ROBUST FRUIT DETECTION AND LOCALIZATION FOR ROBOTIC HARVESTING

By

Pengyu Chu

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering—Doctor of Philosophy

2023

ABSTRACT

Automated apple harvesting has attracted significant research interest in recent years due to its potential to revolutionize the apple industry, addressing the issues of shortage and high costs in labor. One key enabling technology towards automated harvesting is robust apple detection and localization, which poses great challenges because of the complex orchard environment that involves varying lighting conditions and foliage/branch occlusions. In this dissertation, I first propose a suppression Mask RCNN to generally improve the accuracy for apple detection. The developed feature suppression network significantly reduces false detection by filtering non-apple features learned from the feature learning backbone. In addition, I propose a novel deep learning-based object detection method Occluder-Occludee Relational Network (O2RNet), which addresses the challenge of detecting and isolating clustered apples in apple orchards. This was motivated by the observation that previous object detection techniques have exhibited limited success in handling fruit occlusion and clustering, which are common issues in agricultural settings. To overcome these challenges, O2RNet employs a two-stage approach, where in the first stage, I use a customized deep Feature Pyramid Network (FPN) architecture to generate candidate regions of interest (ROIs) for potential fruit objects. The second stage feeds these candidate ROIs into the occluder branch and occludee branch respectively using a feature expansion structure (FES). By leveraging this two-stage approach, O2RNet can effectively isolate individual apples from clustered regions, thereby facilitating accurate apple detection. Furthermore, I focus on developing an Active Laser-Camera Scanning (ALACS) scheme to achieve a high-precision 3D localization of detected apples and overcome existing localization challenges like varying illumination conditions, complex occlusion scenarios, and limited geometric information. The hardware of ALACS includes a red line laser, an RGB camera, and a linear motion slide. All these components are seamlessly integrated for fruit localization by using an active scanning scheme and laser-triangulation technique. The technique integrates semantic information from O2RNet's detection results with bounding boxes to generate accurate 3D coordinates for each detected

apple. Last but not least, I propose a Skeleton-lead Segmentation Network (SkeSegNet) and integrated it with the Panoptic-Deeplab. SkeSegNet is developed to address the challenges of segmenting complex branches by treating branches as a set of skeletons. Combined with depth map, SkeSegNet generates 3D branches locations for efficient obstacle avoidance. I evaluated each approach in the comprehensive experiments and superior experimental results demonstrated the effectiveness of the proposed approaches.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Zhaojian Li and my co-advisor Dr. Renfu Lu, for offering me the opportunity to join the Apple Harvesting Robot project at Michigan State University. I am very honored to have Dr. Li as my primary advisor. Dr. Li's mentorship and encouragement have made me accomplish goals that I thought impossible for myself. I truly learned a lot from his knowledge, experience and vision in not only research but also other perspectives. It is my great pleasure to have Dr. Lu as my co-advisor. The time I spent brainstorming, discussion on research ideas, polishing papers has significantly improved my skills in critical thinking, academic writing and presentation. I would like to express my gratitude to my committee members Dr. Daniel Morris and Dr. Vabhaiv Srivastava for their advice, insight and mentorship.

I am deeply thankful for the opportunity to collaborate with an exceptional group of colleagues, faculty, and researchers throughout my Ph.D. program. I would like to thank Dr. Xiaoming Liu for offering valuable advice at the beginning of my work, and Dr. Jiayu Zhou for enriching my learning experience. I am also grateful to work with Dr. Kaixiang Zhang, Kyle Lammers and Keyi Zhu. The collaborative work presented in this dissertation would not have been possible without their valuable contributions. I am grateful for my labmates from RIVAL Lab, and I will never forget the good memories of us at RIVAL Lab.

Last but certainly not least, I want to thank my friend Dr. Liang Han for his valuable advice and help at my PhD early stage. I want to thank my parents for their endless and unconditional love, faith and support that they have always given me.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION AND MOTIVATION

In this chapter, I first introduce the motivation of this thesis and the challenges for robotic fruit harvesting. Then, I introduce the specific research objectives of this thesis and provide a summary of the key contributions.

1.1 Motivation

Fruit harvesting is highly labor-intensive and cost-heavy; it is estimated that the labor needed for apple harvesting alone is more than *10 million worker hours* annually, attributing to approximately 15% of the total production cost in U.S. [37]. Growing labor shortage and rising labor cost have steadily eroded the profitability and sustainability of the fruit industry. Furthermore, manual picking activities constitute great risks of back strain and musculoskeletal pain to fruit pickers due to repetitive hand motions, awkward postures when picking fruits at high locations or deep in the canopy, and ascending and descending on ladders with heavy loads [33]. Therefore, there is an imperative need for the development of robotic mass harvesting systems to tackle labor shortage, lower human injury risks, and improve productivity and profitability of the fruit industry.

The first and foremost task in robotic harvesting is fruit detection and localization, which identifies fruit in the area of interest and provides targets for the robot to perform subsequent actions. Due to the low cost of cameras and the tremendous advances in computer vision [87], image-based fruit detection systems have gained great popularity in robotic fruit harvesting since the late 1980s. Although robust apple detection in the presence of complex tree structures and varying lighting conditions is a challenging task, the deep learning-based perception techniques [63, 49, 48, 114, 70] have enabled robotic fruit harvesting in the reality.

1.2 Challenges in Robotic Fruit Harvesting

Fruit harvesting presents a myriad of challenges, encompassing various aspects that significantly impact the efficiency and precision of the process [23].Despite progresses [55, 116,

89, 25, 132], several important challenges in developing a fully functional robotic harvesting system remain, and no commercially-viable systems are yet available in the market. One key challenge is posed by fruit clusters, where multiple fruits grow closely together, making it intricate to isolate and identify individual fruit. Occlusion, another formidable challenge, arises when fruits are hidden or partially obscured by foliage or branches and always causes failed detection. Additionally, the 3D localization of fruits under occlusion generates imprecise positioning and may lead to harvesting errors. Branch obstacles further compound these difficulties, as the harvesting robot must navigate through the complex three-dimensional structure of the plant while avoiding damage to both the fruits and the tree itself [3]. These challenges make robotic fruit harvesting harder in the complex environments.

1.3 Summary of Research Contributions

This dissertation studies detection and 3D localization of fruits in complex orchards for robotic apple harvesting application. In particular, I propose a suppression Mask R-CNN Network and an Occluder-occludee Relational Network to enhance detection precision for fruits. I also explore using a novel laser-based device to improve accuracy for 3D fruit localization. Then I study applying panoptic perception algorithms for orchard segmentation to help our harvesting robot avoid branch obstacles. This dissertation delivers the following contributions:

1. A new deep network, suppression Mask R-CNN, is proposed to remove false detections due to occlusion and increases the accuracy and robustness of apple detection.
2. A comprehensive apple dataset of 1246 images for two varieties of apple under different lighting conditions and occlusion levels are collected from two orchards during two harvesting seasons.
3. A novel Occluder-Occludee Relational Network (O2RNet) is proposed for enhanced apple detection in the presence of occlusions due to apple clusters.
4. A novel Active Laser-Camera Scanning system, consisting of a red line laser, an RGB-D camera, and a linear motion slide, is designed and developed for accurate fruit 3D

localization.

5. A Laser Line Extraction (LLE) algorithm is proposed and implemented for robust feature matching to enable stable 2D-3D transformation for ALACS.
6. An image annotation tool, PicA, is developed and open-sourced to alleviate the burden of manual annotation by leveraging the concept of superpixels and pre-trained models associated with panoptic segmentation annotation.
7. Skeleton-lead Segmentation Network (SkeSegNet) is proposed to address the challenges of segmenting complex branches, and generates 3D branches for efficient obstacle avoidance using depth map.

1.4 Thesis Organization

The remainder of this dissertation is organized as follows:

Chapter 2: Background and Related Work

This chapter presents a comprehensive introduction to fruit harvesting robots and our developed RIVAL's apple harvesting robot. Besides, I give an overview of existing works for fruit perception in harvesting robots, which give supports to our research work.

Chapter 3: Suppression Mask R-CNN for Apple Detection

This chapter shows my methods for collecting a comprehensive apple dataset for two varieties of apples with distinct yellow and red colors under different lighting conditions from the real orchard environment. A novel suppression Mask R-CNN was developed to robustly detect apples from the dataset. Our developed feature suppression network significantly reduced false detection by filtering non-apple features learned from the feature learning backbone. Our suppression Mask R-CNN demonstrated superior performance, compared to state-of-the-art models in experimental evaluations.

Chapter 4: O2RNet: Occluder-occludee Relational Network for Clustered Apple Detection

In this chapter, I discuss the challenges associated with the detection of clustered apples, a task that demands heightened precision and adaptability due to the complex spatial

arrangements of fruit clusters within orchards. I propose a novel solution, the Occluder-occludee Relational Network (O2RNet), designed to address the specific challenge posed by occlusion and cluster proximity. Subsequently, I provide a comprehensive evaluation of the OR2Net's performance, assessing its efficacy in real orchard environments and under varying conditions.

Chapter 5: ALACS: Active Laser-Camera Scanning for 3D Apple Localization

In this chapter, I review several consumer RGB-D cameras and depict our proposed localization technique, called Active Laser-Camera Scanner (ALACS). I propose a feature-matching algorithm, called Laser Line Extraction (LLE), to help ALACS transform the 2D fruit positions to 3D fruit positions, thus achieving accurate mapping even under complex fruit morphology, variable lighting conditions and occlusions.

Chapter 6: SkeSegNet: Skeleton-lead Segmentation Network for Branch Segmentation

In this chapter, I delve into the realm of panoptic segmentation, presenting a comprehensive overview of the field. I introduce our novel approach, SkeSegNet, aimed at advancing branch segmentation by leveraging skeletal information for enhanced accuracy and robustness. Furthermore, I explore the by-product of SkeSegNet to generate 3D branch representations.

Chapter 7: Conclusion and Future Work

This chapter concludes the thesis and discusses the future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, I provide a comprehensive introduction to fruit harvesting robots and our developed RIVAL's apple harvesting robot. Besides, I give an overview of existing works for fruit perception in harvesting robots, which give supports to our research work.

2.1 Introduction

The apple industry relies heavily on manual labor. For instance, in the United States alone, it is estimated that the seasonal labor force needed for apple harvesting is more than 10 million worker hours each year, attributing to about 15% of the total production costs [38]. The growing labor shortage and increased labor cost have thus become major concerns for the long-term sustainability and profitability of the apple industry. In the meantime, the past decade has seen great transitions in apple production systems; traditional unstructured orchards have been replaced with high-density orchard systems where trees are smaller and more uniformly structured (i.e., v-trellis, vertical fruiting wall, etc.). These modern tree structures can greatly facilitate orchard automation, and thus there has been a renewed interest in pursuing robotic harvesting as a promising solution to reduce the harvesting cost and dependence on manual labor.

Over the past few years, several robotic systems have been designed to autonomously harvest different horticultural crops, including sweet pepper [64], strawberry [123], apple [100], and kiwifruit [119]. For apple harvesting, the automation system designs can be mainly grouped into two categories. The first category is the *shake-and-catch* harvesting [135], where vibrations are applied to the tree trunk and/or branches to detach the fruits. Although the shake-and-catch harvesting systems are efficient in detaching fruits from trees, they often result in a high rate of apple bruising that is not acceptable for fresh market. The other category is the *fruit-by-fruit harvesting* where manipulators are used to pick fruits in a controlled manner, and thus can substantially reduce fruit damage. However, designing such systems with high picking efficiency and practical viability presents a great challenge.

So far, several fruit-by-fruit robotic apple harvesting systems have been developed [4, 100, 50, 133, 15]. For instance, Baeton et al. combines a 7 degree-of-freedom (DOF) industrial manipulator with a vacuum activated, funnel shaped gripper for apple detachment, and the harvesting cycle time is 8-10 s/fruit [4]. In [100], both hardware and software designs of an apple harvester are presented. Field tests conducted on a v-trellis orchard show that this system is able to pick 84% of 150 apples attempted with the overall harvesting time being 7.6 s/fruit. In [50], Hohimer et al. developed a harvesting robot based on a pneumatic soft-robotic end-effector, and the average time that the system takes from apple detachment to transported to storage bin is 7.3 s/fruit. Despite the aforementioned progresses, the low picking efficiencies of existing systems are still unsatisfactory for their practical use in the real orchard environment [73].

