to approximately 960×540 , then padding to the shorter side to achieve a final size of 960×960 . For the Mask R-CNN, the image resizing process initially chose the smaller ratio between the input size and the image size ($1333/1920$ and $800/1080$) and downsampled the image to approximately 1333×750 pixels according to this ratio. Since the model required input sizes that were multiples of 32 to match the kernel and stride of the models, the image was padded to the final dimensions of 1344×768 pixels. Additionally, the YOLACT and Mask R-CNN models' learning rates were first linearly warmed up for 1000 iterations and then decayed using the step learning rate scheduler. The YOLOv5 model's learning rates were initially linearly warmed up for 1000 iterations and then decayed using the one-cycle learning rate policy (Smith and Topin, 2019). In contrast, the YOLOv7 and YOLOv8 models' learning rates were first linearly warmed up for 100 iterations and decayed using the one-cycle learning rate policy (Smith and Topin, 2019). The training was conducted using Python 3.7.9 and PyTorch 1.10.1 on a CentOS Linux 7 (Core) system with Xeon(R) Silver 4110 for CPU, NVIDIA GeForce RTX 3090 for GPU, and CUDA 11.2 for Computation Unified DEVICE Architecture.

3.2.4.2 Algorithm of pollen grain area calculation



Based on the count results of honey bees entering and leaving the hive, the time when each entering honey bee appeared in the image under the observation box was recorded. However, because the slight difference in the shooting time of the three cameras existed, it could result in the image of the pollen grains carried by honey bees not being captured. To ensure consistency in the data, two steps were taken. First, the footage from the three cameras was carefully aligned. This alignment process ensured that the captured images were synchronized and properly matched. Second, to capture comprehensive information about each honey bee, the videos of the cameras on both sides were configured to screenshot five images for each bee. This allowed for a more detailed and complete representation of the bees' activities. The videos were captured with 25 frames per second. The frame at which the bee first appeared in the video was marked as a zero frame. The 13th and 25th frames before and the 13th and 25th frames after that were extracted from the video clips of the cameras on both sides.

To find the pollen grain area, after the instance segmentation model had detected the area of pollen grains, I calculated the area of the masked image through image processing. To avoid calculating the area of the non-mask area, through the HSV color space, the bounding box area was found from the segmented image, and the bounding box area was cropped. Then the mask area was found from the bounding box area. Finally, the number of pixels contained in the mask area was calculated to obtain the pollen grain area. For the same bee, I calculated the pollen grain area for a zero frame, the 13th and 25th frames before, and the 13th and 25th frames after, respectively. To ensure the optimum pollen area was found, I took the maximum value as the representative of the bee's pollen area.

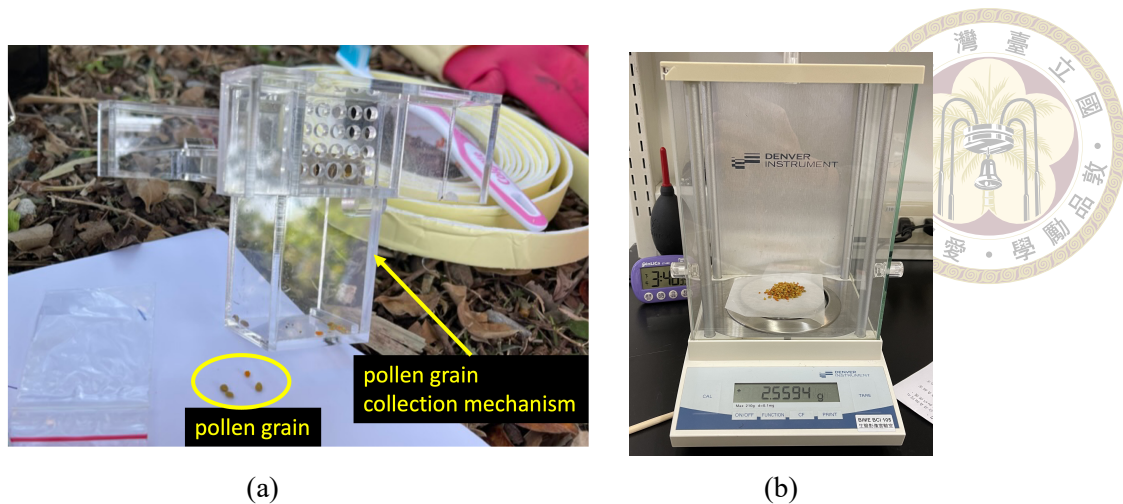


Figure 3.15: (a) The pollen grains scraped by the mechanism. (b) The electronic scale (TP-214).

3.2.5 Correlation analysis between calculated area and actual weight of pollen grains

The calculation result of the pollen area algorithm with the actual pollen grain weight was analyzed to ensure that the pollen area output by the system could replace the actual pollen weight to evaluate the actual pollination status of crops. First, version 2 of the observation box was used with a narrow aisle for pollen grain collection (Fig. 3.15(a)) that could automatically scrape the pollen grains from the honey bees. Then, according to the operation of the honey bee monitoring system, the pollen grains scraped by the mechanism in the first minute of every 10 minutes were collected in sequence and weighed by the TP-214 electronic scale (Fig. 3.15(b)). The DENVER INSTRUMENT TP-214 electronic scale has a measurement accuracy of 0.1 mg. Finally, the correlation analysis of the pollen grain area calculation result of the recorded video and the actual pollen grain weight in the corresponding time interval was carried out. It could be observed whether the area of pollen grains carried by the honey bees actually reflected the weight of pollen grains collected from the honey bees. The validity of the estimation method can ensure that the pollen area could be used as an indicator for enhancing the pollination of honey bees.



Chapter 4 Results & Discussion

This chapter discusses two main aspects: pest monitoring and optimization of bee pollination. In the first part, we dive into the methods and techniques used for monitoring pests, with the aim of enhancing the efficiency of pest control measures. The second part focuses on the optimization of bee pollination, exploring strategies to maximize pollination effectiveness and crop yield. These two critical aspects play a crucial role in sustainable agriculture and the overall success of crop cultivation in various settings.

4.1 Monitoring of pests

4.1.1 Model comparison

To determine the optimal solution for greenhouse pest counting, three different types of models were evaluated for their pest detection accuracy. These models included the one-stage model YOLO (Jocher, 2020; Wang et al., 2021, 2023; Jocher et al., 2023; Li et al., 2023), the two-stage model Faster R-CNN (Ren et al., 2015), and the transformer series model RetinaNet with Swin-T backbone (Liu et al., 2021). For conducting a comparison between the models, the training and validation sets of pests consisted of 1,098 and 275 pest images, respectively. Table 3.1 displays the training parameters. The model test-

ing results, with an Intersection over Union (IoU) threshold of 0.5, are shown in Table 4.1. Please note that the precision and recall were not calculated during the model inference of YOLOv6, Faster R-CNN, and RetinaNet. Among the YOLO series of one-stage models, YOLOv5 with backbone “x6” achieved the highest precision of 0.938, recall of 0.919, and mean average precision (mAP) of 0.953, which were superior to the latest YOLOv8 model. Despite the official claim of YOLOv6 surpassing YOLOv7 and YOLOv8 in detection accuracy, it fell short when applied to pest detection tasks, exhibiting the lowest mAP among the YOLO series models. Furthermore, YOLOv5 with backbone “x6” was also compared to other models. It was found that both YOLOv5 with backbone “x6” and RetinaNet with transformer family Swin-T backbone achieved higher mAP. However, YOLOv5 model with “x6” backbone reached faster convergence than the RetinaNet model with Swin-T backbone, as shown in Table 3.1. Hence, YOLOv5 with backbone “x6” was chosen as the optimal solution for field pest counting.

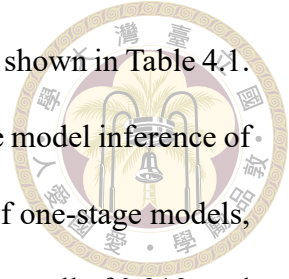


Table 4.1: Comparison of the precision, recall, and mAP of the models.

Model	Backbone	Precision	Recall	mAP
YOLOv5	YOLOv5x	0.875	0.842	0.880
YOLOv5	YOLOv5x6	0.938	0.919	0.953
YOLOR	YOLOR-D6	0.741	0.871	0.862
YOLOv7	YOLOv7x	0.901	0.867	0.915
YOLOv8	YOLOv8x6	0.867	0.846	0.896
YOLOv6	YOLOv6-L6	-	-	0.806
Faster R-CNN	ResNet101	-	-	0.774
RetinaNet	Swin-T	-	-	0.946

4.1.2 Results obtained by YOLOv5 model with “x6” backbone.

Typically, different colored sticky paper traps are utilized to attract whiteflies and thrips. However, as the experimental asparagus greenhouse had no thrips present, images of thrips captured on sticky paper traps from another greenhouse were randomly inserted



Figure 4.1: Results of pest detection with YOLOv5 model using “x6” backbone. The thrips sticky paper trap images are represented by blue and white rectangles.

into the whitefly images as a testing set. The testing set consisting of images of whiteflies and thrips was used to assess the efficacy of the YOLOv5 model with backbone “x6”, which demonstrated the highest mAP, in detecting and locating these pests. Fig. 4.1 illustrates the outcomes of the YOLOv5 model with backbone “x6”, where the bounding boxes accurately display the pest’ positions. The model successfully detected the majority of pests on the images of sticky paper traps. Please note that the count of bounding boxes generated by the model was also compared to the count of manually labeled pests.

The YOLOv5 model with backbone “x6” was evaluated based on its precision, recall, and F1 score for detecting black and white whiteflies under different thresholds. The model predicted bounding boxes based on the IoU threshold of 0.5. If the IoU was greater than 0.5, the model considered the predicted bounding box as positive. Otherwise, the model considered it as negative. Please note that this was just what the model thought, not the real true positive (TP), false positive (FP). The above-mentioned prediction results with the threshold of 0.5 could be compared with the ground truth, and values such as precision, recall, and F1 score could be calculated.

However, it was not enough to calculate precision, recall, and F1 score with IoU as the

threshold, and confidence should be added as the threshold because sometimes a predicted bounding box with a very low IoU and high confidence will be generated. In other words, because the bounding box predicted by the model only captured a small part of the features of the pest, although its category was correct (high confidence), the bounding box could not cover the overall outline of the pest (low IoU). Hence, in addition to IoU, needed to use confidence as the threshold.

Confidence was defined as the probability that the predicted bounding box belonged to a certain category. For example, the confidence was used as the threshold, so the probability of 0% 100% was divided into 1000 equal parts to form 1000 confidence thresholds, combined with the threshold of IoU equal to 0.5, precision, recall, and F1 score under 1000 threshold were calculated and drawn as a curve, which was precision versus recall curve and F1 score curve. In precision versus recall curve and F1 score curve, when the Area Under the Curve (AUC) is larger, it means that the category classification effect of the model is better, otherwise it means that the category classification effect of the model is worse.