2.2 RIVAL's Apple Harvesting Robot

Despite the significant progress, the existing robotic apple harvesting systems are still far from being commercially viable mainly because they are unreliable in performance and inefficient or too slow in picking fruit in the real orchard environment and are too complicated or expensive to be economically sound [43]. Hence, I designed a high-performance robot platform [134] for automated fruit-by-fruit apple harvesting.

The principal objective of the hardware system design is to build a fully self-contained, modular harvesting platform that can support various payloads, endure rough terrains, and be easily modifiable and extensible. To facilitate the movement in the orchard environment, the whole system is built on a trailer base which can be hauled by a farm vehicle or robotic moving platform. The developed robotic apple harvesting system is shown in Figure 2.1, which consists of three major modules: the support hardware module, the computer & operation module, and the robotic harvester.

The support hardware module includes a 5.5 kW Honda gas-powered electric generator and a Delfin industrial vacuum. The generator provides a 240 V power source and can continuously support the whole system running for approximately 5.5 hours if fully refilled.

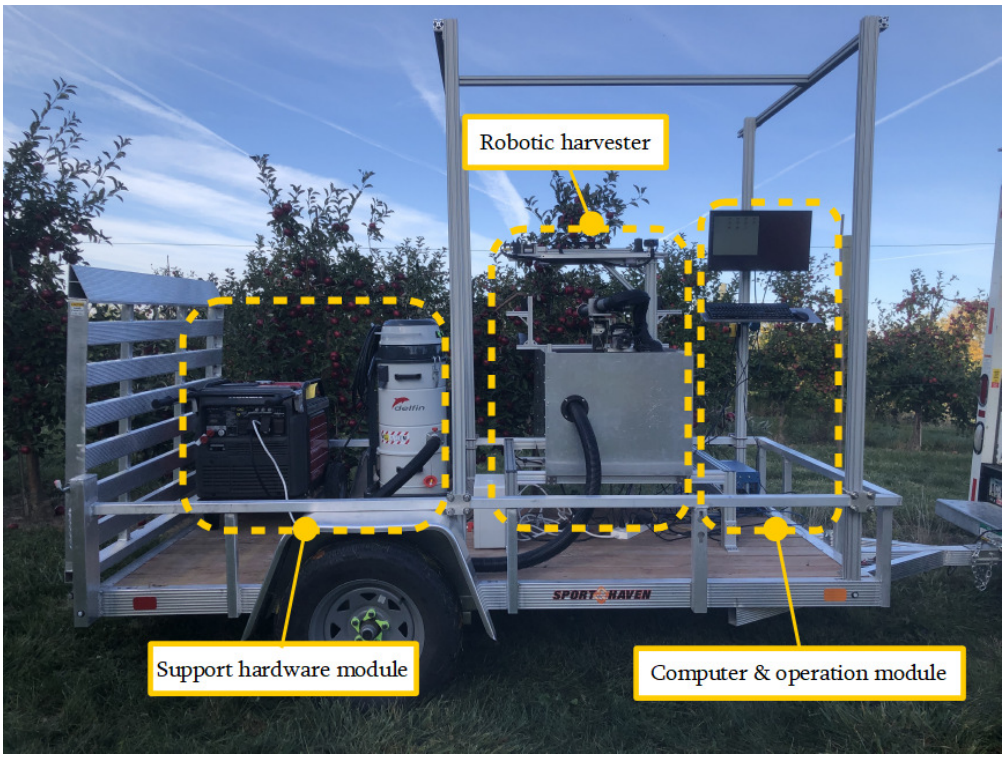

Figure 2.1 Hardware modules of the developed robotic apple harvesting system.

The Delfin industrial vacuum has two powerful bypass motors with independent cooling and can generate a peak horsepower of 5.5 HP. During fruit harvesting, this vacuum machine runs continuously to provide vacuum flow, generating suction forces to enable the soft effector to detach fruits (see Section 2.2.3 for descriptions on the end-effector).

The computer & operation module is comprised of a high-performance industrial computer and a workstation where users can monitor and control the robotic system. The industrial computer has an Intel®Xeon E2176G processor, 64 GB of RAM, and a NVIDIA GeForce RTX 2080 Ti graphic processing unit. This computer hosts all software algorithms and the communication connections to all components.

The main components of the robotic apple harvesting system are shown in Figure 2.2, including a perception component, a 4-DOF manipulator, a vacuum-based soft end-effector, and a dropping component. More detailed descriptions on these four components are given in the following sections.

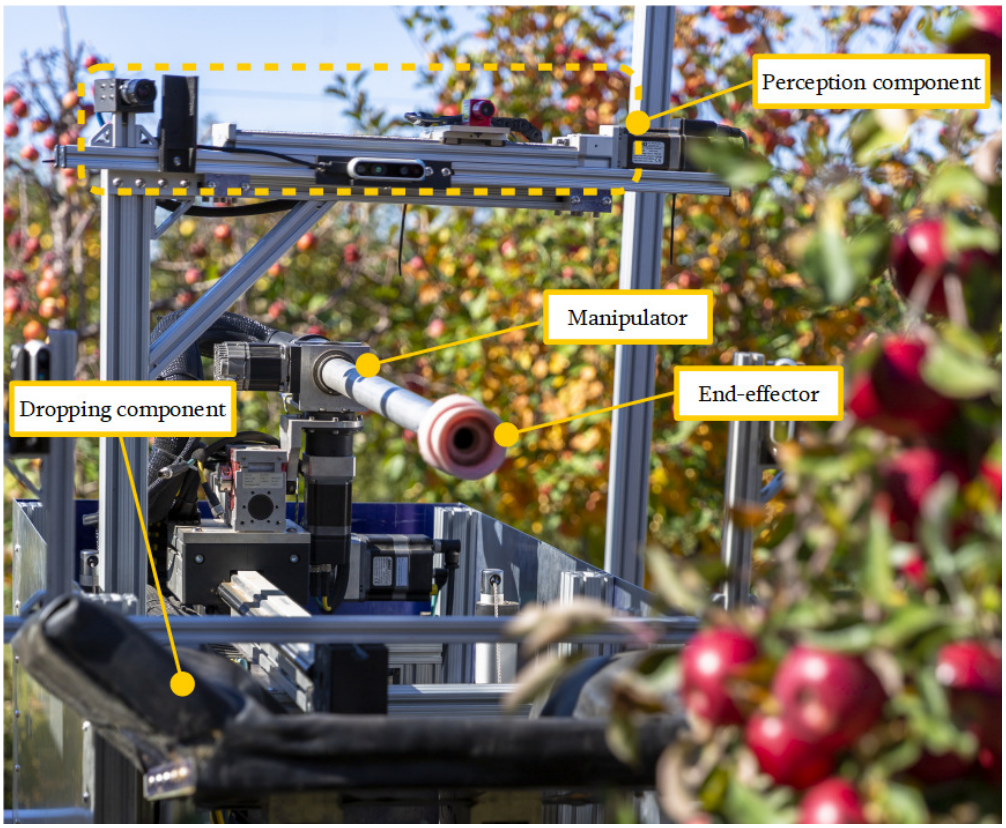

Figure 2.2 Main components of the robotic harvester for apple picking.

2.2.1 Perception Component

For robotic apple harvesting, the first and foremost task is to detect and localize the fruits. As shown in Figure 2.2, an Intel RealSense D435i RGB-D camera and a custom-built laser-camera unit are integrated as the sensor set to achieve apple detection and localization. The RGB-D camera is mounted on a horizontal frame that is above the manipulator. Different from the other robotic harvesting systems (e.g., [**?**]) that attach the camera to the manipulator or the end-effector, our installation scheme ensures that the RGB-D camera can provide a global view of the scene, which facilitates the use of multiple manipulators planned in our future versions. Since the depth measurement of consumer RGB-D camera is not stable under leaf/branch occlusions and/or challenging lighting conditions (see [79, 35] for an in-depth review on localization performance of commercial RGB-D sensors), I design a new laser-camera unit to address this issue. The specially designed laser-camera unit is comprised of a red line laser (635 nm), a Flir RGB camera, and a linear motion slide. The

line laser is mounted on top of the linear motion slide which enables the laser to move back and forth horizontally with a full stroke of 20 cm. Meanwhile, the Flir RGB camera is installed at the rear end of the linear motion slide with a relative angle to the laser scanner. The RGB-D camera and the laser-camera unit are fused synergistically to achieve high perception accuracy and robustness, where the fusion scheme will be detailed in Chapter 5.

2.2.2 Manipulator

A 4-DOF manipulator is designed and assembled with compact mechanical structure for efficient manipulation in the workspace. As illustrated in Figure 2.3, the manipulator consists of three revolute joints and one prismatic joint. The first and second revolute joints are connected by an L-shaped aluminum plate to form a pan-and-tilt mechanism. The prismatic joint is used as the base of the pan-and-tilt mechanism to extend the manipulator's workspace. A hollow aluminum link is installed on the pan-and-tilt mechanism to enable the end-effector to reach the apples and serve as a vacuum tube for grasping the fruits during the harvesting process. The third revolute joint is assembled at the rear end of the aluminum tube to create a rotation mechanism. After the end-effector has grasped the fruit, this rotation mechanism is triggered to rotate the aluminum tube to detach the fruit from the tree. In addition, all the joints are driven by servo motors instead of using a hybrid pneumatic/motor actuation mechanism as in our previous design [133], which simplifies the actuation and facilitates an integrated control scheme design.

Different from most existing studies [? ? ? ?] that rely on high DOF industrial manipulators, the developed 4-DOF manipulator is simple and compact in structure and highly efficient in picking fruit. This hardware design allows convenient development of adaptable control algorithms such that agile manipulation can be achieved. It also facilitates future extensions of multiple robotic arms for coordinated multi-arm apple harvesting.

2.2.3 End-Effector

In our robotic system, a vacuum-based soft end-effector is used to grasp and detach fruits. The end-effector is a vacuum cup made of silicone rubber and is attached to the

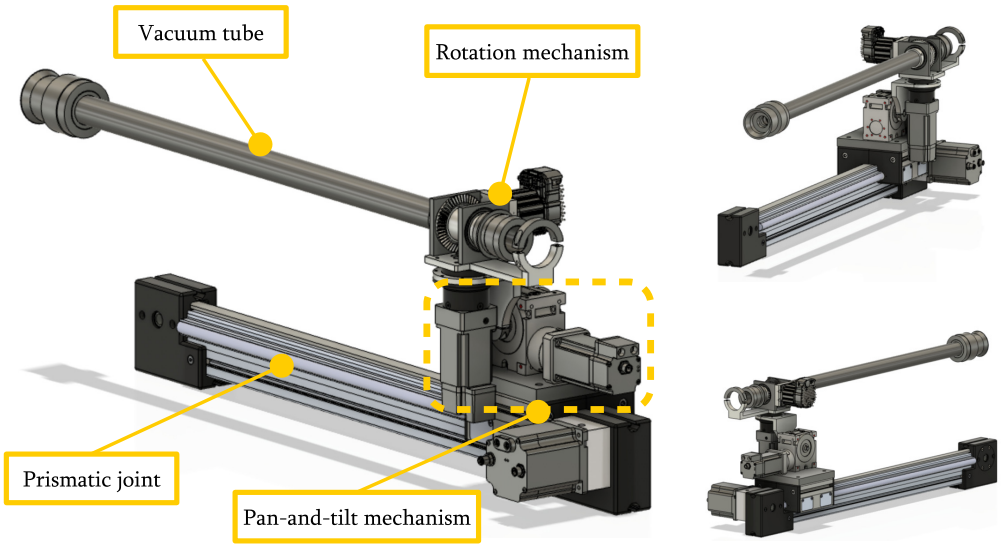

Figure 2.3 CAD model of the 4-DOF manipulator.

front end of the aluminum tube (i.e., the front end of the manipulator). Through laboratory experiments, a silicone material with a hardness of 40 Shore A and a cup shape geometric design shown in Figure 2.4 are selected for the end-effector. Compared with our previous designs [72], the current cup shape geometric design has a smaller inner diameter and a larger outer-lip. The silicone material and the improved cup's geometry allow for conformity to the fruit contours to generate a sufficient suction force needed for holding and detaching the fruit, while also being flexible or deformable to minimize or eliminate fruit bruising. The end-effector is securely connected to the aluminum tube through an adaptor and the rear end of the aluminum tube is connected to the Delfin industrial vacuum via a flexible and expandable tube. One major advantage of using the vacuum-based end-effector is that it can tolerate some approaching inaccuracy; it is able to attract fruits within a distance of about 1.5 cm when operated under the current vacuum flow. This allows the manipulator to grasp the fruit even if it does not approach the fruit accurately.