Figs1. 4.2-4.3 show the precision-recall (PR) and F1 score curves of the YOLOv5 model with backbone “x6”. Fig. 4.2 displays the AUCs for each category, with thrips having the highest and black whiteflies having the lowest AUCs. This outcome is because black whiteflies can be easily mistaken for dust or debris, while thrips have distinct morphological characteristics and a larger size, making them simpler to identify. The F1 score, which represents the harmonic mean of precision and recall, is used to assess the model’s performance in solving multiclass classification problems. A perfect F1 score has a value of 1.0, implying both ideal precision and recall. Conversely, a score of 0 suggests that either precision or recall is 0. Setting the IoU threshold to 0.5 yields a precision of 93.8%,

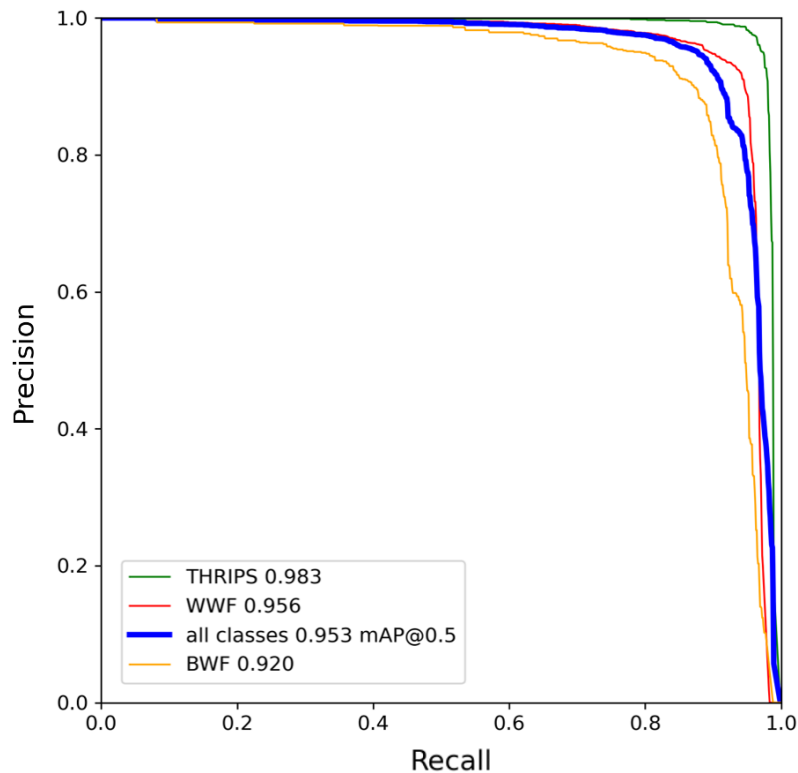
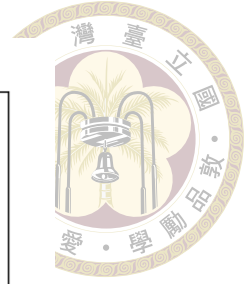


Figure 4.2: Curve of precision versus recall for pest detection with YOLOv5 model using “x6” backbone.

a recall of 91.9%, and an mAP of 95.3%, as achieved by the model. The probability for identifying white whiteflies and thrips, displayed on the diagonal of the confusion matrix in Fig. 4.4, is above 96%, while for black whiteflies it is around 90%. Hence, the selection and classification of whiteflies and thrips’ bounding boxes exhibit strong precision and dependability in the selected pest detection model.

The proposed system using AIoT technology aimed to establish favorable growth situations for asparagus by sensing the environment and monitoring pests in greenhouses. The primary objective of the object detection model was to detect changes in pest populations on a daily basis, while simultaneously monitoring the air temperature and humidity to avoid potential pest outbreaks. Fig. 4.5 displays the detection results of the YOLOv5 model with backbone “x6”. The model detected white whiteflies (WWF), black whiteflies

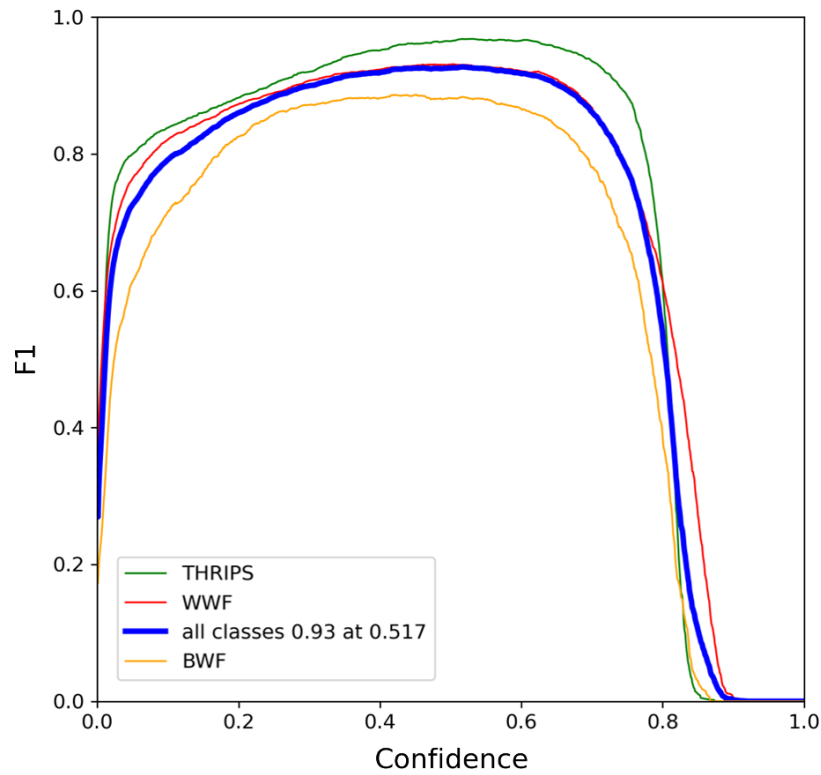
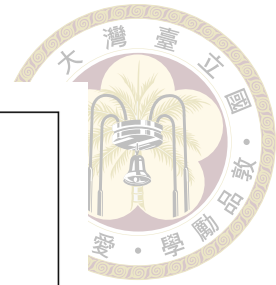


Figure 4.3: Curve of F1 scores for pest detection with YOLOv5 model using “x6” backbone.

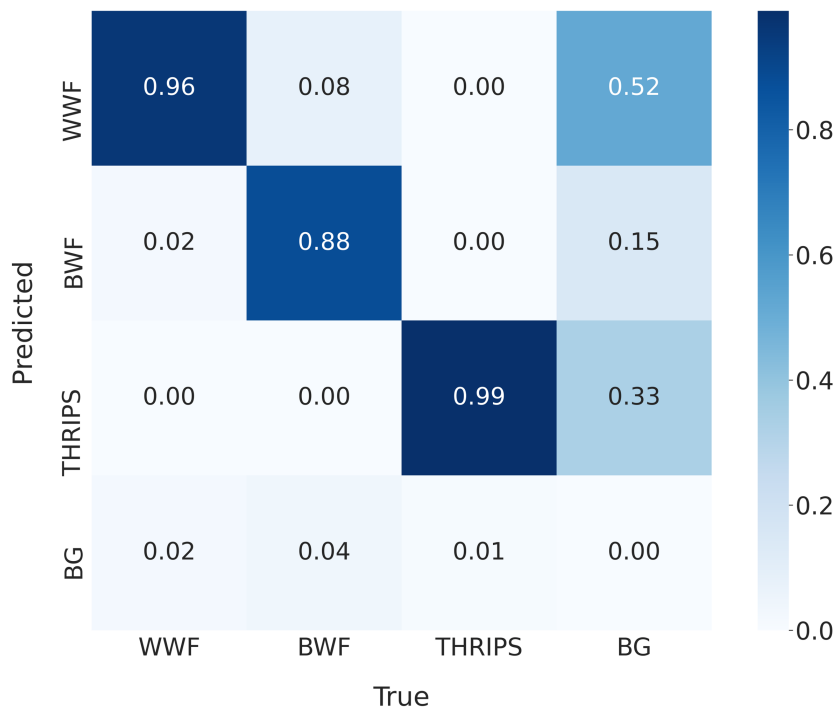


Figure 4.4: Confusion matrix of pests detected using YOLOv5 model with backbone “x6”.



Figure 4.5: Results of object classification and counting using YOLOv5 model with “x6” backbone

(BWF), and thrips, denoted by the corresponding labels. “WWF_label”, “BWF_label”, and “THRIPS_label” correspond to the numbers of labeled white whiteflies, black whiteflies, and thrips, respectively. “WWF_ratio”, “BWF_ratio”, and “THRIPS_ratio” represent the count accuracy of detected and labeled white whiteflies, black whiteflies, and thrips calculated in Eq. 4.1:

$$\text{count accuracy (\%)} = \left(1 - \frac{|D - M|}{M}\right) \times 100 \%, \quad (4.1)$$

where D represents the number of pests detected by the model and M is the number of manually labeled pests. This formula directly calculates the error between the model detected and the manually labeled number of pests. However, this algorithm may misestimate the actual results. Therefore, it is still necessary to compare with other indexes to obtain the actual model training results. Fig. 4.5 shows that the count accuracy of white and black whiteflies is 93.7% and 100%, respectively, in this example. Because no thrips were observed in the greenhouse, none was labeled nor detected.

To assess the practical usefulness and applicability of the trained pest detection model,



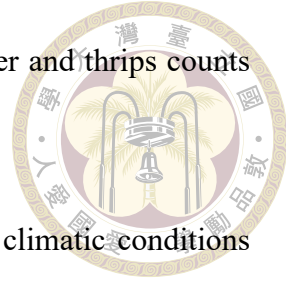
it was utilized to analyze images captured for one week, ranging from April 15, 2022, to April 21, 2022. Table 4.2 presents the results of the counting accuracy, which indicate an average accuracy of 95.8%.

Table 4.2: Mean count accuracy per day for seven consecutive days.

Date	Mean count accuracy
2022/04/15	0.972
2022/04/16	0.931
2022/04/17	0.957
2022/04/18	0.950
2022/04/19	0.974
2022/04/20	0.972
2022/04/21	0.951

4.1.3 Combining environmental sensing and pest monitoring measures

The influence of weather on crop disease and field pests is of utmost importance. Sunlight, wind direction, temperature, and rainfall are some of the weather parameters that significantly affect the population dynamics of pests. In particular, hot and dry weather has been found to have a positive correlation with the population growth of whiteflies and thrips (Das et al., 2011; Kumar, 2016; Abedin et al., 2020). The count of whiteflies displayed an upward trend with rising temperature, as shown in Fig. 4.6. Specifically, an increase was observed from day 1 to day 3, day 4 to day 10, and day 11 to day 22. A notable surge occurred from days 36 to 38, coinciding with a rise in temperature and a decrease in humidity. Moreover, the correlation analysis presented in Table 4.3 indicates that the whitefly counts exhibit a positive correlation with environmental temperature and a negative correlation with humidity. The pest numbers increased significantly on the same day with high temperature, whereas low humidity resulted in a rise in pest counts



the next day. Unfortunately, the correlation analysis between weather and thrips counts could not be performed due to their absence in the field.

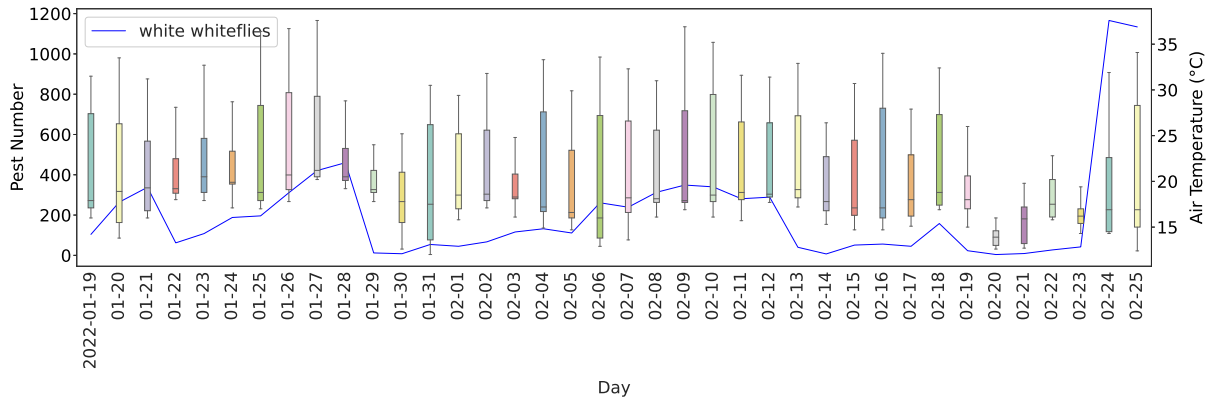
The findings of the correlation analysis align with the typical climatic conditions known to affect whiteflies, thus highlighting the possibility of establishing a link between ambient conditions and pest counts. This approach can provide early detection and preemptive warning of potential pest outbreaks. The correlation analysis findings from pest monitoring can be also effectively integrated into the on-site control system, enabling proactive measures such as activating fans to regulate temperature in case of excessive heat and spraying water mist when humidity levels are too low. This integration facilitates the realization of Integrated Pest Management (IPM) principles, which aim to address pest issues through a comprehensive and sustainable approach. By utilizing the correlation analysis results, the on-site control system can implement timely and targeted interventions, minimizing the reliance on chemical pesticides and promoting environmentally friendly pest management strategies.

Table 4.3: Cross-correlation of environmental temperature and humidity with pest numbers on various days. d_0 , d_{-1} , d_{-2} , and d_{-3} represent the current day, 1 day prior, 2 days prior, and 3 days prior, respectively.

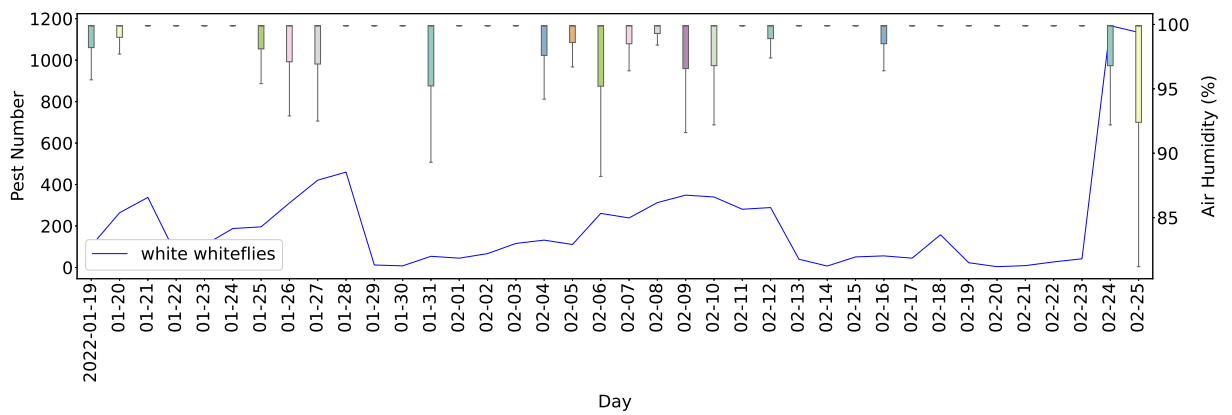
Environment temperature	Correlation coefficient	Environment humidity	Correlation coefficient
d_0	0.519	d_0	-0.065
d_{-1}	0.123	d_{-1}	-0.793
d_{-2}	0.162	d_{-2}	-0.399
d_{-3}	-0.056	d_{-3}	0.295

4.1.4 Practical applications

In addition, to enable real-time monitoring of pests, cameras in the asparagus field captured images of sticky paper traps every 10 minutes from 6 am until 8 pm, which



(a) Changes in pest counts and air temperature over 38 days.



(b) Changes in pest counts and air humidity over 38 days.

Figure 4.6: Combining sensing values and pest numbers. Boxplots of sensing values and line graphs of pest numbers from January 19, 2022, to February 25, 2022, for (a) air temperature and (b) air humidity.

were then transmitted to the FTP server. The proposed model was applied to identify whiteflies or thrips in the images, and the identified results were uploaded to the database. However, despite the high accuracy of our model in identifying both pests, thrips were absent in the experimental field. The website presents sensing values from the AIoT system, providing up-to-date information such as air temperature, humidity, pest population, and other ambient conditions in the experimental field of asparagus. (http://140.112.94.126/project/asparagus/Yizhu_Station_camdata1.php)

Compared to previous studies, our pest detection model's architecture is more effective for detecting little objects, resulting in higher accuracy and faster convergence. With the method's improved pest counting capabilities, farmers can now monitor pest populations in real-time and implement targeted pest control measures at different stages of growth, leading to more effective pest management in asparagus greenhouses. Control measures may include spraying with water, releasing natural predators, applying pesticides, and using pheromones to either attract and kill pests or disrupt their mating.

During the data collection period, no pest outbreaks were observed due to the diligent efforts of the cooperative farmers in maintaining optimal environmental conditions. However, Fig. 4.7 displays a pest outbreak of early 2020 indicated by an immediate surge in pest population on the fourth day. Our AIoT system of asparagus anticipates potential pest outbreaks by analyzing daily changes in pest populations (Fig. 4.7) in combination with environmental factors such as environmental temperature, humidity, and more. This approach provides farmers with additional sensing data to develop suitable control strategies for pests in different ambient conditions.

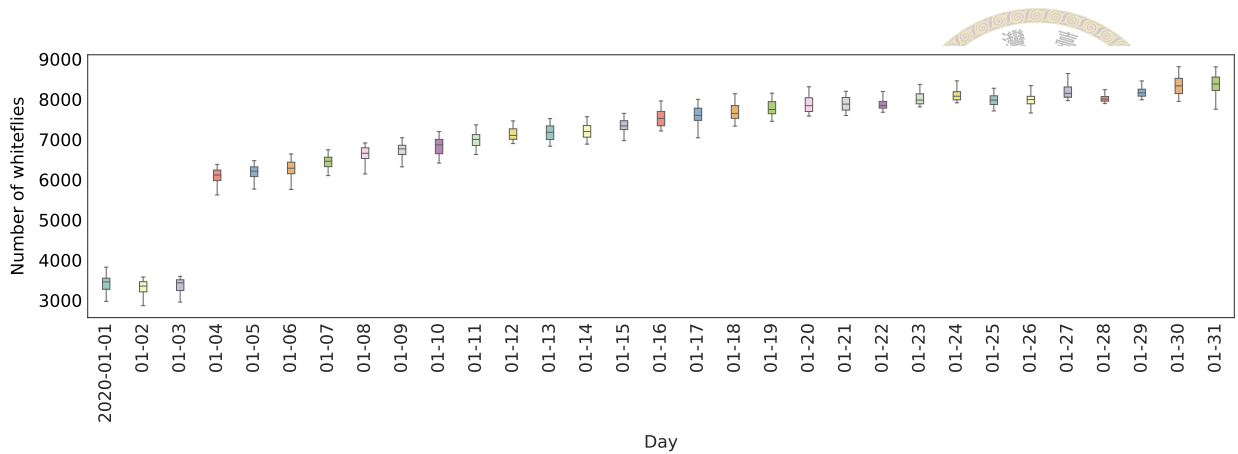


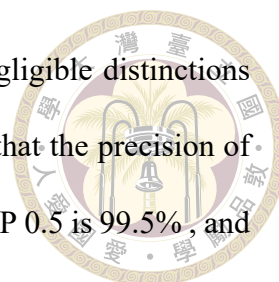
Figure 4.7: Results showing variations in the whitefly population during the pest outbreak from January 01, 2020 to January 31, 2020.

4.2 Optimization of bee pollination

4.2.1 Counting of honey bees entering and leaving the hive

4.2.1.1 Comparison of object detection models

The camera under the observation aisle was used to record the videos of honey bees entering and leaving the hive. To determine the optimal detection method for honey bees, we evaluated the performance of several models, including YOLOv3 with the “tiny” backbone (Redmon and Farhadi, 2018), YOLOv5 with the “n6” and “s6” backbones (Jocher, 2020), YOLOv7 with the “tiny” backbone (Wang et al., 2023), and YOLOv8 with the “n” and “s” backbones (Jocher et al., 2023). The training and validation sets consisting of 549 and 137 honey bee images, respectively, were used to compare the accuracy of each model. The training parameters used in the evaluation are listed in Table 3.2. Precision, recall, and mAP with an IoU threshold of 0.5 and average mAP with an IoU threshold of 0.5 to 0.95 are shown in Table 4.4. Because the difference in precision, recall, and mAP among the models was minimal when using an IoU threshold of 0.5, the average mAP within the IoU range of 0.5 to 0.95 was incorporated. This broader IoU threshold



range ensures a comprehensive evaluation while considering the negligible distinctions observed at the IoU threshold of 0.5 for each model. It can be seen that the precision of YOLOv8 with the backbone “s” is 98.7%, the recall is 99.6%, the mAP 0.5 is 99.5% , and the mAP 0.5:0.95 is 88.6%, which are all the best.

Table 4.4: Comparison of the precision, recall, and mean average precision (mAP) of all models.

Model	Backbone	Precision	Recall	mAP 0.5	mAP 0.5:0.95
YOLOv3	YOLOv3-tiny	0.990	0.987	0.993	0.782
YOLOv5	YOLOv5n6	0.990	0.997	0.996	0.811
YOLOv5	YOLOv5s6	1	0.999	0.996	0.825
YOLOv7	YOLOv7-tiny	0.997	0.990	0.995	0.836
YOLOv8	YOLOv8n	1	0.996	0.995	0.865
YOLOv8	YOLOv8s	0.987	0.996	0.995	0.886

4.2.1.2 Results of the YOLOv8 model with backbone “s”

The model’s performance was evaluated on testing data consisting of video screenshots that were not included in the training and validation sets. The YOLOv8 model with backbone “s” was considered as the optimal model due to its high mAP and low computation time. This model was then applied to classify and locate honey bees in the testing data, with the resulting bounding boxes shown in Fig. 4.8. All honey bees in the observation aisle images were accurately detected. The number of bounding boxes detected by the model was compared to the number of manually labeled honey bees.

By setting the IoU threshold to 0.5, the high AUC observed for honey bees in Fig. 4.9, along with the 100% probability shown on the diagonal of the confusion matrix in Fig. 4.11, clearly demonstrates the accurate identification of honey bees by the model. This success can be attributed to the distinctive morphological characteristics and larger size of honey bees. As a result, the selection of bounding boxes for honey bees exhibited good precision,

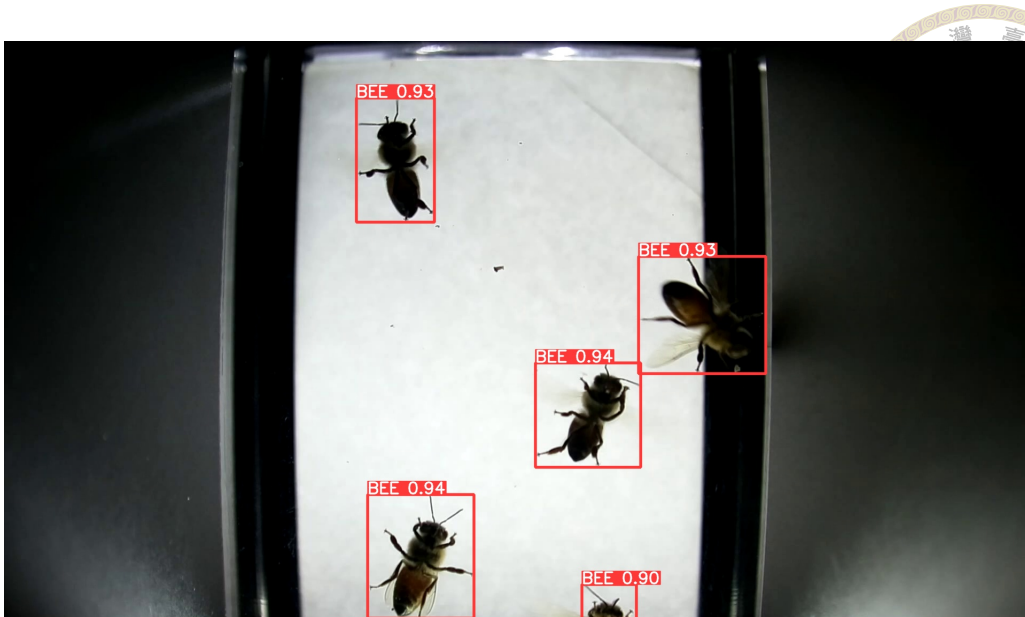


Figure 4.8: The resulting bounding boxes of honey bee using the YOLOv8 model with “s” backbone.

thus demonstrating the reliability of our honey bee detection model.

4.2.1.3 Comparison of the tracking performance

The AIoT system proposed aimed to optimize the pollination of honey bees in crop greenhouses. This was achieved by tracking the number of honey bees entering and leaving the hive, as well as the area of pollen grain carried by the bees. To prevent duplicate counting of honey bees and pollen grains, the tracking algorithm was designed to primarily monitor the honey bees’ entry and exit from the hive.