2.2.4 Dropping/Catching Component

A dropping/catching component is assembled and attached to the robot platform to enable faster release or dropping of the picked fruit. The base of the dropping component is made up of a rectangular aluminum plate covered with a soft foam cushion of 50 mm in

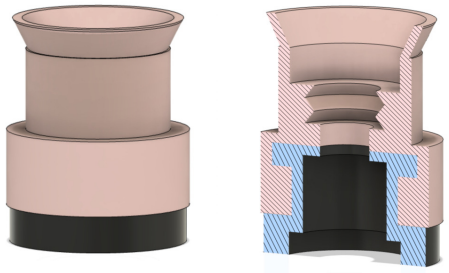

Figure 2.4 CAD model of the vacuum-based soft end-effector.

thickness. Laboratory tests showed that the foam cushion allows apples to drop from the highest position of the end-effector (approximately 80 cm) without causing fruit bruising, while keeping fruit bouncing to minimum. The manipulator can drop the picked apples at any spots from above the dropping component without fully returning to its home position, thus reducing the overall fruit picking cycle time. After an apple has fallen onto the sloped surface of the dropping component, it rolls down to a screw-driver conveyor, which transports the apple to the destination or a bin [136].

2.2.5 Software Design

The software suite is designed within the robot operating system (ROS) framework. Different software components are primarily communicated via custom messages sent through ROS actions and services. Figure 2.5 shows the main logic/algorithm flow of the software system during apple harvesting. It is apparent that the software design of our robotic system requires multi-disciplinary advances to enable various synergistic functionalities and coordination for achieving reliable automated apple harvesting. The logic flow of the apple harvesting cycle is detailed in the following.

At the beginning of each harvesting cycle, the RGB-D camera is triggered to acquire images at 30 fps. Based on the obtained image information, deep learning and active camera laser scanning are exploited to detect and localize the fruits within the manipulator's workspace. A list of 3-dimensional (3D) apple locations will be generated, and then the one on top of the list, by following the pre-defined criteria, will be selected as the target fruit. Since location results provided by the RGB-D camera might not be sufficiently accurate,

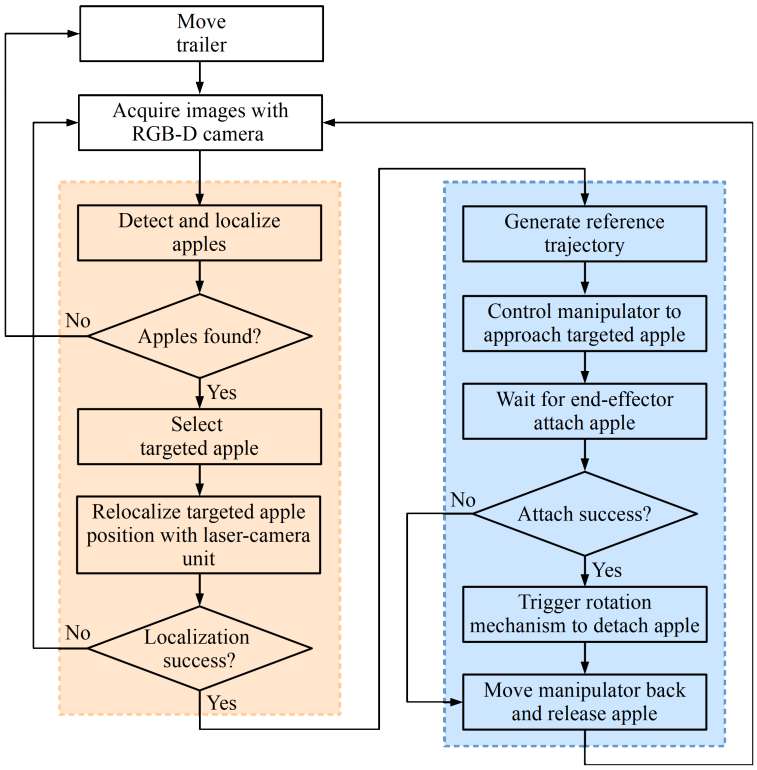

Figure 2.5 Logic flowchart in apple harvesting.

the developed laser-camera unit and corresponding perception scheme are triggered to scan the target fruit and calculate its 3D position. Given the ameliorative target apple location, the planning algorithm is used to generate a reference trajectory, and the control module will actuate the manipulator to follow this reference trajectory to reach the fruit. Once the fruit is successfully attached to the end-effector (detected by a pressure sensor mounted inside the tube), the rotation mechanism is triggered to rotate the whole aluminum tube by a certain angle, and the manipulator then retracts to pull and detach the apple (if the rotation action has not resulted in complete detachment of the fruit). After the manipulator reaches a pre-determined dropping spot, a vacuum control valve installed between the outlet of the vacuum machine and the inlet of the flexible vacuum hose actuated, which causes rapid loss of vacuum pressure in the tube, thus enabling the fruit to fall off the end-effector by gravity to the drop‘ping component. The fruit finally rolls down the slope of the dropping component to the screw conveyor, from which the apple is transported to the destination.

2.2.6 Field Tests Environment

To fully evaluate the performance of the developed apple detection (see Chapter 3, 4) and localization (see Chapter 5) algorithoms, field tests were conducted in two Michigan State University's research orchards in East Lansing and Holt, Michigan, USA, respectively, during the 2021, 2022, and 2023 harvest season (Figure 2.6). The first orchard had been planted with 'Gala' apple trees of two years old, which is a popular bicolored variety with a red color on the foreground and a yellow background. There were fewer fruits grown on these young trees with less occlusions by branches and foliage, but many of the fruits were grown in clusters (Figure 2.6(b)). In the second orchard were 'Ida Red' apple trees of seven years old (Figure 2.6(c)). Since the trees had not been pruned and thinned during the winter and early spring seasons, there were dense and unstructured foliage and branches. A high percentage of apples were grown in clusters and occluded by leaves and branches, which presented a significantly more challenging environment for our robot. It should be mentioned that most reported studies tested their robotic apple harvesting systems in high-density tree orchards with well-trained tree architectures with few clustered apples and less dense foliage (either naturally or being removed).

2.3 Existing Works in Perception of Fruit Harvesting

Fruit perception is one of the key functionalities in robotic harvesting. Several research groups have been developing robotic harvesting systems [55, 116, 89, 25]. Despite progresses, several important challenges in developing a fully functional robotic harvesting system remain, and no commercially-viable systems are yet available in the market. One key challenge that is pointed out by the existing works is efficient and robust fruit detection in the presence of varying light conditions and fruit/foliage occlusions. Indeed, the perception system provides the robot system with information on target fruits, which are first and foremost for subsequent planning and control tasks. In addition, fruit perception techniques have also been used in other applications of interest, including yield estimation and crop health status monitoring [86]. Perception in unstructured orchard environments, however, is a daunting

Figure 2.6 Field trails of the robotic apple harvesting system. (a) Image of the platform operating in the orchard environment; (b) Example images of young and well-pruned trees in the first orchard; and (c) Example images of older trees with dense foliage in the second orchard.

task as a result of variations in illumination and appearance, noisy backgrounds, and cluttered environments with occlusions [23]. The goal of this paper is thus to present a novel deep learning-based detection algorithm to convergently address the aforementioned challenges. I show that the developed algorithm is able to achieve state-of-the-art performance. Before describing the technical details, I review relevant backgrounds and state-of-the-art approaches to put our algorithm in better context.

2.3.1 Image Sensing Techniques

Vision-based perception schemes can be classified into four categories based on the sensor used: monocular camera scheme, binocular stereovision scheme, laser active visual scheme, and thermal imaging scheme, which cover both two-dimension imaging schemes and three-dimension imaging schemes [138]. Specifically, the monocular scheme uses a single camera to acquire image data, and it is widely used in fruit harvesting due to its low cost and

rich information provided by the RGB images. For instance, [109] developed an improved YOLOv3 [93] model based on a single camera to detect apples with an accuracy of 85.0%. In [56], the authors proposed a new LedNet model for apple detection that achieves an accuracy of 85.3%. The main disadvantage of the monocular scheme is that the color images are sensitive to fluctuating illumination.

Different from the monocular camera schemes, the binocular stereovision schemes exploit two cameras separated in a certain distance/angle to obtain two image data on the same scene. The point cloud of fruit can then be constructed through triangulation on extracted features [106]. For instance, [99] used a stereo camera to detect and localize mature apples in tree canopies, and achieved an accuracy of 89.5%. In [122], the authors developed a clustered tomato detection method based on a stereo camera, and the recognition accuracy was 87.9%. Although the stereovision scheme tends to render better results, it suffers from high complexity, long computation time, and uncertainties in stereo matching [46].

On the other hand, the laser active visual schemes obtain three-dimensional features using laser scans, where laser beam reflections are exploited to generate a 3D point cloud based on the time-of-flight principle. The 3D point cloud can then be used to reconstruct the scene. For example, [108] utilized infrared laser scanning devices to recognize cherry on the tree. [129] acquired a total of 200 images for independent 'Fuji' apples and developed an apple recognition method using the near-infrared linear-array structured light for 3D reconstruction. [113] proposed a point cloud based apple detection method using a LiDAR laser scanner and reached a 88.2% overall accuracy on the defoliated tree dataset [113]. Note the defoliated scene is significantly less challenging than the real orchard conditions during the harvest season. Furthermore, the laser point cloud is generally sparse and it is challenging to be used in real-world orchards with dense backgrounds. The high cost and complexity also limit its practical application in agricultural applications.

Finally, the thermal imaging schemes make use of the distinct thermal characteristics of fruit and leaves (e.g., the different temperature distributions) to obtain the visualization of

infrared radiation [71]. In [13], citruses are successfully segmented using a thermal infrared camera according to the largest temperature difference in both day and night conditions. An enhanced approach for fruit detection [14] was developed using the combination of the thermal image and the color image. The results showed a promising performance under weak lighting environments. However, in the thermal imaging scheme, the accuracy of recognition is largely affected by the shadow of the tree canopy [104].

Considering the cost, performance, and real-time constraints, our work focuses on the monocular camera scheme, the state-of-art of which will be discussed next.

2.3.2 Recognition Approaches

Image-based fruit recognition approaches can be classified into feature analysis approaches and *deep learning-based* approaches, depending on how features are obtained. In *feature analysis* approaches, hand-crafted features are first extracted based on the fruit characteristics, and classification approaches are then developed to recognize fruit. [103, 102] developed thresholding methods to classify fruit from other background objects using smoothing filters that remove irrelevant noises. The large segmented regions are then recognized as fruits. This method is capable of segmenting fruit regions in simple backgrounds but it is susceptible to varying lighting conditions and complex canopies. [118, 10] proposed a circular Hough Transform approach to obtain binary edge images and then used a voting matrix to identify fruits. This approach is sensitive to complex structured environments and it generally fails in a dense scene. In [90, 16, 65, 137], they combined the shape and texture of the fruit to obtain a richer set of feature representations. Then, extracted features between fruit and leaves are compared and contrasted to identify the fruits. However, this method is also sensitive to lighting conditions and occlusions.

On the other hand, deep learning-based approaches have found great successes in object detection and semantic image segmentation [97, 9]. They can learn feature representations automatically without the need of manual feature engineering. Compared to conventional methods, Convolutional Neural Networks (CNNs) have been showing great advantages in

the field of object detection in recent years. The CNN makes it possible to recognize fruits in complex situations due to its deep extraction of high-dimensional features of objects. R-CNN and its variants Fast R-CNN and Faster R-CNN [42, 41, 94] have enjoyed particular successes. Their key idea is to first obtain regions of interest and then perform classification in the region. The Region proposal network (RPN) is employed to reduce high computational costs so that the model can simultaneously predict and classify object boundaries at each location. The parameters of the two networks are shared, which results in much faster inference and are thus optimized for real-time purposes. Faster Region-Based CNN, proposed by [97], employed transfer learning using ImageNet, and used both early fusion and late fusion to integrate RGB and NIR (near infrared) inputs. Modified Inception-ResNet (MI-ResNet) [91] used deep simulated learning for yield estimation. The model was developed to address challenges including the varying degree of fruit sizes and overlap, natural lighting, and foliage occlusions. The overhead for object detection and localization is optimized by utilizing synthetic data for training, and reaching the accuracy of 91% on their fruit dataset. You Only Look Once (YOLO) [93], a representative of the one-stage object detector, detects the fruit on the entire image and classifies fruit variety into uncertainty retail conditions without the help of RPN. Specifically, YOLO uses logistic regression to predict an objectless score for each bounding box. Due to the simple optimization pipeline, YOLO enjoys much faster inference than the aforementioned region-based methods. EfficientDet [107], an augmented variant of YOLO, exploits a pyramid network to enable the detection of scaling targets.