To validate the effectiveness and practicality of the object detection model and tracking algorithm, the system for counting honey bees entering and leaving the hive was applied to videos captured over six consecutive days from July 06, 2022 to July 11, 2022. There were 50 videos, each 1 minute in length, recorded at different times by the honey bee monitoring system as testing data. The videos’ resolution was 1920×1080 with 25 frames per second. In general, only the honey bees entering the hive would carry pollen

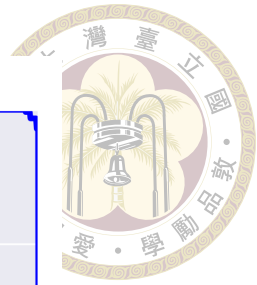
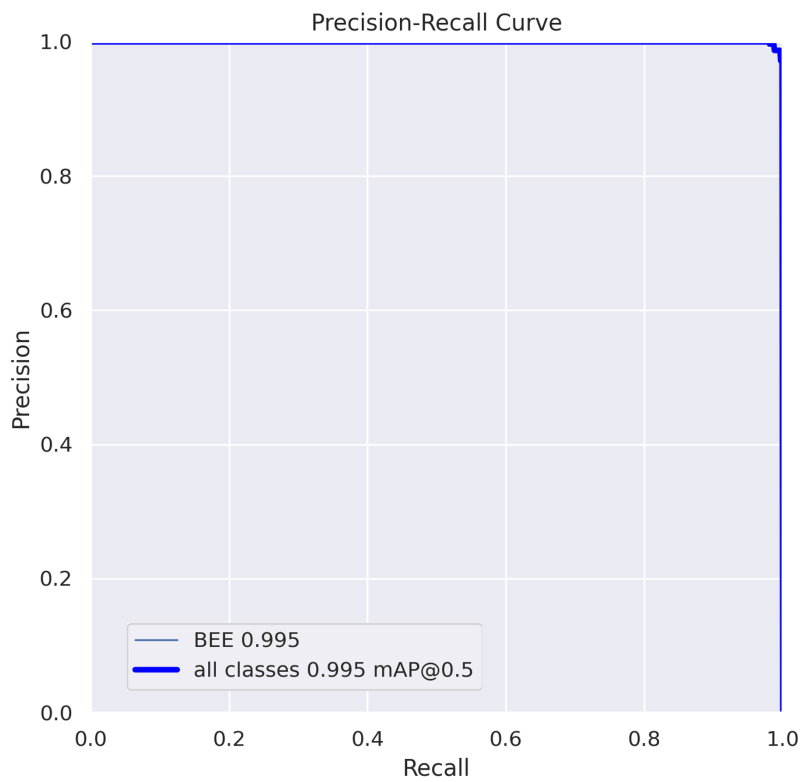


Figure 4.9: Curve of precision vs. recall for honey bee with YOLOv8 model using “s” backbone.

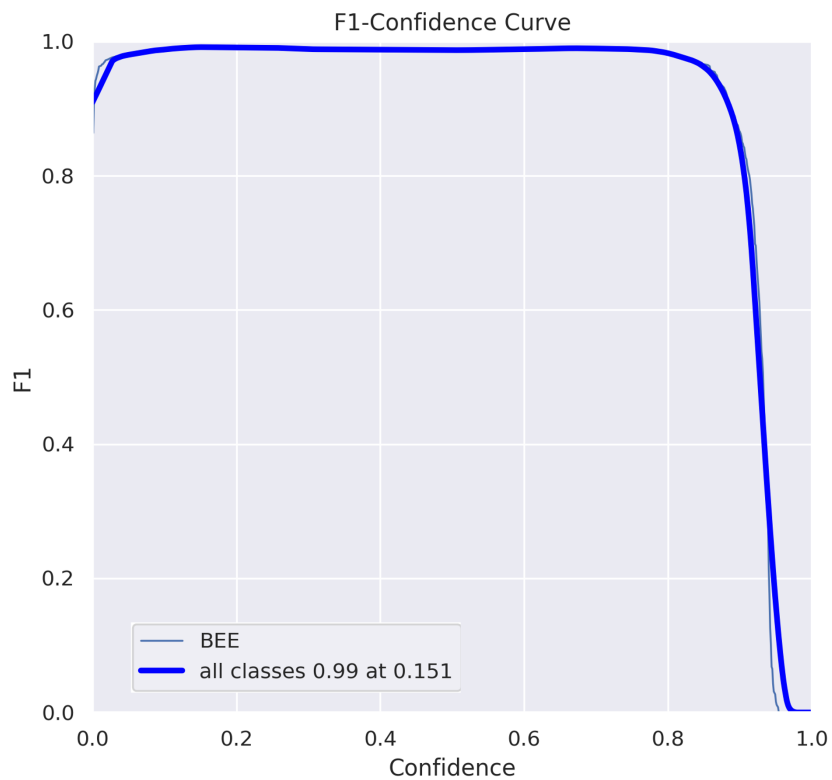


Figure 4.10: Curve of F1 scores for honey bee with YOLOv8 model using “s” backbone.

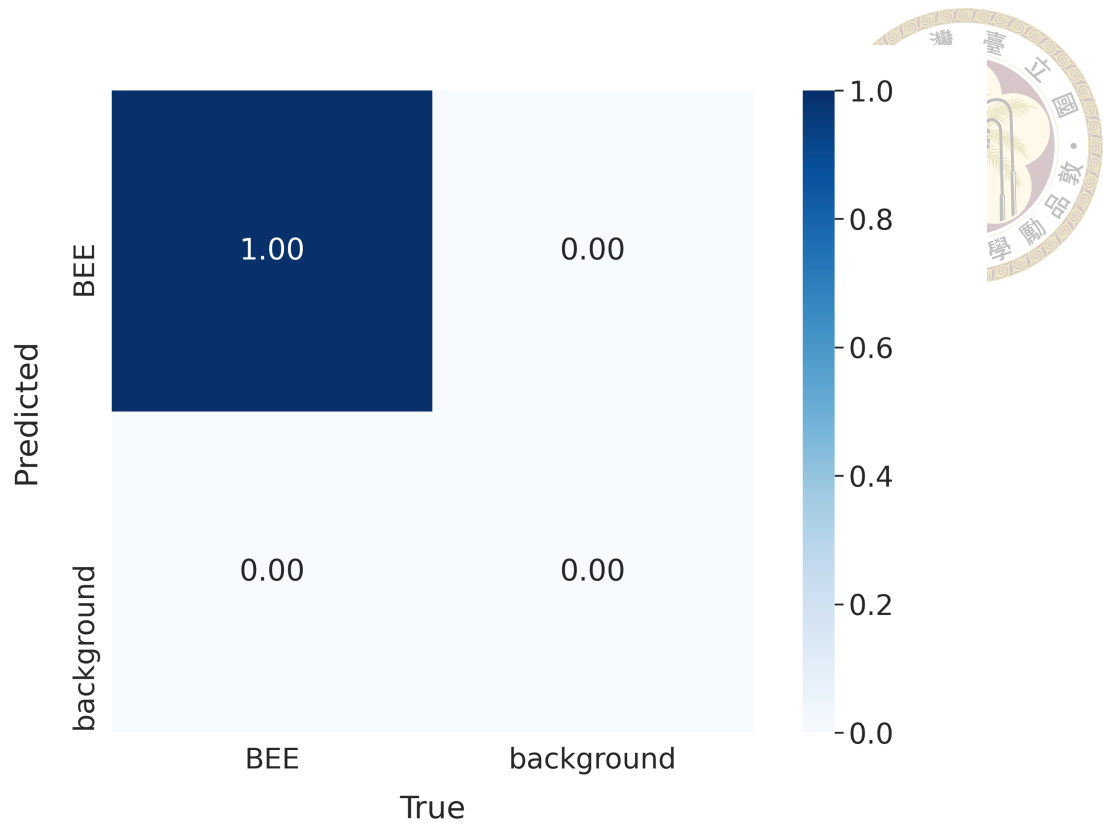


Figure 4.11: Confusion matrix of honey bees detected with YOLOv8 model using “s” backbone.

grains. Therefore, the number of honey bees entering the hive in the videos was counted for subsequent pollen grain area calculations. The testing result automatically counted by the DeepSORT or StrongSORT tracking algorithm was compared with the result manually counted. The tracking accuracy of automatically and manually counted honey bees calculated in Eq. (4.2) (Ngo et al., 2021):

$$\text{Tracking accuracy (\%)} = \frac{1}{n} \left(1 - \frac{|A - M|}{M} \right) \times 100 \%, \quad (4.2)$$

where n is the number of testing videos, A represents the total number of honey bees counted automatically by the tracking algorithm in n testing videos, and M is the total number of honey bees counted manually in n testing videos.

When the same weight file of YOLOv5 with backbone “s6” was used, the tracking

accuracy of DeepSORT and StrongSORT were 73.7% and 93.4%. Since StrongSORT was modified based on DeepSORT, it has better tracking performance. In addition, ID Switch is a common challenge in the tracking algorithm, that is, when two objects overlap, the IDs of the two objects will be exchanged, which will lead to tracking errors. However, through the improvement of each component of StrongSORT and our repeated testing, it was also found that the ID Switch situation was less common after tracking through StrongSORT from the testing results.

After confirming that StrongSORT did have better tracking performance than DeepSORT, I put the weight files of the YOLOv3 model with backbone “tiny”, the YOLOv5 model with backbone “n6” and “s6”, and the YOLOv8 model with backbone “n” and “s” into StrongSORT to compare its tracking accuracy, as shown in Table 4.5.

Table 4.5: Comparison of the StrongSORT tracking accuracy of different model weight files.

Model	Backbone	Tracking accuracy
YOLOv3	YOLOv3-tiny	0.892
YOLOv5	YOLOv5n6	0.901
YOLOv5	YOLOv5s6	0.934
YOLOv7	YOLOv7-tiny	0.911
YOLOv8	YOLOv8n	0.967
YOLOv8	YOLOv8s	0.977

In general, the backbone of the object detection model paired with the tracking algorithm should be as simple as possible to save the calculation time of each frame. However, since the calculation of the pollen grain area after the counting of honey bees entering and leaving the hive does not require real-time calculation, accuracy is more important than computation speed. Therefore, the counting of honey bees entering and leaving the hive in this study should focus on accuracy instead. In Table 4.4, the detection performance of YOLOv3, YOLOv5, and YOLOv7 models is not much different, while the detection per-

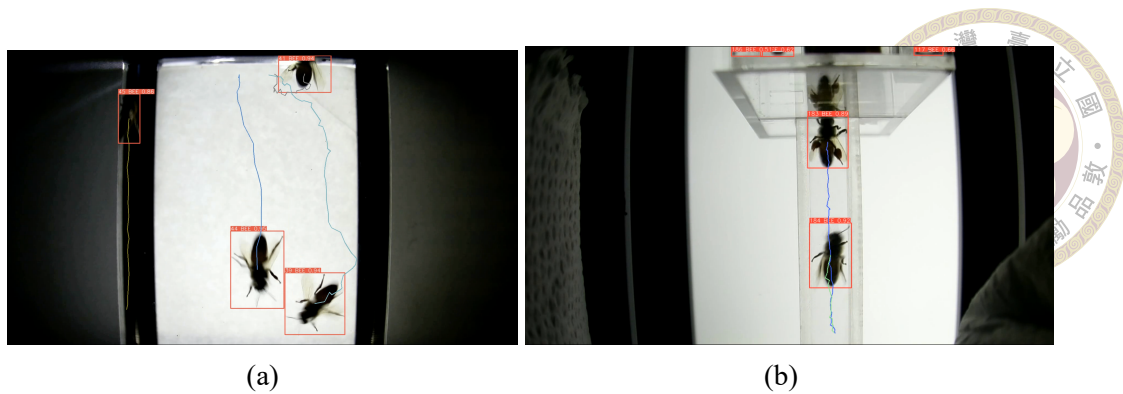


Figure 4.12: The honey bee tracking results of version 1 (a) and version 2 (b) observation boxes using the YOLOv8 model with “s” backbone in conjunction with the StrongSORT (Du et al., 2023) tracking algorithm.

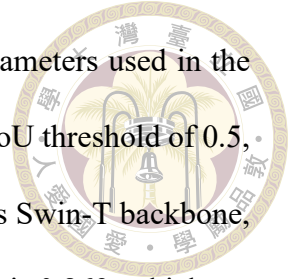
formance of YOLOv8 is significantly better, and the detection performance of YOLOv8 with backbone “s” is the best. Moreover, as depicted in Table 4.5, there is a noticeable enhancement in tracking accuracy when utilizing newer object detection models or employing the same model with more intricate backbones. Therefore, YOLOv8 with backbone “s” was considered to be paired with StrongSORT to count honey bees entering and leaving the hive. The honey bee tracking results of the combination are shown in Fig. 4.12.

4.2.2 Calculation of pollen grain area

4.2.2.1 Comparison of instance segmentation models

To find the best calculation result for the area of pollen grain carried by honey bees, three types of models were evaluated for their pollen grain segmentation accuracy. These models included the one-stage model YOLACT (Bolya et al., 2019, 2022), YOLOv5 model (Jocher, 2020), YOLOv7 model (Wang et al., 2023), YOLOv8 model (Jocher et al., 2023), the two-stage model Mask R-CNN using ResNet50 backbone (He et al., 2017) and the transformer series model Mask R-CNN using Swin-T backbone (Liu et al., 2021). The training and validation sets consisting of 1025 and 259 pollen grain images, respectively,

were used to compare the accuracy of each model. The training parameters used in the evaluation are listed in Table 3.3. The model testing results, with an IoU threshold of 0.5, are shown in Table 4.6. For the Mask R-CNN with transformer series Swin-T backbone, it can be seen that the bounding box mAP is 0.884 and the mask mAP is 0.869, which are all the best.



Although the mask mAP provides an evaluation of the combined performance of bounding box localization and mask generation, when the individual effects of localization and mask need to be clarified, it is necessary to focus solely on the bounding box mAP. Analyzing Table 4.6 reveals that there is minimal difference between the YOLACT model and the Mask R-CNN model with the Swin-T backbone in terms of mask mAP. In fact, YOLACT even outperforms Mask R-CNN with the Swin-T backbone in this aspect. However, when considering only the bounding box mAP, YOLACT falls short compared to Mask R-CNN with the Swin-T backbone. In conclusion, considering the already satisfactory mask generation effects, this task leans towards selecting the Mask R-CNN model with the Swin-T backbone due to its superior localization capabilities. After all, when the detection of pollen grains is not achieved, the quality of mask generation becomes irrelevant. Therefore, prioritizing accurate localization is crucial for ensuring the effectiveness of the model in this task.

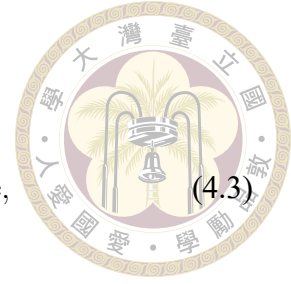
Table 4.6: Comparing the mAP of all instance segmentation models.

Model	Backbone	Bounding box mAP	Mask mAP
YOLACT	ResNet50	0.878	0.872
YOLOv5	YOLOv5x-seg	0.872	0.845
YOLOv7	YOLOv7-seg	0.891	0.859
YOLOv8	YOLOv8x-seg	0.888	0.862
Mask R-CNN	ResNet50	0.831	0.821
Mask R-CNN	Swin-T	0.884	0.869

To further compare the pollen grain segmentation of various models, we calculate the

segmentation score through Eq. (4.3):

$$\text{segmentation score} = \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M \text{Pr} \cdot \text{IoU score}, \quad (4.3)$$



where N is the number of testing images. M is the number of segmented objects in each image. Pr represents whether each object is detected, and it is 1 if it is detected, otherwise it is 0. IoU score represents the ranking of the mask segmentation effect between the models by manual recognition. Please note that it will be distinguished between different models with a score value, such as 6, 5, 4, 3, 2, and 1.

Scoring examples of the segmentation score are shown in Fig. 4.13. Taking Fig. 4.13(a) as an example, the image of the first row of bees in YOLOv8 is analyzed. In this case, M represents a single image, and N indicates the total number of segmented objects in the row, which is 2. The upper object ($\text{Pr} = 1$) is considered the best segmentation result among the models in the same row, resulting in an IoU score of 6. However, the lower object is not detected ($\text{Pr} = 0$). Taking Fig. 4.13(b) as another example, the image of the third row of bees in YOLOv5 is considered. Similarly, M represents a single image, and N indicates the total number of segmented objects in the row, which is 3. Among these objects, the upper-left one ($\text{Pr} = 1$) is ranked third based on the segmentation results of the models in the same row, yielding an IoU score of 4. The lower-left object is a reflection and should not be recognized, so as long as it is not detected, it receives an IoU score of 6. Finally, the right object ($\text{Pr} = 1$) is considered the second-best segmentation result in the row, resulting in an IoU score of 5.

The segmented images of several models, including the YOLACT model, YOLOv5 model, YOLOv7 model, YOLOv8 model, the Mask R-CNN using ResNet50 backbone,



YOLOv5	YOLOv7	YOLOv8	YOLACT	Mask R-CNN	Swin
 (0+0)/2	 (0+5)/2	 (6+0)/2	 (0+0)/2	 (4+0)/2	 (5+6)/2
 (6)/1	 (6)/1	 (6)/1	 (6)/1	 (6)/1	 (6)/1
 (5+4)/2	 (4+4)/2	 (5+4)/2	 (6+5)/2	 (3+6)/2	 (3+6)/2
 (0+4+6)/3	 (5+0+6)/3	 (5+0+6)/3	 (0+5+6)/3	 (6+0+0)/3	 (6+6+6)/3

(a) Version 1

YOLOv5	YOLOv7	YOLOv8	YOLACT	Mask R-CNN	Swin
 (0)/1	 (4)/1	 (0)/1	 (0)/1	 (5)/1	 (6)/1
 (5)/1	 (5)/1	 (5)/1	 (0)/1	 (0)/1	 (6)/1
 (4+6+5)/3	 (4+6+5)/3	 (5+6+5)/3	 (6+6+6)/3	 (6+0+6)/3	 (6+6+6)/3
 (6)/1	 (6)/1	 (6)/1	 (6)/1	 (0)/1	 (6)/1

(b) Version 2

Figure 4.13: Scoring examples of segmentation score for version 1 and 2 observation boxes, where red numbers are segmentation scores.

and the Mask R-CNN model using Swin-T backbone, were evaluated. The pollen grain testing set was used in the comparison, with 259 pollen grain images. The model segmentation results are shown in Table 4.7. For the Mask R-CNN with transformer series Swin-T backbone, it can be seen that the segmentation score is 4.994, which is all the best.

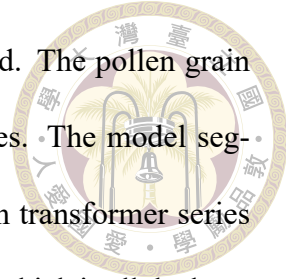


Table 4.7: Comparing the segmentation result of all instance segmentation models.

Model	Backbone	Segmentation score
YOLOACT	ResNet50	4.404
YOLOv5	YOLOv5x-seg	3.809
YOLOv7	YOLOv7-seg	3.971
YOLOv8	YOLOv8x-seg	3.830
Mask R-CNN	ResNet50	4.484
Mask R-CNN	Swin-T	4.994

By amalgamating the findings presented in Fig. 4.13 and Table 4.7, two key observations come to light. Firstly, the one-stage model YOLACT falls slightly behind the transformer series model in terms of segmentation outcomes but outperforms the two-stage model. Secondly, within the one-stage model, the YOLO family of models prioritizes bounding box detection over generating complete masks. This is evident in the first row of Fig. 4.13(a) and the second row of Fig. 4.13(b), where the bounding box detection results are comparable to those of the transformer series models, but the mask is not as refined.

In relation to the first observation, the reason behind YOLACT achieving the highest segmentation score among the one-stage models appears to lie in its increased emphasis on mask extraction within its model architecture. This heightened focus enables YOLACT to produce more precise and intricate masks once the bounding box is delineated. Furthermore, YOLACT outperforms the two-stage Mask R-CNN model with a ResNet50 backbone in terms of segmentation results. This can be attributed to YOLACT's utilization of newer feature extraction techniques, which allow the one-stage model to extract

features that are on par with those of the two-stage model.

The second observation relates to the YOLO series models' inability to generate fine masks. I believe this limitation stems from the fact that the YOLO series models are primarily built upon the architecture of object detection models, with added components for mask feature extraction and generation. While these models excel in bounding box detection and demonstrate commendable performance in this aspect, their capacity for comprehensive mask feature extraction is not as refined as that of other models, such as the YOLACT model, the Mask R-CNN using ResNet50 backbone, and the Mask R-CNN model using Swin-T backbone.

However, given the emphasis on achieving high segmentation accuracy in this particular task of pollen grain segmentation, the final selection was made based on a comprehensive evaluation of bounding box detection capabilities and complete mask generation capabilities. As a result, the Mask R-CNN model, utilizing the Swin-T backbone from the transformer series, emerged as the preferred choice due to its exceptional performance in both aspects.

4.2.2.2 Results of the Mask R-CNN model with backbone Swin-T

The Mask R-CNN model with Swin-T backbone, which had the highest mAP, was applied to segment pollen grains of honey bees in the testing data. The images on both sides of the observation box were equally allocated to the dataset. The masks of pollen grains detected by the Mask R-CNN model with backbone Swin-T are shown in Fig. 4.14, respectively. It can be seen that almost all the pollen grains on images on both sides of the observation box have been segmented. The results of segmentations for pollen grains



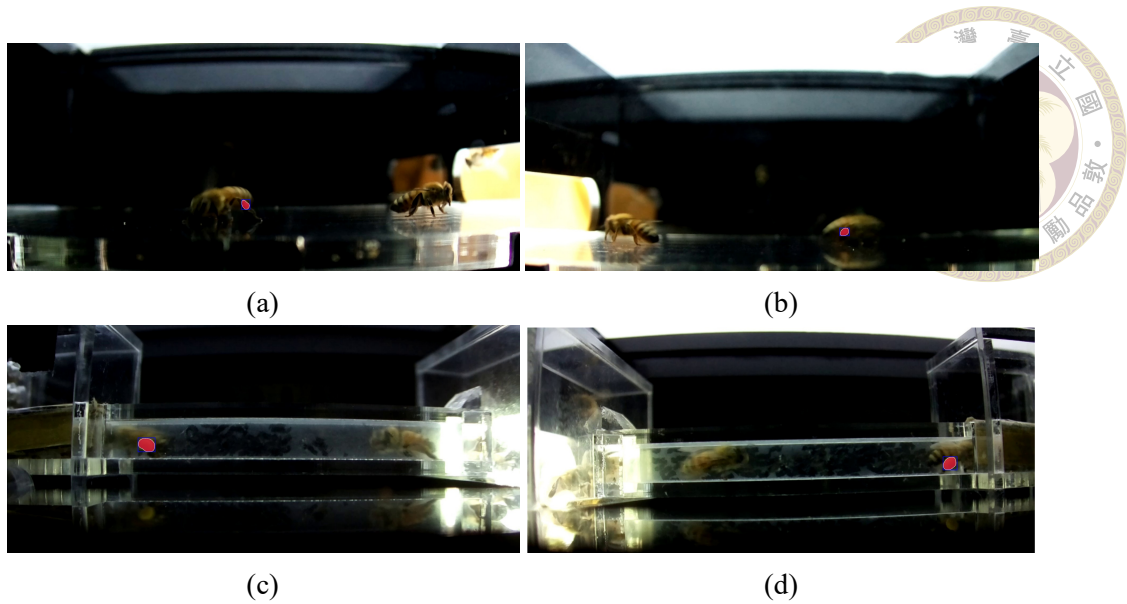


Figure 4.14: The Mask R-CNN model with Swin-T backbone was used to obtain segmentation results for pollen grains in both version 1 and version 2 observation boxes. Images (a) and (b) show the segmentation results for pollen grains on the left and right sides of version 1. Images (c) and (d) display the segmentation results for pollen grains on the left and right sides of version 2.

were compared with the results of manual labels.

By setting the IoU threshold to 0.5, the AUC depicted in Fig. 4.15-4.16, combined with the 87% probability displayed on the diagonal of the confusion matrix in Fig. 4.17, serves as evidence of the model's proficiency in performing instance segmentation for pollen grains. The task presents a significant challenge as pollen grains can be easily mistaken for the honey bee's body color and reflections. However, despite this difficulty, the model achieved commendable precision in detecting and segmenting pollen grains, thereby establishing the reliability of our pollen grain model.