2.4 Summary of the Chapter

In this chapter I first introduce the background of fruit harvesting robots. Based on the existing robots, I give our own solutions, called RIVAL's Apple Harvesting Robot, to address fruit harvesting challenges in apple orchards. I present the hardware and software development of the robotic apple harvesting system, which is not only compact in structure design but also effective in utilizing multi-disciplinary advances to enable synergistic harvesting functionalities. The perception strategy design is the basic of subsequent chapters

(Chapter 3, 4, 5, 6).

Additionally, I discuss the related works in perception techniques and approaches in fruit harvesting, including different sensors and vision-based methods. RGB-D cameras have been a popular sensor choice for fruit detection and localization. Although consumer RGB-D cameras are compact and can provide dense image and depth information, the depth measurements by these RGB-D cameras are not robust under varying lighting conditions or when apples are partially occluded by foliage, which would result in inaccurate fruit localization and eventually degrades the harvesting performance of the robotic system.

CHAPTER 3

SUPPRESSION MASK R-CNN FOR APPLE DETECTION

In this chapter, I showed my methods for collecting a comprehensive apple dataset for two varieties of apples with distinct yellow and red colors under different lighting conditions from the real orchard environment. A novel suppression Mask R-CNN [23] was developed to robustly detect apples from the dataset. Our developed feature suppression network significantly reduced false detection by filtering non-apple features learned from the feature learning backbone. Our suppression Mask R-CNN demonstrated superior performance, compared to state-of-the-art models in experimental evaluations.

3.1 Introduction

The accurate detection of fruits is of paramount importance in the realm of fruit harvesting, serving as a foundational element for the efficiency, precision, and overall success of automated harvesting systems [8]. Fruit detection plays a pivotal role in overcoming the inherent challenges [44] associated with the diverse and complex environments within orchards and fields. Several state-of-the-art deep learning-based apple detection approaches have been developed. In particular, DaSNet [55], a deep convolutional neural network that exploits the techniques of spatial pyramid pooling and gate feature pyramid network, is proposed for apple detection. It uses a lightweight residual network as its backbone to achieve improved computational efficiency. Although DaSNet has a decent performance (0.832 F1-score) and a lightweight overhead, the algorithm is only trained and validated on a dataset that contains a single apple variety with good lighting. YOLOv3 [93], another lightweight network that combines Region Proposal Network (RPN) and classification network into a single architecture, is applied in [109] for apple detection. While the network offers a fast detection rate, it has a relatively low F1-score of 0.817. Mask R-CNN [48], a popular object detection algorithm, is also deployed for apple detection [52]. The Mask R-CNN is a two-stage detector that involves a RPN and a classification network. The former searches the location of region of interest (ROI), whereas the latter predicts the class of ROI and

regresses the bounding box of the ROI candidates. The Mask R-CNN is successfully applied to apple detection in [52] with promising performance demonstrated. However, the dataset they use only has one apple variety with good lighting conditions, making the results less compelling. Despite the aforementioned developments, accurate apple perception to support robotic harvesting in real orchard environments remains a great challenge. Existing methods either provide insufficient accuracy [93, 55] or are based on simple structured orchards with little occlusion and stable lighting conditions [28, 12].

In this chapter, I use a comprehensive orchard database that contains multiple apple varieties under various lighting conditions. Further, I develop a novel Suppression Mask R-CNN that has superior performance as compared to the aforementioned approaches. The contributions of this work are summarized as follows:

1. I collect and process a comprehensive orchard dataset with multiple apple varieties under various lighting conditions in real orchard environment.
2. I develop a new deep network, suppression Mask R-CNN, to remove false detections due to occlusion and thus increase the accuracy and robustness of apple detection.
3. Extensive evaluations show that the proposed suppression Mask R-CNN achieves state-of-the-art performance.

3.2 Data Collection and Processing

In this study, apple images of 'Gala' and 'Blondee' varieties were taken in two commercial orchards in Sparta, Michigan, USA during the 2019 harvest season. The two apple varieties have distinct color characteristics; 'Gala' apples are red over a yellow background, while 'Blondee' apples have a smooth yellow skin (see Figure 3.1). An RGB camera with a resolution of $1,280$x720 was used to take images of apples at a distance of $1 \sim 2$ meters to the tree trunk, which is the typical range of harvesting robots [25]. The images were collected across multiple days to cover both cloudy and sunny weather conditions. In a single day, the data were also collected at different times of the day, including 9:00am in the morning, noon, and 3:00pm in the afternoon, to cover different lighting angles: front-lighting, back-lighting,

side-lighting, and scattered lighting. When capturing images, the camera was placed parallel to the ground and directly facing the trees to mimic the harvesting scenario. A total of $1,500$ images were captured where two sample images are shown in Figure 3.1.

I next processed the collected raw orchard images into formats that can be used to train and evaluate deep networks. Specifically, apples in the images were annotated by rectangles using VGG Image Annotator [32] and the annotation was then compiled into the human-readable format. Compared to polygon and mask annotations, rectangular annotation used here accelerates data preparation, particularly in dense images like our dataset. The annotated dataset was then split into training, validation, and test subsets with the apple quantities of $10,530$, $4,203$, and $4,795$, respectively.

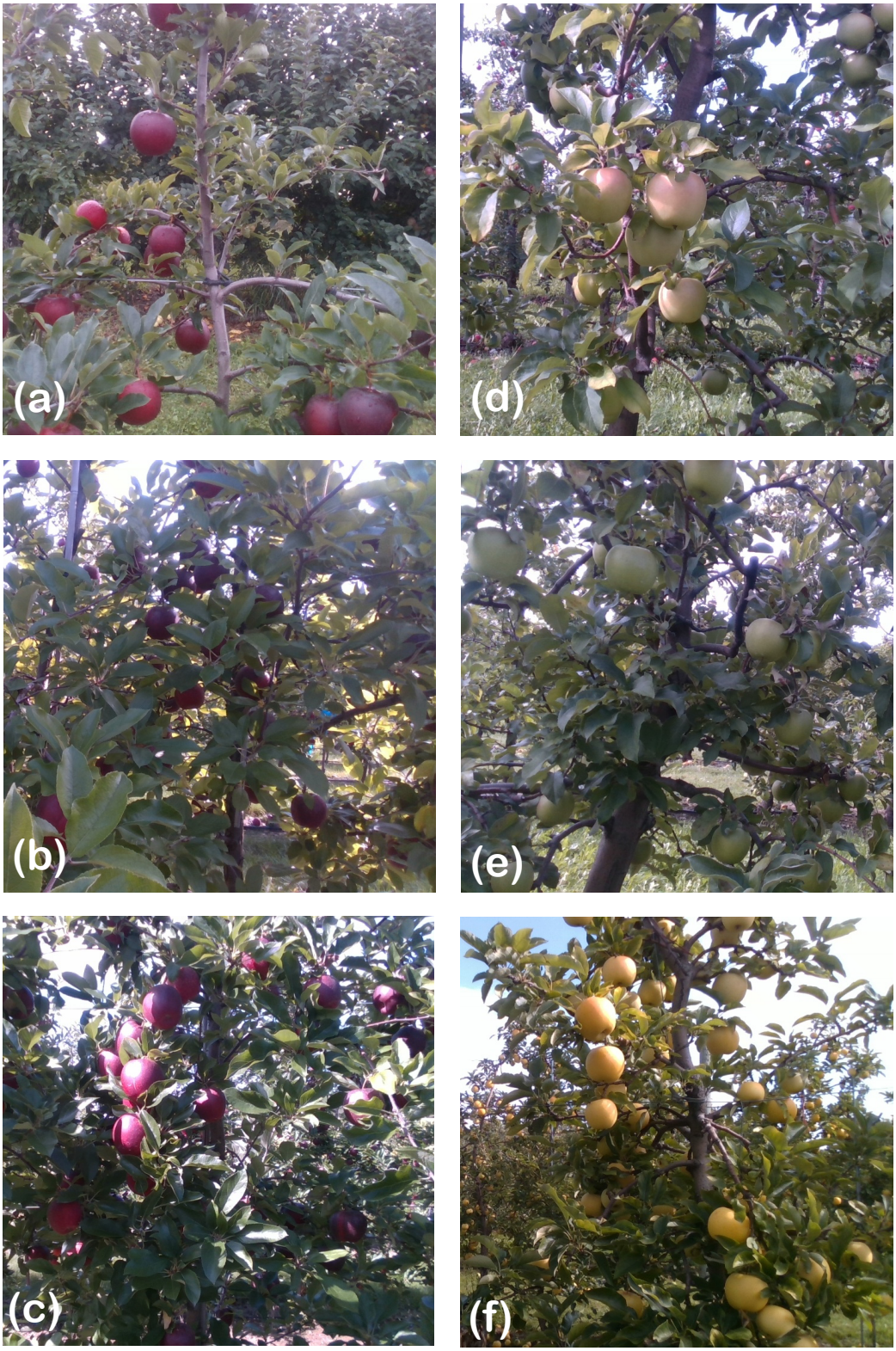

Figure 3.1 Six sample images from the collected dataset: (a)-(c) apples on older trees under overcast, back-lighting, and direct lighting conditions, respectively; and (d)-(e) apples on younger trees under overcast, back-lighting, and direct lighting conditions, respectively.

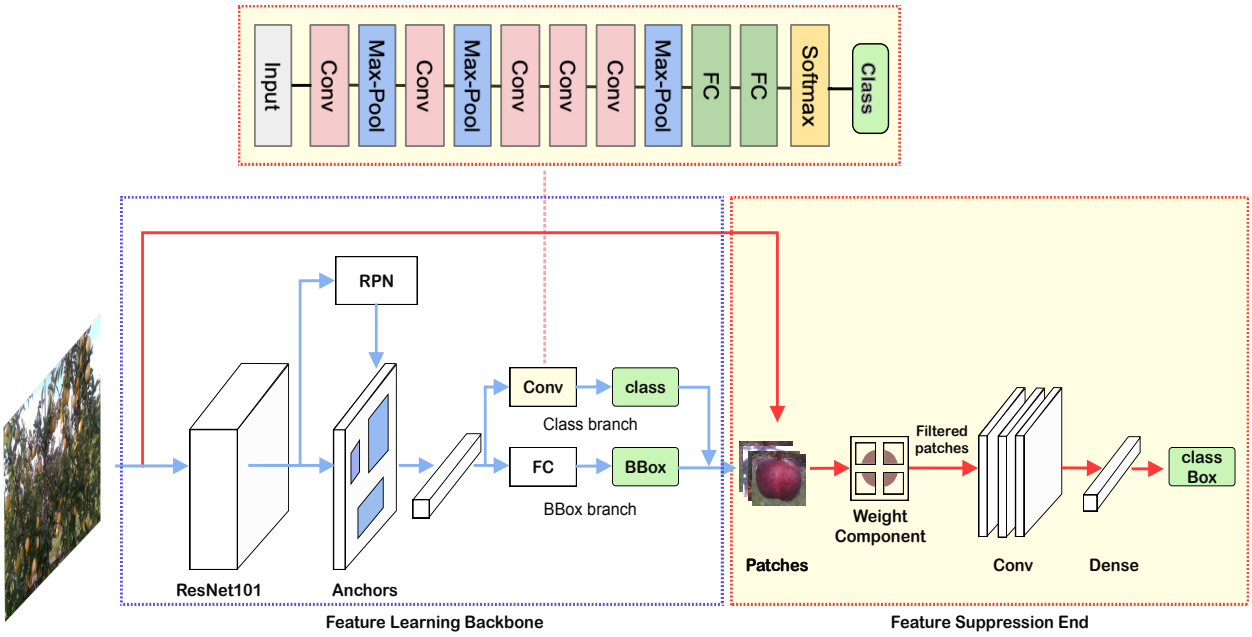

Figure 3.2 Structure of the suppression Mask R-CNN. It consists of a feature learning backbone and a feature suppression end. The feature learning backbone is a deep network to learn apple features while the feature suppression end, consisting of a weighting component and a shallow ConvNet, is used to filter non-apple regions.