4.2.2.3 Results of the algorithm of pollen grain area calculation

As can be seen from Fig. 4.18, after the Mask R-CNN model with backbone Swin-T segmented the pollen grain area, the bounding box area would be cropped by setting the range of the HSV color space to avoid calculating the area of the non-mask area. Then the

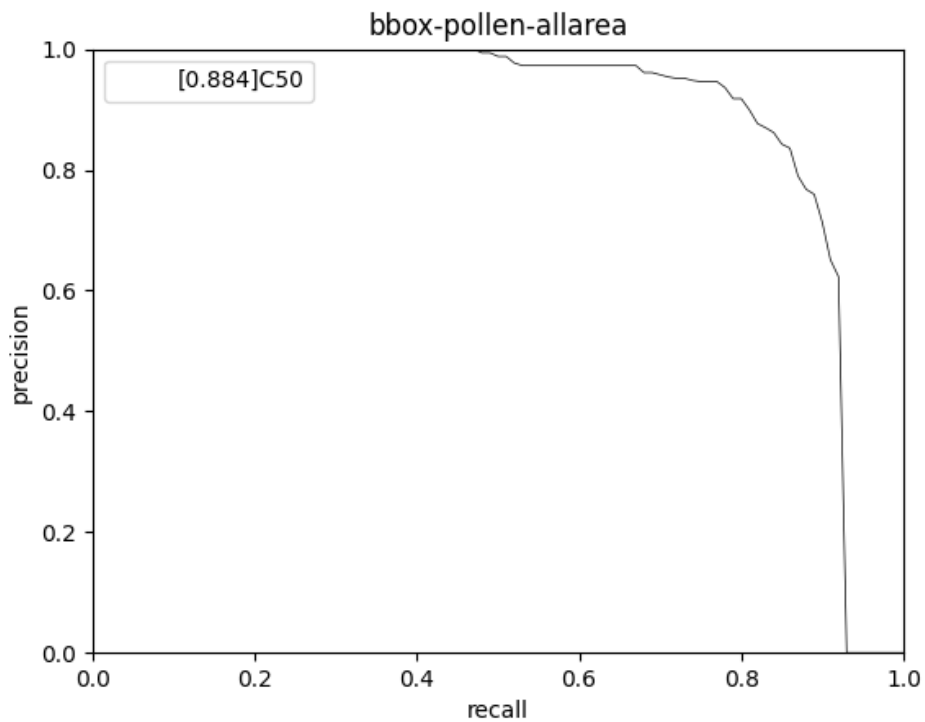
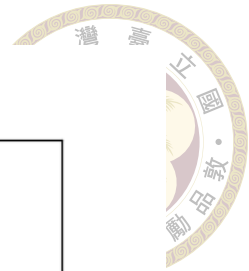


Figure 4.15: The detection curve of precision vs. recall for pollen grains with the Mask R-CNN model using Swin-T backbone. The numbers in the upper left corner of the figure represent bounding box mAP.

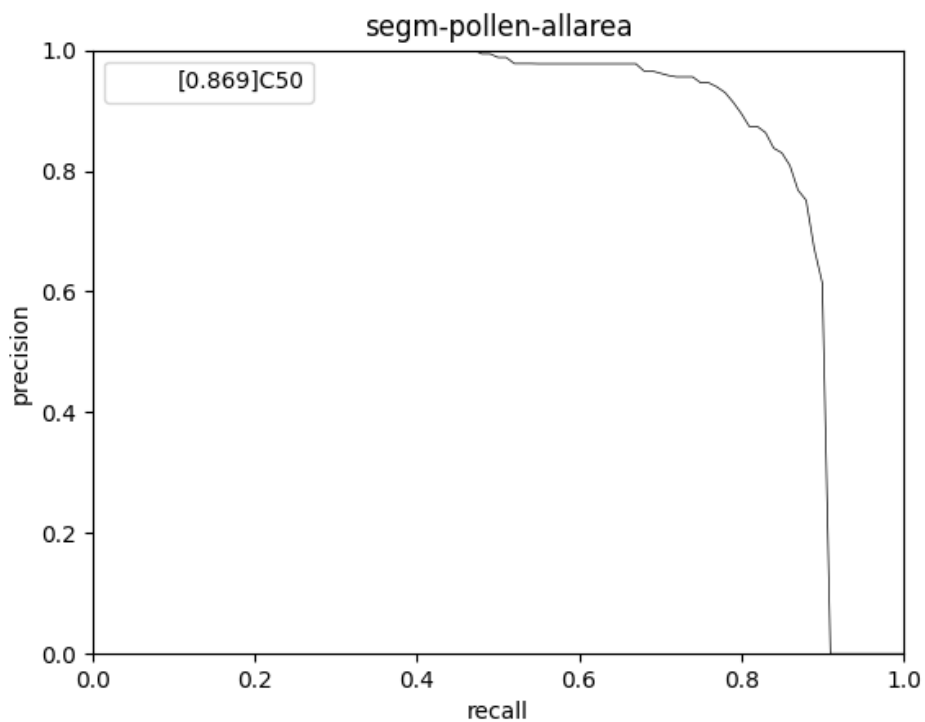


Figure 4.16: The segmentation curve of precision vs. recall for pollen grains with the Mask R-CNN model using Swin-T backbone. The numbers in the upper left corner of the figure represent mask mAP.

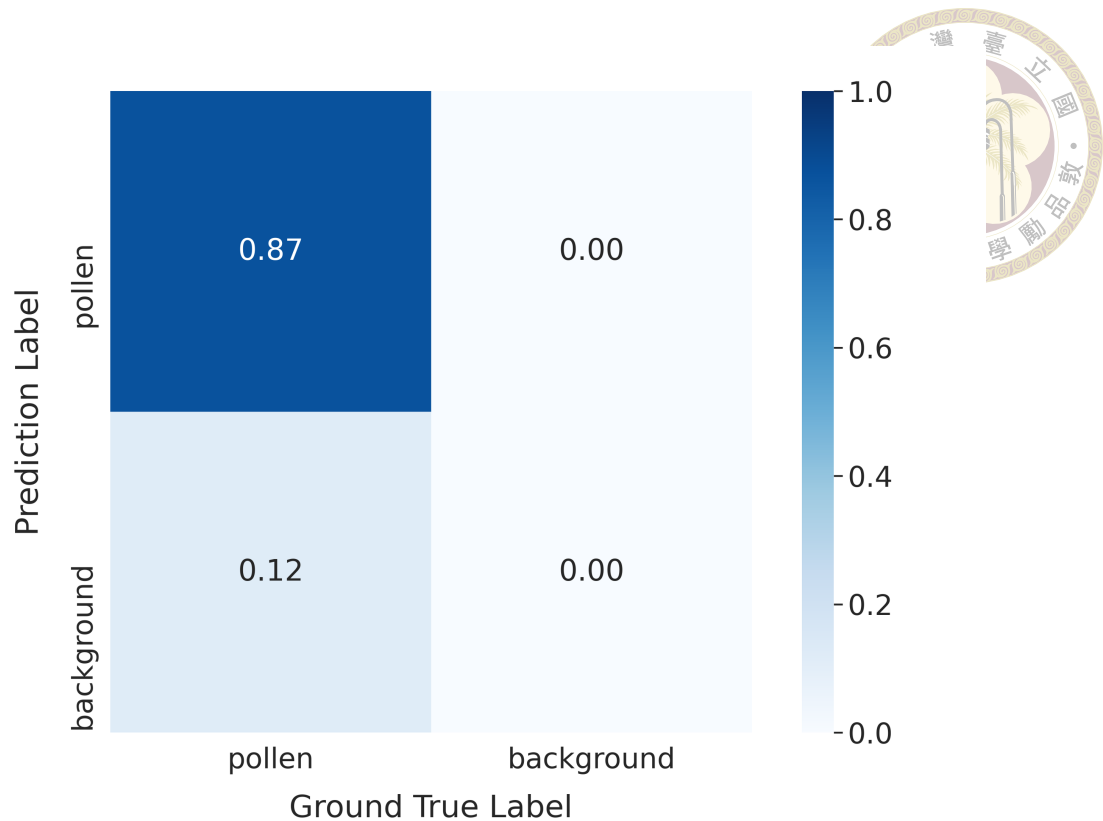


Figure 4.17: Confusion matrix of pollen grains detected with the Mask R-CNN model using Swin-T backbone.

mask area was found from the bounding box area by setting the range of the HSV color space again. Finally, the number of pixels contained in the mask area was calculated to obtain the pollen grain area, as shown in Fig. 4.18(b) and Fig. 4.18(d).

4.2.3 Correlation analysis between pollen count and actual pollination status of crops

The experiment data of the comparison are shown in Tables 4.8-4.10. During the experiment, data were collected at approximately 10-minute intervals. However, due to unforeseen circumstances, there were variations in the time intervals recorded, ranging from 10 minutes to 30 minutes, as shown in Tables 4.8-4.10. To ensure the fairness of data processing, the area and weight of all pollen grains corresponding to each time interval were still recorded. The calculation results of the pollen grain area algorithm were

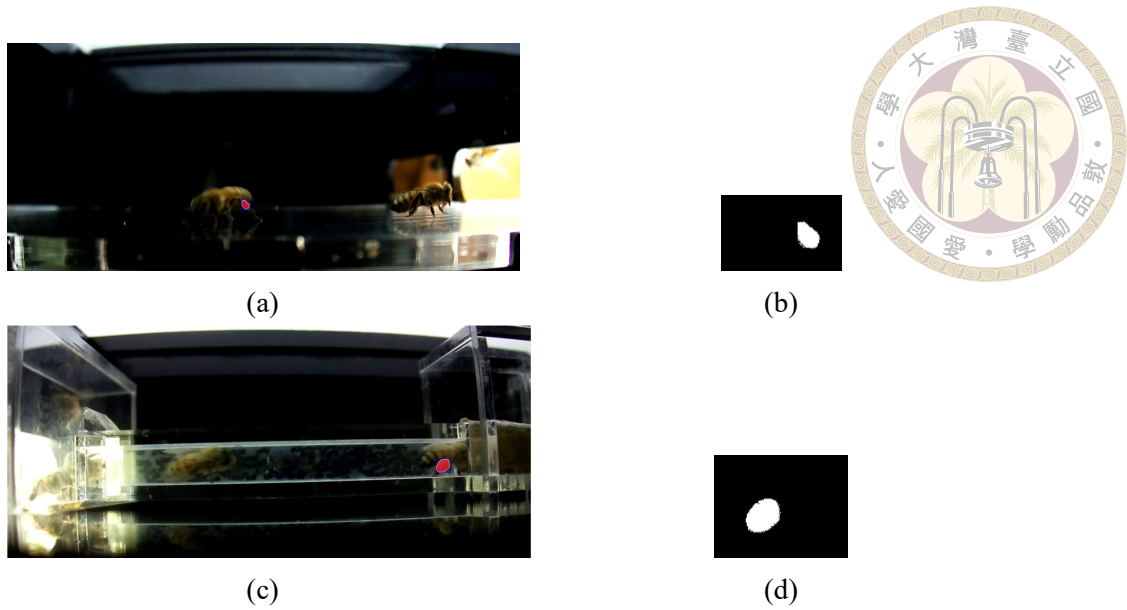


Figure 4.18: Images (a) and (c) show the segmentation results for pollen grains in version 1 and version 2 observation boxes, respectively. The Mask R-CNN model with Swin-T backbone was used for both. Images (b) and (d) depict the calculation results for version 1 and version 2, respectively, using the algorithm (white pixels indicate calculated pixels).

compared with the actual pollen grain weight, and we found that the coefficient of determination between the two was 0.930, shown in Fig. 4.19. The translucent bands surrounding the regression line in Fig. 4.19 represent the 95% confidence interval for the regression estimate. Therefore, we can confirm that the pollen area algorithm can indeed reflect the real pollen weight, then infer the actual pollination status of the crop, and optimize the pollination of honey bees.

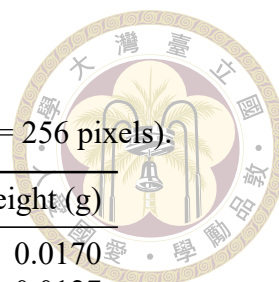


Table 4.8: The experimental data on January 13, 2023 (1 mm² = 256 pixels).

Time interval	Pollen grain area (mm ²)	Pollen grain weight (g)
1040-1050	74.433594	0.0170
1050-1100	67.125000	0.0127
1100-1110	110.335938	0.0260
1110-1120	65.664062	0.0105
1120-1130	69.796875	0.0080
1130-1140	74.183594	0.0092
1140-1150	47.886719	0.0077
1150-1200	14.542969	0.0027
1210-1230	103.339844	0.0318
1230-1400	163.183594	0.0503
1400-1420	51.597656	0.0238
1430-1510	160.308594	0.0078
1510-1540	159.207031	0.0019

Table 4.9: The experimental data on March 05, 2023 (1 mm² = 256 pixels).

Time interval	Pollen grain area (mm ²)	Pollen grain weight (g)
1020-1030	41.453125	0.1963
1030-1040	669.492188	0.4333
1040-1050	111.605469	0.2338
1100-1110	15.023437	0.0709
1110-1120	130.863281	0.1540
1120-1130	175.566406	0.1605
1130-1140	47.285156	0.0437
1140-1150	0.000000	0.0079
1150-1200	39.851562	0.0276
1200-1220	26.683594	0.0684
1220-1320	115.812500	0.1777
1320-1340	70.000000	0.1291
1350-1400	45.855469	0.0437
1400-1420	22.421875	0.0358
1420-1430	62.796875	0.0611
1430-1440	29.828125	0.0210
1440-1450	35.207031	0.0344
1450-1500	20.480469	0.0124
1500-1510	51.71875	0.0417
1510-1520	46.023438	0.0491
1520-1530	18.535156	0.0097

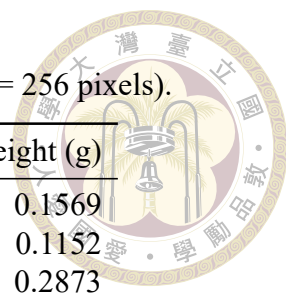


Table 4.10: The experimental data on March 06, 2023 ($1 \text{ mm}^2 = 256 \text{ pixels}$).

Time interval	Pollen grain area (mm^2)	Pollen grain weight (g)
1010-1020	356.941406	0.1569
1020-1030	96.6875	0.1152
1030-1050	344.867188	0.2873
1050-1100	189.082031	0.1306
1100-1110	107.500000	0.0445
1110-1120	81.867188	0.0215
1120-1130	153.941406	0.0691
1130-1140	245.859375	0.0769
1140-1150	110.191406	0.0358
1150-1200	79.894531	0.0242
1200-1210	112.023438	0.0291
1210-1310	186.054688	0.0586
1310-1330	155.417969	0.0486
1330-1340	54.828125	0.0129
1340-1350	105.531250	0.0411
1350-1400	95.941406	0.0318
1400-1410	35.433594	0.0150
1410-1420	109.695312	0.0316
1420-1430	34.019531	0.0093
1430-1440	56.667969	0.0232
1440-1450	151.210938	0.0391
1450-1500	79.265625	0.0293
1500-1510	100.410156	0.0290
1510-1520	205.144531	0.0522
1520-1530	36.570312	0.0225

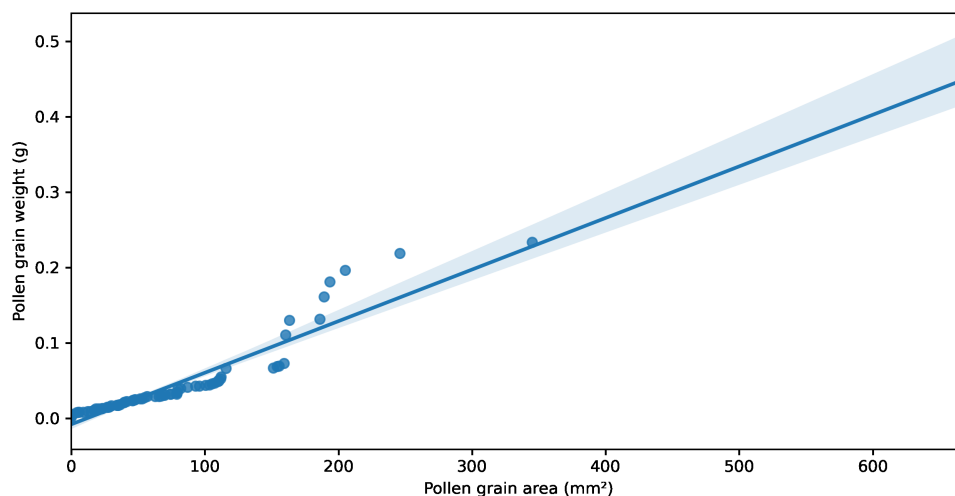


Figure 4.19: Regression curve of the experimental data on January 13, March 05, and March 06, 2023.

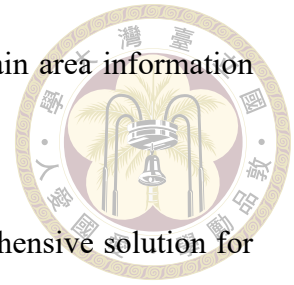


Chapter 5 Conclusion

This thesis introduced an innovative AIoT system that integrated IoT-based environmental sensor measurements and pest count analysis using an AI-based YOLOv5 model with the backbone “x6”. The selection of the YOLOv5 model with backbone “x6” was based on its superior mAP compared to two-stage models and its faster convergence rate compared to transformer series models. By harnessing big data analysis, the system utilized environmental data and pest population information to forecast pest outbreak trends, thereby maximizing the yield of high-quality asparagus through appropriate control strategies for pests in facility cultivation.

Additionally, the thesis proposed an AIoT system that combined an IoT-based observation box with AI-based models to address honey bee monitoring and pollination optimization. The YOLOv8 object detection model with the backbone “s”, the latest YOLO model, along with the StrongSORT tracking algorithm, accurately captured the counts of honey bees entering and leaving the beehive. Furthermore, the calculation of pollen grain area was performed using the Mask R-CNN instance segmentation model with the Swin-T backbone from the transformer series, in conjunction with image processing technology. The choice of the Mask R-CNN model with the Swin-T backbone prioritized accuracy in pollen grain area estimation rather than segmentation speed, as it demonstrated excellent mask generation by practical tests and the best bounding box localization capabilities in

model comparison. By applying correlation analysis, the pollen grain area information can be effectively utilized to enhance honey bee pollination.



These research findings presented a more efficient and comprehensive solution for greenhouse monitoring of pests and optimization of honey bee pollination, contributing to the realization of the vision of smart agriculture.



Chapter 6 Future work

In the future, several follow-up studies can be conducted based on the current research findings. Firstly, the correlation analysis results of pest monitoring can be integrated with the on-site control system, such as turning on the fan when the temperature is too high, and spraying water mist when the humidity is too low, so as to achieve the vision of IPM. Secondly, a correlation analysis between the area of pollen grains and the actual pollination status of crops can be carried out. Since suitable crops were unavailable during the research period, this experiment can be pursued in the future to further validate the research results. Thirdly, big data analysis of the pollen grain area results and environmental sensing data can be performed to optimize bee pollination that aligns with environmental monitoring values. This analysis will enhance the understanding of the relationship between pollen grain area and environmental factors. Lastly, real-time pest counting and bee pollination optimization can be conducted in the field using advanced edge devices or faster and more reliable communication technologies. This would allow farmers to promptly receive the pest population and bee pollination optimization results, enabling them to make immediate adjustments to their pest control and pollination methods. This approach would provide convenience and timely feedback for optimizing pest control and pollination practices.



References

- Abedin, S., Alireza, Z.-R., Hassan, V., Amin, A., Reza, F.-F., Amir, H., and Ali, H. (2020). Relationship between some environmental and climatic factors on outbreak of white-flies, the human annoying insects. *Journal of Arthropod-Borne Diseases*, 14:78–87.
- Agarap, A. F. (2018). Deep learning using rectified linear units (ReLU). *arXiv preprint arXiv:1803.08375*.
- Ajao, L. A., Agajo, J., Kolo, J. G., Maliki, D., and Adegboye, M. A. (2017). Wireless sensor networks based-internet of thing for agro-climatic parameters monitoring and real-time data acquisition. *Asian Scientific Research*, 7:240–252.
- Baronti, P., Pillai, P., Chook, V. W., Chessa, S., Gotta, A., and Hu, Y. F. (2007). Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. *Computer communications*, 30(7):1655–1695.
- Bernauer, O. M., Tierney, S. M., and Cook, J. M. (2022). Efficiency and effectiveness of native bees and honey bees as pollinators of apples in new south wales orchards. *Agriculture, Ecosystems & Environment*, 337:108063.
- Bochkovski, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.



Bolya, D., Zhou, C., Xiao, F., and Lee, Y. J. (2019). YOLACT: Real-time instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9157–9166.

Bolya, D., Zhou, C., Xiao, F., and Lee, Y. J. (2022). YOLACT++ better real-time instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):1108–1121.

Caicedo-Ortiz, J. G., De-la Hoz-Franco, E., Ortega, R. M., Piñeres-Espitia, G., Combita-Niño, H., Estévez, F., and Cama-Pinto, A. (2018). Monitoring system for agronomic variables based in wsn technology on cassava crops. *Computers and Electronics in Agriculture*, 145:275–281.

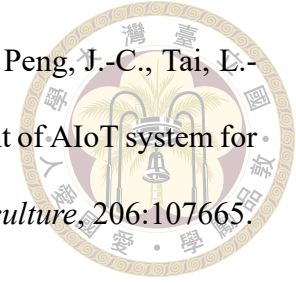
Chang, C.-L., Chung, S.-C., Fu, W.-L., and Huang, C.-C. (2021). Artificial intelligence approaches to predict growth, harvest day, and quality of lettuce (*Lactuca sativa* L.) in a IoT-enabled greenhouse system. *Biosystems Engineering*, 212:77–105.

Chang, S.-C., Zhong, Z.-P., Guo, M.-C., Hsieh, M.-H., Peng, J.-C., Tai, L.-C., Chung, P.-L., Wang, J.-C., Jiang, J.-A., and Chou, C.-Y. (2022). AIoT based pest counting system for asparagus cultivation. In *Proceedings of the 10th International Symposium on Machinery and Mechatronics for Agriculture and Biosystems Engineering (ISMAB)*.

Chautá-Mellizo, A., Campbell, S., Bonilla, M., Thaler, J., and Poveda, K. (1999). Effects of natural and artificial pollination on fruit and offspring quality. *Basic and Applied Ecology*, 13:524–532.

Chen, C.-J., Huang, Y.-Y., Li, Y.-S., Chang, C.-Y., and Huang, Y.-M. (2020). An AIoT based smart agricultural system for pests detection. *IEEE Access*, 8:180750–180761.

Chou, C.-Y., Chang, S.-C., Zhong, Z.-P., Guo, M.-C., Hsieh, M.-H., Peng, J.-C., Tai, L.-C., Chung, P.-L., Wang, J.-C., and Jiang, J.-A. (2023). Development of AIoT system for facility asparagus cultivation. *Computers and Electronics in Agriculture*, 206:107665.



Das, S., Pandey, V., and Patel, H. R. (2011). Effect of weather parameters on pest-disease of okra during summer season in middle gujarat. *Journal of Agrometeorology*, 13:38–42.

Delaplane, K. S., Mayer, D. F., et al. (2000). *Crop pollination by bees*. CABI publishing.

Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., and Sun, J. (2021). Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13733–13742.

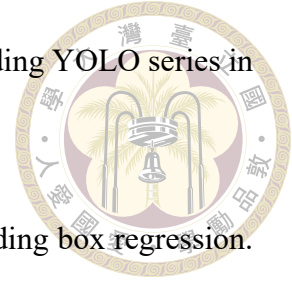
Du, Y., Wan, J., Zhao, Y., Zhang, B., Tong, Z., and Dong, J. (2021). Giaotracker: A comprehensive framework for mcmot with global information and optimizing strategies in visdrone 2021. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.*, pages 2809–2819.