3.3 Suppression Mask R-CNN

This section describes the development of a new deep learning-based apple detection approach that systematically combines a DNN backbone and an RGB feature-based suppression network. As shown in Figure 3.2, the proposed suppression Mask R-CNN consists of two parts: a feature learning backbone from Mask R-CNN [48] and a feature suppression end. The former is used to learn apple features and generate region proposals. In the meantime, due to the foliage and branch occlusions, it will also learn foliage and branch features that can cause false detection. As such, I introduce a suppression network to filter non-apple features to improve detection performance by exploiting a combination of clustered features and convoluted features. These two networks are trained separately to avoid generating similar feature maps. I next discuss the two networks in more details.

3.3.1 Feature Learning Backbone

The feature learning network uses the Mask R-CNN backbone [48] and follows Mask R-CNN's two-stage learning procedures with two modifications. First, the convolutional

backbone in Mask R-CNN is used for feature extraction over an entire image, and is applied as the network backbone for bounding-box recognition. In this study, I instantiate feature learning backbone with ResNet-101-FPN [48] as its backbone. ResNet101 outperforms other single ConvNet mainly because it maintains strong semantic features at various resolution scales. Even though ResNet101 is a deep network, the residual blocks and dropouts function help it avoid gradient vanishing and exploding problems. Then similar to [48], I use a Region Proposal Network (RPN) [94] to generate object regions. RPN is a small convolutional network which can convert feature maps into scored region proposals around where the object lies. These proposals with certain height and width are called anchors, which are a set of predefined bounding boxes. The anchors are designed to capture the scale and aspect ratio of specific object classes and are typically determined based on object sizes in the dataset. In the second stage, class and box offset are predicted by virtue of Faster R-CNN [94] that applies bounding box classification and regression in parallel. As shown in Figure 3.2, another network is employed to take the proposed regions from the first stage and assign them to specific areas of a feature map obtained at the second stage. After scanning these areas, the network generates object classes and bounding boxes simultaneously [48].

Second, for improving the recall or true detection of our algorithm, I introduce a convolutional structure (as shown in Figure 3.2) in the class branch to learn additional feature representations. The features condensed from the Mask R-CNN backbone and fully connected layers may have lost considerable details of apples. Since images have many occlusions in our dataset, the deep network can treat some partial foliage features as apple features. These additional feature representations will enable the identification of certain regions in an image as an occluded apple or foliage. Furthermore, I freeze the layers in the ResNet101 backbone and train this class branch independently in case there are many overlaps compared to our main network.

3.3.2 Feature Suppression End

After the feature learning step, bounding boxes of apple candidates are obtained. The image patches inside the bounding boxes are then fed into a feature suppression end to remove mis-labeled candidates. Since the feature learning backbone may have learned wrong inference features like leaves with apple-like shapes, the purpose of this suppression network is to avoid that non-apple regions flow into the last decision layer.

Specifically, the suppression network consists of a weighting component and a shallow ConvNet. The weighting component is a 2x2 grid clustering layer that aims to determine apple regions in terms of apple pixel counts. The motivation is that in our annotated dataset, each apple is annotated in the center of a bounding box and occupies the major area in that bounding box. Even though the canopies always partially occlude the apple, the pixels corresponding to the apple are still in the majority. Based on our observation of dataset, the four regions (a, b, c, d as shown in Figure 3.3-(3)) generally contain most apple pixels. Therefore, as shown in Figure 3.3, I divide each bounding box in the training dataset into four regions, a, b, c, d, as a 2x2 grid. The four regions a, b, c, d is, respectively, located near the left top, right top, left bottom, and right bottom with a margin of 5% pixels to the box edges. Furthermore, I use K-means clustering [60] to group similar pixels and obtain several clusters. After clustering, I label each pixel with its class number i, $i = 1, 2, 3, ...n$, with n being the pre-specified cluster numbers (In our experiments, I use $n = 3$). Since the class associated with the most pixels will correspond to the apple region, I select the "apple" region from the four grids and define its pixel counts as N^a, N^b, N^c, and N^d, respectively. I will then set the apple region pixels as 1 whereas other pixels are assigned to zero. A sample output is shown in Figure 3.3. The weighting component keeps the objective information and generates an output with only apple pixels, which makes it more efficient to train feature suppression network that I will discuss later. The other merit of weighting component is that if the previous network recognizes a leaf as an apple, only leaf pixels are treated as objectives and flow to next ConvNets. That makes suppression network easy to discriminate apple and

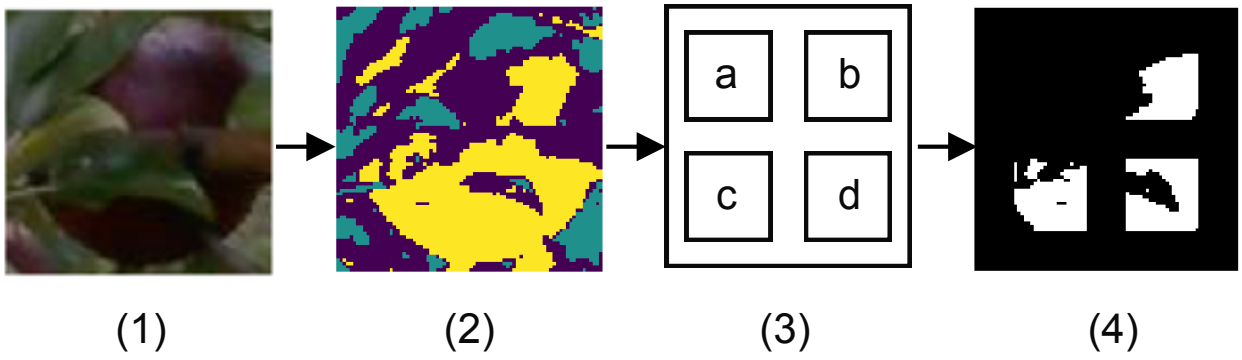

Figure 3.3 Illustration of the proposed weighting scheme: (1) sliced image inside the bounding box of a detected apple; (2) pixel clustering using K-means with $k = 3$ where each cluster is shown in one of the three colors; (3) image partitioning into 4 regions and counting pixel numbers of each cluster in the 4 grids; and (4) apple pixel determination by assigning the pixels corresponding to the cluster with most pixel counts in the 4 grids as apple pixels.

non-apple objectives.

The second component is a shallow convolutional network that is used to learn apple features based on filtered patches generated by the weighting component. Compared to the feature learning backbone, the features to learn in this shallow network is less. Only three convolution layers (3x3x32, 3x3x32, 3x3x64) associated with pooling layers (17x17x32, 7x7x32, 2x2x64) and ReLU as activation are used to fit the discrimination function. Two additional dense layers are employed to flatten feature maps and produce decision. This network has a total of $45,153$ trainable parameters. The detailed architecture is described in Figure 3.2. With the help of feature suppression end, I suppress non-apple class flowing into the decision layer and it does not significantly increase inference time since the depth of the feature suppression end is small. The proposed feature suppression end can be viewed as a filter to efficiently reduce false alarms.

3.3.3 Loss Functions

Since I train the feature learning backbone and the suppression network separately, I define two loss functions as follows. For the feature learning backbone, I use the same loss function with Mask R-CNN [48], which defines a multi-task loss on each sampled region of interest as $L_{backbone} = L_{cls} + L_{box}$, where L_{cls} and L_{box} are, respectively, classification loss

and bounding box loss defined as:

$$L_{backbone} = \frac{1}{N_{cls}} \Sigma_i L_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{box}} \Sigma_i p_i^* \cdot L_{box}(t_i, t_i^*), \tag{3.1}$$

$$L_{cls}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*) \log(1 - p_i), \tag{3.2}$$

where p_i and p_i^* are, respectively, the predicted probability and ground truth of anchor i; t_i and t_i^* are, respectively, predicted coordinates and ground-truth coordinates; N_{cls} and N_{box} are normalization terms of batch size and number of anchor locations; the loss function L_{box} is the L1-smooth function [41]; and λ is a parameter that controls the balance between the classification loss and the bounding box loss [117]. In our network, I use $\lambda = 1$ as I assign equal weights to the two losses.

For feature suppression end, I define L_{end} as the average binary cross-entropy loss. For a patch associated with ground-truth class, L_{end} is defined as:

$$L_{end} = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})], \tag{3.3}$$

where y is the ground truth and $\hat{y}$ is the prediction.

3.4 Experimental Results

3.4.1 Implementation

In this section, I evaluate the efficacy of the suppression Mask R-CNN with the processed data as discussed in Section 3.2. The network hyper-parameters, including the momentum, learning rate, decay factor, training steps, and batch size, are set as 0.9, 0.001, 0.0005, 934, and 1, respectively, through cross-validation. The input image size is $1,280$x720, which is aligned with the camera resolution. To better analyze the training process, I set up 100 epochs for training. I exploit a pre-trained model on COCO dataset [67] to warm-start the training process and it generally only needs 50 epochs to converge. A detection example is shown in Figure 3.4, where green boxes represent correctly identified apples while red boxes represent missed detection.

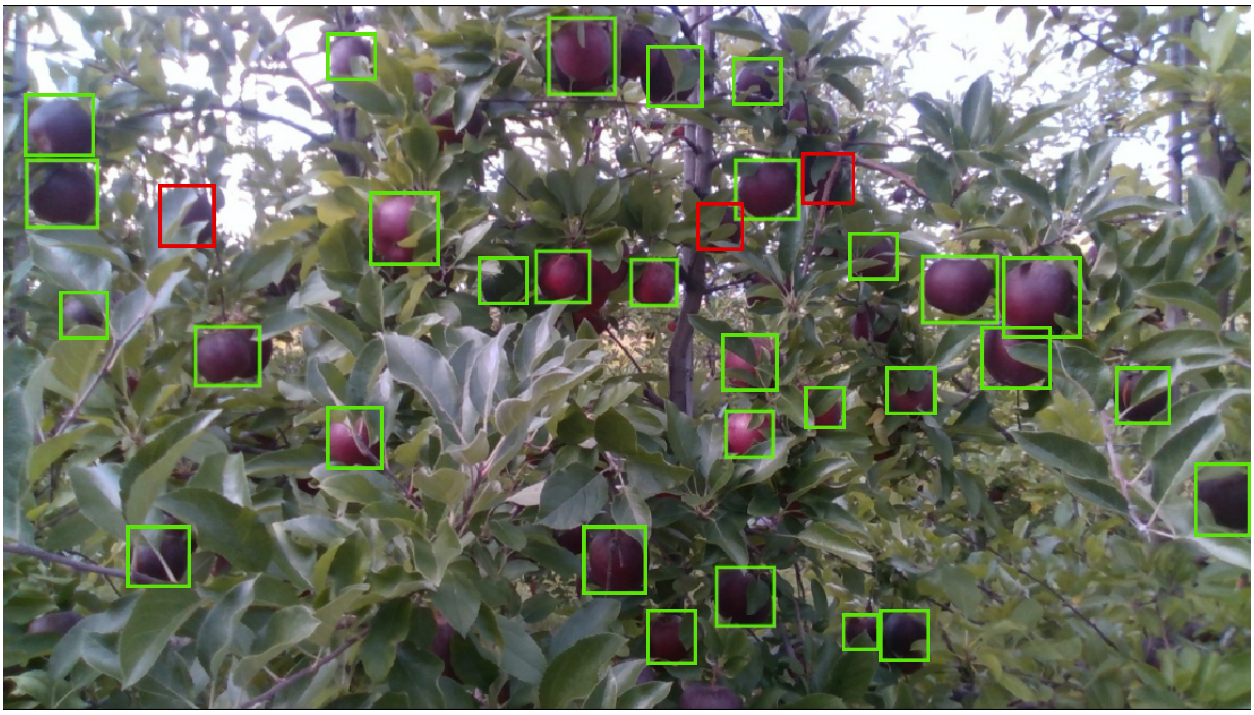

Figure 3.4 An example of Gala apple detection using our suppression Mask R-CNN. It shows that the majority of apples are detected (green bounding boxes) but there are still 3 apples missed (red bounding boxes) due to heavy occlusion.