Du, Y., Zhao, Z., Song, Y., Zhao, Y., Su, F., Gong, T., and Meng, H. (2023). Strongsort: Make deepsort great again. *IEEE Transactions on Multimedia*, pages 1–14.

Evangelidis, G. and Psarakis, E. (2008). Parametric image alignment using enhanced correlation coefficient maximization. *IEEE transactions on pattern analysis and machine intelligence*, 30:1858–1865.

Fisher, R. and Pomeroy, N. (1989). Pollination of greenhouse muskmelons by bumble bees (hymenoptera: Apidae). *Journal of Economic Entomology*, 82:1061–1066.

Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. (2021). YOLOX: Exceeding YOLO series in 2021. *arXiv preprint arXiv:2107.08430*.



Gevorgyan, Z. (2022). SIoU loss: More powerful learning for bounding box regression. *arXiv preprint arXiv:2205.12740*.

Guo, M.-C., Hsieh, M.-H., and Chang, C.-C. (2019). Benefit of water mist reduction of small pest density for facility asparagus. *Tainan District Agricultural News NO.107*, pages 09–12.

Guo, M.-C., Hsieh, M.-H., Chang, C.-C., and Chen, S.-H. (2017). Application of spray (water) to reduce the harm of small pests. *Tainan District Agricultural News NO.102*, pages 04–06.

Guo, M.-C., Hsieh, M.-H., Chang, W.-P., Jiang, J.-A., Chou, C.-Y., Wang, J.-C., and Chang, C.-C. (2020). Facility asparagus and thrips friendly integrated control tips. *Tainan District Agricultural News NO.113*, pages 13–16.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969.

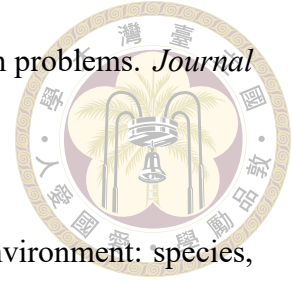
Heuvel, R. M. V. (1996). The promise of precision agriculture. *Journal of Soil and Water Conservation*, 51:38–40.

Hsieh, M.-H., Guo, M.-C., Jiang, J.-A., Chou, C.-Y., Liu, L.-Y., and Wang, J.-C. (2020). Facility asparagus friendly integrated thrip control demonstration observation session. *Tainan District Agricultural News NO.112*, pages 25–29.

Jocher, G. (2020). YOLOv5 by Ultralytics.

Jocher, G., Chaurasia, A., and Qiu, J. (2023). YOLO by Ultralytics.

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82D:35–45.



Kevan, P. G. (1999). Pollinators as bioindicators of the state of the environment: species, activity and diversity. *Agriculture, Ecosystems Environment*, 74:373–393.

Kim, M.-G., Yang, J.-Y., Chung, N.-H., and Lee, H.-S. (2012). Photo-response of tobacco whitefly, *bemisia tabaci gennadius* (hemiptera: Aleyrodidae), to light-emitting diodes. *Journal of the Korean Society for Applied Biological Chemistry*, 55:567–569.

Klein, A., Vaissière, B., Cane, J., Steffan-Dewenter, I., Cunningham, S., Kremen, C., and Tscharntke, T. (2006). Importance of pollinators in changing landscapes for world crops. In *Proceedings of the Royal Society B: Biological Sciences.*, volume 274, pages 303–313.

Kuhn, H. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2:83–97.

Kumar, A. (2016). *Weather Induced Insects/Pests/Diseases Occurrence Their Management*, pages 65–72.

Li, C., Li, L., Geng, Y., Jiang, H., Cheng, M., Zhang, B., Ke, Z., Xu, X., and Chu, X. (2023). Yolov6 v3. 0: A full-scale reloading. *arXiv preprint arXiv:2301.05586*.

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., et al. (2022). Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*.

Li, L.-C., Lin, C.-Y., Yang, C.-K., and Hsu, W.-H. (2020a). Research and analysis on

the development status of smart agriculture application and potential talent demand. *Agriculture Policy & Review*, pages 66–72.



Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J., and Yang, J. (2020b). Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *Advances in Neural Information Processing Systems*, 33:21002–21012.

Liao, M.-S., Chen, S.-F., Chou, C.-Y., Chen, H.-Y., Yeh, S.-H., Chang, Y.-C., and Jiang, J.-A. (2017). On precisely relating the growth of phalaenopsis leaves to greenhouse environmental factors by using an IoT-based monitoring system. *Computers and Electronics in Agriculture*, 136:125–139.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125.

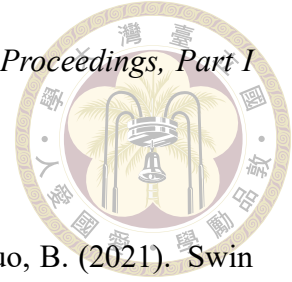
Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017b). Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988.

Liu, J. and Wang, X. (2021). Plant diseases and pests detection based on deep learning: A review. *Plant Methods*, 17.

Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. (2018). Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European*

Conference, Amsterdam, The Netherlands, October 11–14, 2016, *Proceedings, Part I* 14, pages 21–37. Springer.



Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022.

Lopez-Reyes, K., Armstrong, K. F., Van Tol, R. W., Teulon, D. A., and Bok, M. J. (2022). Colour vision in thrips (thysanoptera). *Philosophical Transactions of the Royal Society B*, 377(1862):20210282.

Loshchilov, I. and Hutter, F. (2016). SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

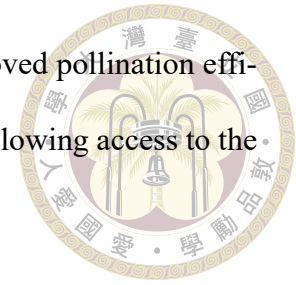
Luo, H., Jiang, W., Gu, Y., Liu, F., Liao, X., Lai, S., and Gu, J. (2019). A strong baseline and batch normalization neck for deep person re-identification. *IEEE Transactions on Multimedia*, 22:2597–2609.

Maraveas, C. and Bartzanas, T. (2021). Application of internet of things (IoT) for optimized greenhouse environments. *AgriEngineering*, 3:954–970.

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

Ngo, T. N., Wu, K.-C., Yang, E.-C., and Lin, T.-T. (2019). A real-time imaging system for multiple honey bee tracking and activity monitoring. *Computers and Electronics in Agriculture*, 163:104841.

Nicodemo, D., Malheiros, E., Jong, D., and Couto, R. (2018). Improved pollination efficiency and reduced honey bee colony decline in greenhouses by allowing access to the outside during part of the day. *Sociobiology*, 65:714–721.



Ouni, R. and Saleem, K. (2022). Framework for sustainable wireless sensor network based environmental monitoring. *sustainability*, 14(14):8356.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788.

Redmon, J. and Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

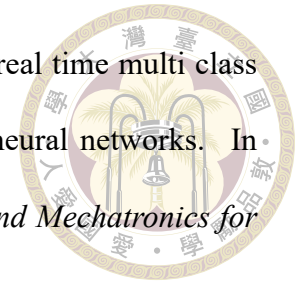
Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28:91–99.

Rupnik, R., Kukar, M., Vračar, P., Košir, D., Pevec, D., and Bosnić, Z. (2017). AgroDSS: A decision support system for agriculture and farming. *Computers and Electronics in Agriculture*, 161:260–271.

Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77:157–173.

Rustia, D. J. A., Chao, J.-J., Chiu, L.-Y., Wu, Y.-F., Chung, J.-Y., Hsu, J.-C., and Lin, T.-T. (2021). Automatic greenhouse insect pest detection and recognition based on a cascaded deep learning classification method. *Journal of Applied Entomology*, 145(3):206–222.

Rustia, D. J. A., Lin, C. E., Chung, J. Y., and Lin, T.-T. (2018). A real time multi class insect pest identification method using cascaded convolutional neural networks. In *Proceedings of the 9th International Symposium on Machinery and Mechatronics for Agriculture and Biosystems Engineering (ISMAB)*.



Rustia, D. J. A., Lin, C. E., Chung, J.-Y., Zhuang, Y.-J., and Lin, T.-T. (2020). Application of an image and environmental sensor network for automated greenhouse insect pest monitoring. *Journal of Asia-Pacific Entomology*, 23:17–28.

Rustia, D. J. A. and Lin, T.-T. (2017). An iot-based wireless imaging and sensor node system for remote greenhouse pest monitoring. *Chemical Engineering Transactions*, 58:601–606.

Sabara, H. and Winston, M. (2003). Managing honey bees (hymenoptera: Apidae) for greenhouse tomato pollination. *J Econ Entomol*, 96:574–54.

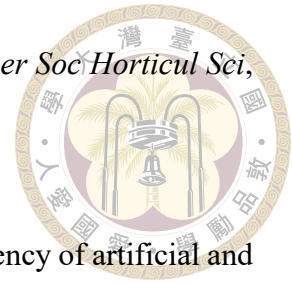
Smith, L. N. and Topin, N. (2019). Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-domain Operations Applications*, volume 11006, pages 369–386. SPIE.

Solomon, M. (1971). Insect pollination of crops by j b free london: Academic press (1970), pp. 544,£ 7.25. *Experimental Agriculture*, 7(4):367–368.

Stadler, D. and Beyerer, J. (2022). Modelling ambiguous assignments for multi-person tracking in crowds. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision.*, pages 133–142.

Sun, Q., Zhao, X., Wu, L., Zhao, J., Yang, Y., and Zhang, Y. (2021). Differences in pollination efficiency among three bee species in a greenhouse and their effects on yield

and fruit quality of northern highbush “bluecrop” blueberry. *Amer Soc Horticult Sci*, 56:603–607.



Sáez, A., Negri, P., Viel, M., and Aiden, M. (2019). Pollination efficiency of artificial and bee pollination practices in kiwifruit. *Scientia Horticulturae*, 246:1017–1021.

Tian, D., Han, Y., Wang, B., Guan, T., Gu, H., and Wei, W. (2022). Review of object instance segmentation based on deep learning. *Journal of Electronic Imaging*, 31(4):041205–041205.

Tripathy, P. K., Tripathy, A. K., Agarwal, A., and Mohanty, S. P. (2021). Mygreen: An IoT-Enabled smart greenhouse for sustainable agriculture. *IEEE Consumer Electronics Magazine*, 10:57–62.

Tzutalin (2015). Labelimg. Free Software: MIT License.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7475.

Wang, C.-Y., Liao, H.-Y. M., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., and Yeh, I.-H. (2020a). CSPNet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 390–391.

Wang, C.-Y., Yeh, I.-H., and Liao, H.-Y. M. (2021). You only learn one representation: Unified network for multiple tasks. *arXiv preprint arXiv:2105.04206*.

Wang, Z., Zheng, L., Liu, Y., Li, Y., and Wang, S. (2020b). Towards real-time multi-

object tracking. in: European conference on computer vision. In *European Conference on Computer Vision.*, pages 107–122.



Washitani, I., Kato, M., J., N., and Suzuki, K. (1994). Importance of queen bumble bees as pollinators facilitating inter-morph crossing in *primula sieboldii*. *Plant Species Biology*, 9:169–176.

Williams, C. and Rasmussen, C. (1995). Gaussian processes for regression. *Advances in Neural Information Processing Systems*, 8.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE.

Xu, Y., Wei, H., Lin, M., Deng, Y., Sheng, K., Zhang, M., Tang, F., Dong, W., Huang, F., and Xu, C. (2022). Transformers in computational visual media: A survey. *Computational Visual Media*, 8:33–62.

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

Zhang, H., Wang, Y., Dayoub, F., and Sunderhauf, N. (2021). VarifocalNet: An IoU-Aware Dense Object Detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8514–8523.

Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Lin, H., Zhang, Z., Sun, Y., He, T., Mueller, J., Manmatha, R., et al. (2022). ResNeSt: Split-Attention Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2736–2746.

Zhao, Z.-Q., Zheng, P., Xu, S.-T., and Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30:3212–3232.



Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., and Ren, D. (2020). Distance-iou loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12993–13000.

Zou, Z., Chen, K., Shi, Z., Guo, Y., and Ye, J. (2023). Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276.