To quantitatively evaluate the detection performance, I use performance metrics including precision, recall and F1-score for algorithm evaluation. All detection outcomes are divided into four types: true positive (TP), false positive (FP), true negative (TN), and false negative (FN), based on the relation between the true class and predicted class. Then precision (P) and recall (R) are defined as follows:

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}. \tag{3.4}$$

Then F1-score is defined based on precision and recall as follows:

$$F1 = \frac{2 \cdot P \cdot R}{P + R}. \tag{3.5}$$

Note that the suppression network offers a tradeoff between recall and precision, that is, aggressive suppression will lead to higher precision but lower recall rate. This tradeoff can be controlled by adjusting two confidence thresholds th_1 in the class branch network

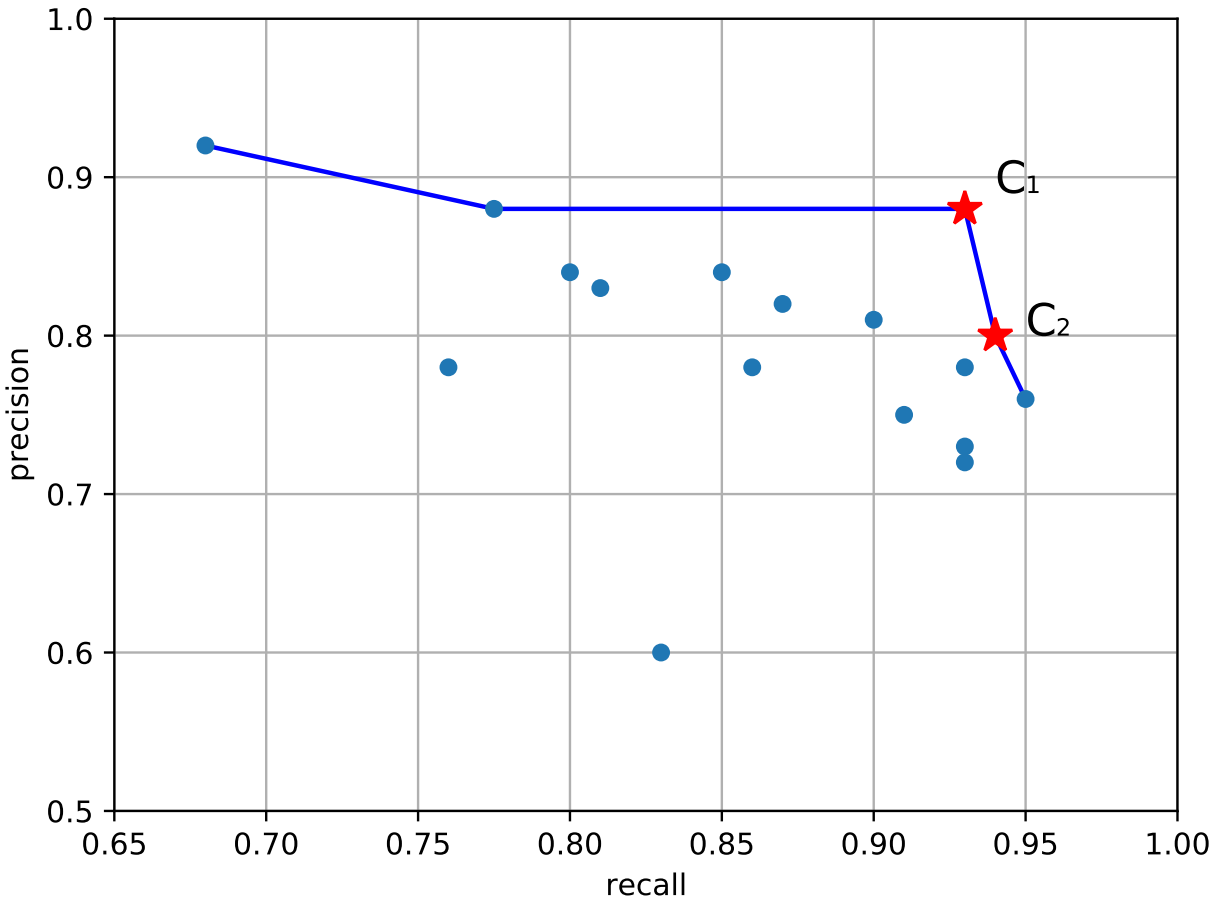

Figure 3.5 The Pareto plot of recall-precision on different combinations of th_1 and th_2.The Pareto front is shown in blue solid lines and the two configurations used to compare with the state-of-art networks (see Table 3.1) are shown in red stars.

and th_2 in the feature suppression end. Then I tune both confidence thresholds during the inference process to obtain the best recall and precision of our entire model. Figure 3.5 shows the Pareto plot, where each point represents the performance of a combination of th_1 and th_2. From the Pareto front (blue solid lines) in Figure 3.5, I choose two "best" configurations C_1 and C_2, among which C_1 represents a better F1-score 0.905 whereas C_2 achieves a better of recall rate of 0.939. The detection performance with C_1 has 10% increase in precision and 0.4% increase in recall whereas 1.6% increase in precision and 1.3% increase in recall are achieved with configuration C_2. These results demonstrate that in both cases, our integrated class branch and the suppression end approach improve the true detection and the C_2 configuration significantly reduce false fruit detection rates.

3.4.2 Comparison with the state-of-the-arts

In order to fully evaluate the performance of the proposed approach, I compare our approach with the state-of-the-art apple detection algorithms based on our comprehensive image dataset. The algorithms that I compare with include *YOLOv3* [109], *DaSNet* [55], *Faster R-CNN* [116], and *Mask R-CNN* [52]. These approaches are trained and evaluated on the same training data and test data. For Mask R-CNN, I consider two configurations: ResNet101 backbone and ResNet152 backbone. The recall-precision curves of these approaches are shown in Figure 3.6. Furthermore, the precision, recall and F1-score are shown in Table 3.1. It can be seen in Figure 3.6 and Table 3.1 that the proposed Suppression Mask R-CNN has superior performance as compared to the existing approaches.

Table 3.1 Performance comparison between the state-of-the-art networks and our proposed Suppression Mask R-CNN with two parameter configurations (C_1 and C_2).

	Precision	Recall	F1-score
YOLOv3	0.703	0.860	0.773
DaSNet	0.693	0.821	0.751
Faster R-CNN	0.761	0.889	0.820
Mask R-CNN(ResNet101)	0.789	0.927	0.852
Mask R-CNN(ResNet152)	0.798	0.928	0.858
Suppression Mask R-CNN(C_1)	**0.880**	0.931	**0.905**
Suppression Mask R-CNN(C_2)	0.801	**0.939**	0.864

3.4.3 Evaluation on different apple varieties and lighting conditions

In addition, I also evaluate my model in different sub-datasets. Specifically, I separate the whole dataset into several sub-datasets based on apple variety and lighting conditions. The evaluations are summarized in the Table 3.2 and results are shown in 3.7. The results show that my model has a better performance for Blondee apples than for Gala. Compared to back lighting conditions, the detection of my model reaches a higher precision under overcast or direct lighting conditions, which indicates that artificial lighting may be helpful for further improving the performance and it will be investigated in our future work.

Table 3.2 Performance evaluation on subset of the data with different apple varieties as well as different lighting conditions.It can be seen that similar performance are obtained in Gala and Blondee apples while back lighting can slightly decrease the performance.

	Dataset Category		Dataset Lighting Condition			Total
	Gala	Blondee	Overcast	Direct Lighting	Back Lighting	
Number	3,357	1,438	3,356	959	480	4,795
Precision	.87	.89	.89	.89	.84	.88
Recall	.93	.93	.93	.93	.93	.93
F1-score	.90	.91	.91	.91	.88	.91

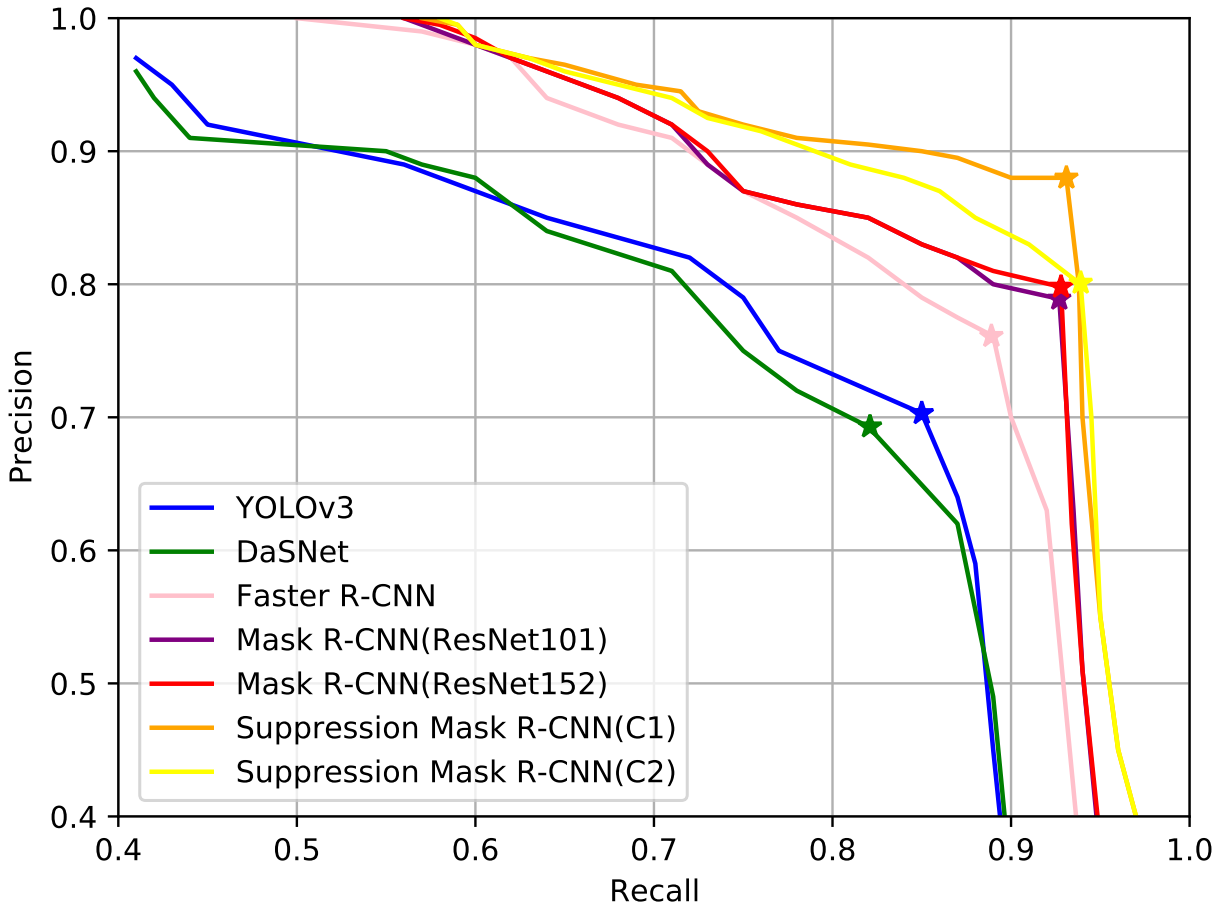

Figure 3.6 The plot of recall-precision curves on different approaches. Our proposed Suppression Mask R-CNN networks (in both configurations) outperform the state-of-the-art algorithms.

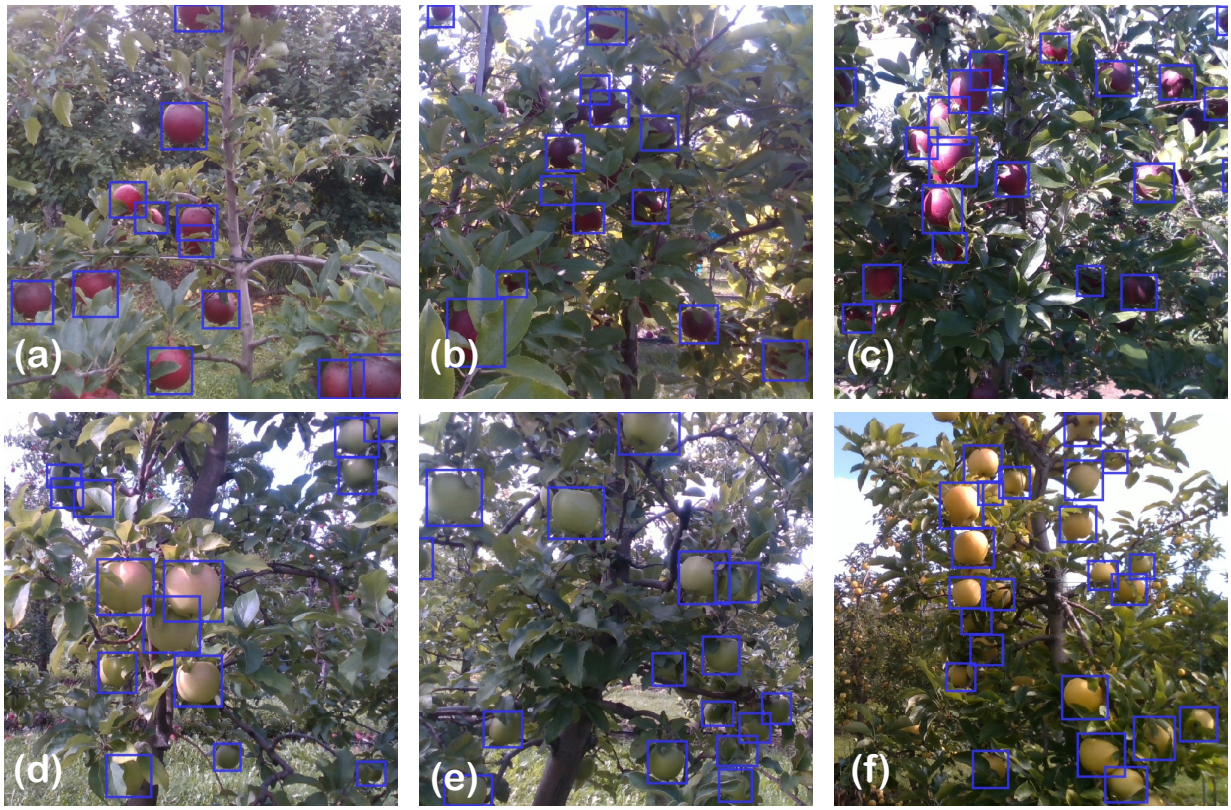

Figure 3.7 Detection results on different apple varieties under various lighting conditions: (a)-(c) detection on Gala apples under overcast, back lighting, and direct lighting conditions, respectively; and (d)-(e) detection on Blondee apples under overcast, back lighting, and direct lighting conditions, respectively.

3.5 Summary of the Chapter

In this chapter, I collected a comprehensive apple dataset for two varieties of apples with distinct yellow and red colors under different lighting conditions from the real orchard environment. A novel suppression Mask R-CNN was developed to robustly detect apples from the dataset. Our developed feature suppression network significantly reduced false detection by filtering non-apple features learned from the feature learning backbone. Our suppression Mask R-CNN demonstrated superior performance, compared to state-of-the-art models in experimental evaluations.

CHAPTER 4

O2RNET: OCCLUDER-OCCLUDEE RELATIONAL NETWORK FOR CLUSTERED APPLE DETECTION

In this chapter, I discuss the challenges associated with the detection of clustered apples, a task that demands heightened precision and adaptability due to the complex spatial arrangements of fruit clusters within orchards. I propose a novel solution, the Occluder-occludee Relational Network (O2RNet) [24], designed to address the specific challenge posed by occlusion and cluster proximity. Subsequently, I provide a comprehensive evaluation of the OR2Net's performance, assessing its efficacy in real orchard environments and under varying conditions.

4.1 Introduction

In the previous chapter, I have discussed detection methods for improving apple detection precision and a number of research works have been developing deep learning-based approaches [109, 120, 75, 115]. However, the aforementioned deep CNN approaches do not address the challenge of overlapping/clustered fruits in real-world orchards. Towards that end, Compositional Convolutional Neural Network (CompNet) [59] was proposed to detect partially occluded objects. The framework exploits a differentiable fully compositional model that uses occluder kernels to localize occluders (the occluding objects). Bilayer Convolutional Network (BCNet) [57], another model to address the occlusion challenge, applies two Graph Convolutional Network (GCN) layers to separately infer the occluding objects (occluder) and partially occluded instance (occludee). Superior performance was reported on occluded scenarios. In apple detection, various approaches are developed to enhance the performance of deep learning-based models in complex orchards. [47] introduced CBL (Convolutional layers, Batch normalization, Leaky-relu activation function [31]) module and CA (coordinate attention) module into YOLOv5 [54], and finally increased 4.41% in the precision compared to the base model. [124] utlized a customized YOLOv3 to reach a recall of 93.4% for overlapped apples. These two approaches are trying to extract the higher-level

features by modifying models to improve the performance. Different from above, [36] took advantage of a depth filter to remove background trees with an RGB-D camera and finally improved apple detection precision by 2.5% on overlapped apples. This paper will model the relationships among overlapped apples and enhance the apple edge features to improve the precision for clustered apples.

In this chapter, I develop a novel Occluder-Occludee Relational Network (O2RNet) to enhance apple detection in the presence of occlusions in clustered apples that are frequently present in real-world orchards. Specifically, I employ ResNet [49] and RPN [94] to extract features of targets and utilize occluder-occludee layers to split candidates into occluder and occludee. Compared to other occlusion models, I only use bounding boxes as labels instead of pixel-level masks that contain more texture and shape information. In addition, I present a new apple dataset[1] collected in two Michigan apple orchards in multiple harvesting seasons. I evaluate the performance against state-of-the-art object detection models and demonstrate superior performances. The contributions of this work are highlighted as follows:

1. A comprehensive apple dataset of 1246 images for two varieties of apple under different lighting conditions and occlusion levels were collected from two orchards during two harvesting seasons.
2. A novel Occluder-Occludee Relational Network (O2RNet) was developed for enhanced apple detection in the presence of occlusions due to apple clusters.
3. The O2RNet outperformed 12 state-of-the-art deep learning-based models for apple detection.

4.2 Data Augmentation

After the same method for data collection and preparation, I apply data augmentation on our dataset. Data augmentation is a method that can be adopted to increase data diversity for achieving robust training and enhanced performance of computer vision models. For example, transformations and rotations are frequently employed to increase the number

[1]The database is open-sourced at https://github.com/pengyuchu/MSUAppleDatasetv2.git.

of images from a single source. It has been shown to be a powerful tool in agriculture applications [120, 105, 29] as it generates additional data from existing orchard data. This is especially useful for applications with a limited dataset by detecting anomalies in images with different transformations and making it possible to generate new training examples without actually acquiring new data.

Specifically, in the considered application of apple detection in orchards, the collected dataset can only cover a limited set of scenarios. Therefore, I applied several data augmentation techniques [22] on the collected and processed data to enhance the data diversity for improving the inference performance of my models. Specifically, besides geometric transformations including scaling, translating, rotating, reflecting, and shearing, I also applied color space augmentations such as modifying the brightness and contrast to fit different intensities. In addition, I injected Gaussian noises on the collected images by randomly modifying the pixel intensities based on a Gaussian distribution. Furthermore, I applied Mixup by randomly selecting two images from the dataset and blending the intensities of the corresponding voxels of the two images [74]. Filtering is another augmentation approach I applied where I modify the intensities of each pixel using convolution [98]. Specifically, I exploited sharpening [98] to detect and intensify the edges of objects found within the image. I applied these additional augmentation techniques on our dataset and the benefits of data augmentation will be demonstrated in the experiment section.

4.3 O2RNet for Apple Detection

In this section, I first present the key challenges of object detection in cluttered environments and an overview of the general object detection framework. Based on those, I describe the proposed Occluder-Occludee Relational Network (O2RNet) with explicit occluder-occludee relation modeling. Finally, I specify the objective functions for the entire network optimization, followed by details on the training and inference processes.

4.3.1 Challenge and Main Idea

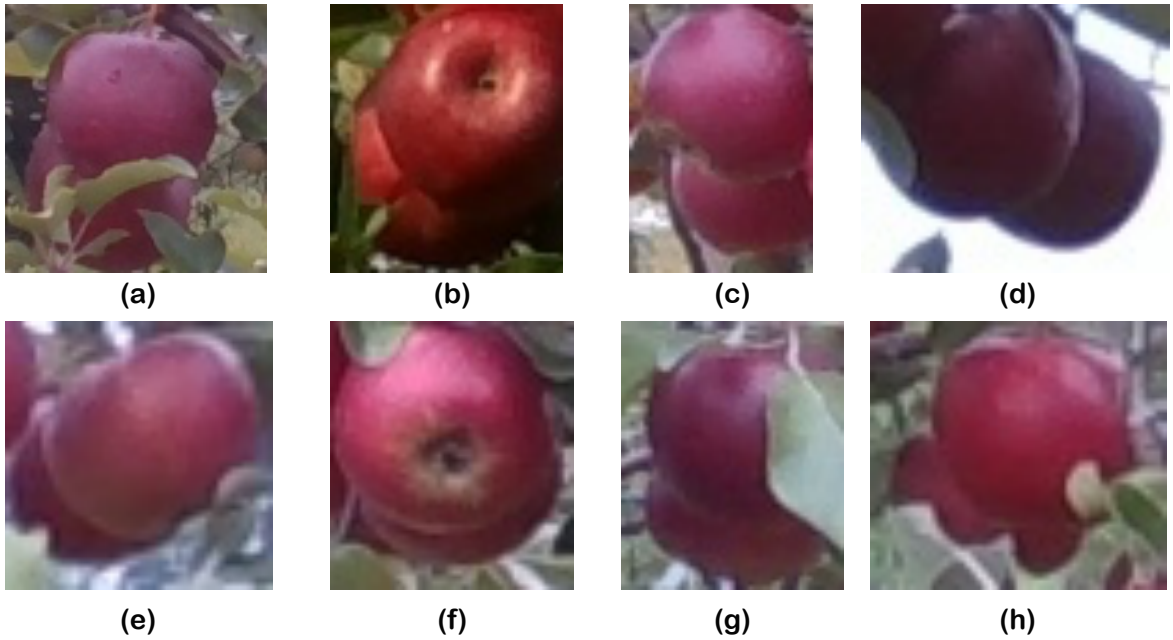

Figure 4.1 Eight sample images from the collected dataset show cascaded apples in different occlusion levels: (a)-(d) apples are in the normal occlusion and can be identified in most models; (e)-(h) apples are highly cascaded and usually detected as one apple.

For images with heavy occlusions, multiple overlapping objects captured in the same bounding box can result in confusing object outlines from both front objects and occlusion boundaries. In apple orchards, the apple clusters are very common (see Figure 4.1 for a few examples). However, the prediction head design of Faster R-CNN directly regresses the occludee with a fully convolutional network, which neglects both the occluding instances and the overlapping relations between objects. With this limitation, Faster R-CNNs will inevitably omit some occludes due to Non-maximum Suppression (NMS). On the other hand, with a properly tuned threshold, the RPN can propose many candidates after feeding the target features from CNN (see Figure 4.2), but the NMS will suppress the nearby bounding boxes and neglect occludees. Motivated by this observation, the proposed O2RNet aims at extending the existing two-stage object detection methods by adding an occlusion perception branch parallel to the original object prediction pipeline. By explicitly modeling the rela-

tionship between occluder and occludee, the interactions between objects within the Region of Interest (RoI) region can be well incorporated during the bounding box regression stage.

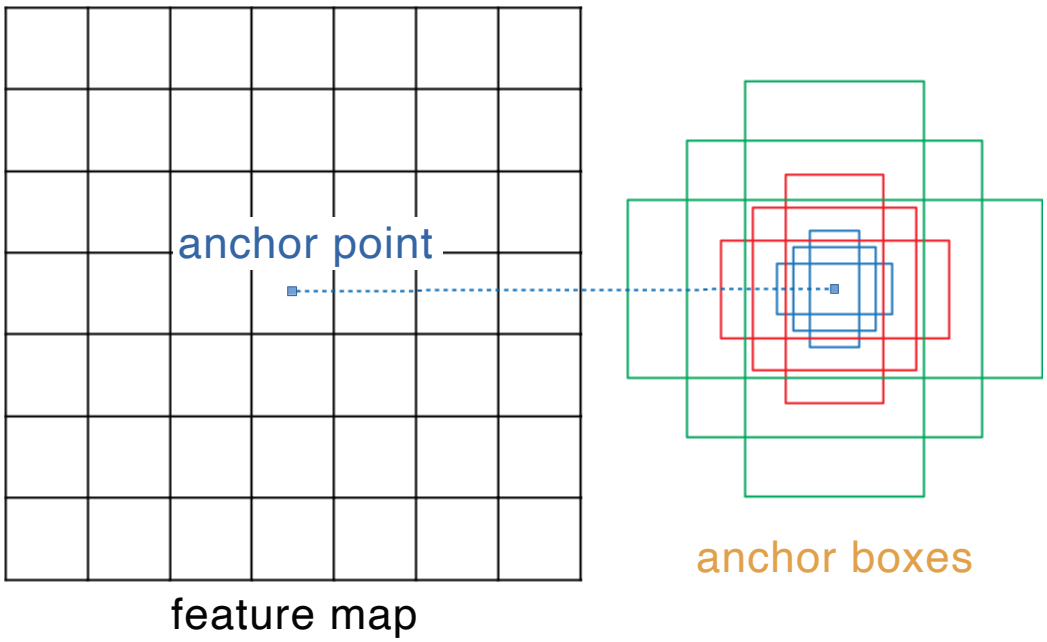

Figure 4.2 Illustration of how RPN works: The RPN selects anchor points on the feature map and generates anchor boxes for each anchor point. The anchor boxes are generated based on two parameters — scales and aspect ratios.

4.3.2 O2RNet Workflow

In this subsection, I describe our proposed O2RNet. As illustrated in Figure 4.3, the O2RNet follows the two-stage architecture used in Faster R-CNN [94] and consists of three main parts. First, I use a Residual Network (ResNet) [49] as the backbone for feature learning/extraction over the entire image. Specifically, I instantiate ResNet-101-FPN [48] as its backbone for feature extraction, as it outperforms other single ConvNets mainly due to its capability of maintaining strong semantic features at various resolution scales. Even though ResNet101 is a deep network, the residual blocks and dropouts function help it avoid gradient vanishing and exploding problems. Second, I employ an RPN [94] to generate object regions, which is a small convolutional network to convert feature maps into scored region proposals around where the object lies. The generated proposals with a certain height and width are

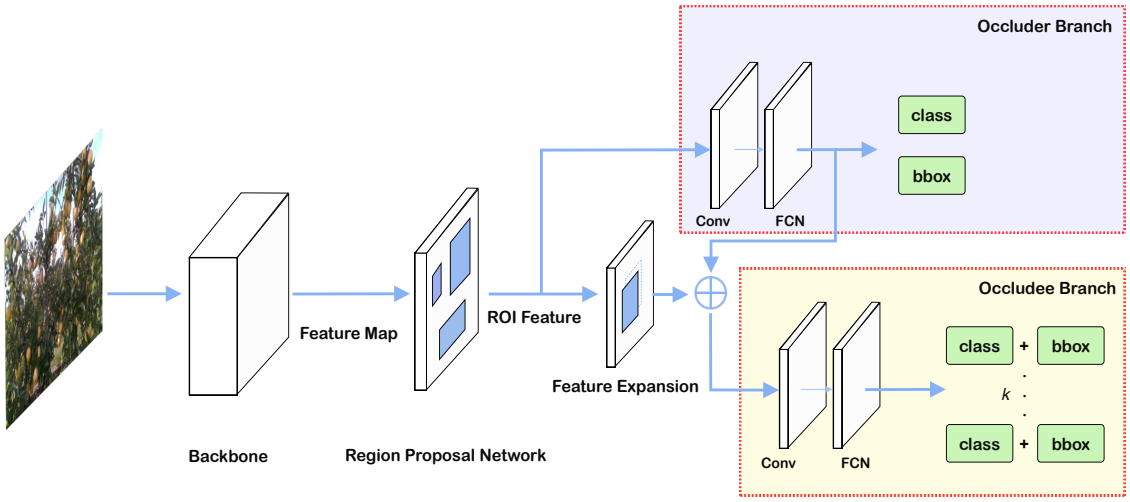

Figure 4.3 Network structure of the proposed Occluder-Occludee Relational Network (O2RNet). It consists of a feature learning backbone, RoI feature extraction, and object detection heads with occluder and occludee branches. The Feature Expansion Structure (FES) provides expanded RoI features along with features from the occluder branch to facilitate the detection of occludee.

called anchors, which are a set of predefined bounding boxes. The anchors are designed to capture the scale and aspect ratio of specific object classes and are typically chosen to be consistent with object sizes in the dataset. RPN is mainly used for predicting bounding boxes in Faster R-CNN but it can also provide enough anchors with different scales that will be exploited in our network as explained in the sequel. Third, I build an occlusion-aware modeling head with a structure of two classification and regression branches for occluder and occludee for decoupling overlapping relations and segments the instance proposals obtained from the RPN. Compared to the traditional class-agnostic classification, I divide this task into two complementary tasks: occluder prediction using the original classification head and occludee modeling with an additional Feature Expansion Structure (FES), where the occluder predictions provide rich foreground cues like textures and the FES predicts the positions of occluding regions to guide occludee object regression.

More specifically, an input image is first processed by the ResNet backbone to extract intermediate convolutional features for downstream processing. The object detection head (i.e., RPN) then predicts bounding box proposals, which are then consumed by the occlusion perception branches into the occluder branch and the occluee branch. For the occluder

branch, I adopt the object detection head in Faster R-CNN [94] to output positions as well as categories for instance candidates and prepare the cropped RoI features for the occludee branch. In the occludee branch, the input consists of both cropped RoI features from the occluder branch and expanded features from FES, which is targeted for modeling occluded regions by jointly detecting boundaries. Essentially, the distilled occlusion features are added to the original input RoI features and passed to the next module. Finally, the occludee branch, which has a similar structure to the occluder branch, predicts the occludee guided by these expanded features and outputs classes and bounding boxes for the partially occluded instances. I next describe the occluder-occludee relational modeling in more details.

4.3.3 Occluder-Occludee Relationship Modeling

For highly-overlapped apples, in typical Faster-RCNN-based models, the generated region proposals corresponding to the partially occluded ones may be separated into disjoint subregions by the occluder. As such, I employ the FES to obtain boundary features from the occludee, where expansion in each direction extends the potential proposals for the occludee. In our implementation, I expand t steps in k ($k = 8$ in this study) directions from the original RoI proposals, and the expanded RoI proposals will contain additional boundary features. The rationale is that irregular occlusion boundaries unrelated to the occludee can cause confusion to the network, which in turn provides essential cues for decoupling occludees from occluders. Therefore, I explicitly model occlusion patterns by detecting bounding boxes of the occluders using the occluder detection branch, and since the occludee detection branch jointly predicts bounding boxes for the occludee, the overlap between the two layers can be directly identified as occlusion boundary that can thus be distinguished from the real object bounding boxes. In order to reach this goal, the occluder modeling module is designed as a simple 3×3 convolutional layer followed by one FCN layer, the output of which is fed to the up-sampling layer and one 1×1 convolutional layer to obtain one channel feature map for occludee branch.

This O2RNet is particularly adept at discerning edge features on apples, a crucial step

in differentiating between occluders and occludees. As illustrated in Figure 4.4, O2RNet's capability becomes evident when dealing with a cluster of apples. Here, the algorithm identifies an occludee, and subsequently refines the bounding boxes for both the occluder and the occludee in a clustered apple. This refinement is pivotal in accurately representing the spatial relationships and physical boundaries of each apple in the cluster. In contrast, when the algorithm encounters an individual apple, devoid of any overlapping or obscuring elements, it classifies the apple solely as an occluder. In this scenario, the bounding box remains unaltered, as there is no need for refinement in the absence of an occludee. This dual functionality of O2RNet underscores its versatility and precision in handling varied scenarios within object detection tasks.

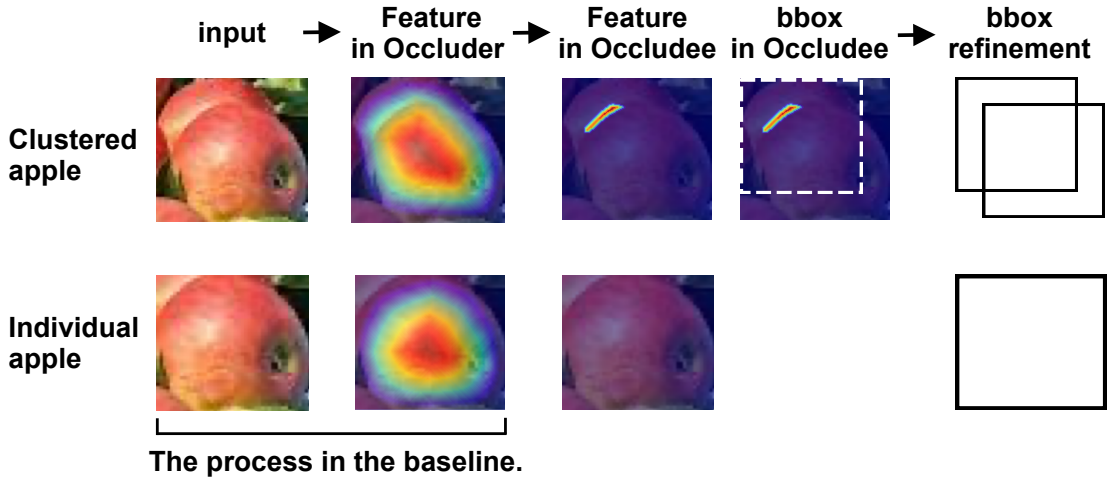

Figure 4.4 How O2RNet works on clustered apples and an individual apple. The first step in the occluder branch are the same with the baseline model(Faster R-CNN), which have similar feature maps and do not isolate cluster apple. In the occludee branch, O2RNet learns occludees feature and successfully splits the input into an occluder and an occludee.

4.3.4 End-to-end Learning

As I have two separate detection heads in the occluder and the occludee branches, I define two loss functions in the following way. For the occluder branch, I adopt the loss function used in Faster R-CNN [94], which defines a multi-task loss on each sampled region of interest

as

$$L_{Occluder} = L_{cls} + L_{bbox}, \tag{4.1}$$

where L_{cls} and L_{bbox} are, respectively, classification loss and bounding box loss defined in Faster R-CNN [94].

The final loss L is a weighted sum of the loss from occluder branch and the loss from occludee branch defined as:

$$L = \lambda_1 L_{Occluder} + \lambda_2 L_{Occludee}. \tag{4.2}$$

Here $L_{Occludee}$ is the occludee branch loss that is the sum of the k expanded proposal losses, i.e.,

$$L_{Occludee} = \sum_{i=0}^{k} (L^i_{cls} + L^i_{bbox}). \tag{4.3}$$

Here λ_1 and λ_2 are two positive linear weights and $\lambda_1 + \lambda_2 = 1$, which are tuned to balance the two loss functions. In my study, λ_1 was tuned to be $\{1.0, 0.75, 0.5, 0.25, 0\}$ on various trials for cross-validation.

4.3.5 Training and Inference

During the training process, I filter out parts of the non-occluded RoI proposals to keep occlusion cases taking up 50% for balanced sampling. SGD with momentum is employed to train the model with $60K$ iterations where it starts with $1K$ constant warm-up iterations. The batch size is set to 2 and the initial learning rate is 0.01 with a weights decay of 0.95. In my study, ResNet-101-FPN is used as the backbone and the input images are resized without changing the aspect ratio, i.e., by keeping the shorter side and longer side of no more than 1200 pixels. For inference, the occludee branch predicts bounding boxes for the occluded target object in the high-score box proposals generated by the RPN, while the occluder branch produces occlusion-aware features as input for the occludee branch. The one with the highest score is then chosen as the output.

Table 4.1 Performance of O2RNet on the customized apple dataset. The step is from FES, which represents how much features expanded. The evaluation uses AP, AR, and F1-score at the different IoUs.

Model	Step	AP	AP_{50}	AP_{75}	AR	AR_{50}	AR_{75}	F1-Score
O2RNet	**t=1**	**0.511**	**0.945**	**0.935**	**0.351**	**0.938**	**0.803**	**0.864**
	t=2	0.490	0.920	0.900	0.330	0.900	0.770	0.820
	t=3	0.490	0.920	0.904	0.328	0.900	0.770	0.820

Table 4.2 Model parameters numbers between the state-of-the-art networks and our proposed Occluder-occludee Relational Network (O2RNet). "M" stands for a million.

Models	Parameters	FPS
FCOS	2.0M	14
YOLOv4	0.6M	63
Faster R-CNN (ResNet50)	2.0M	19
Faster R-CNN (ResNet101)	3.6M	10
EfficientDet-b0	0.1M	48
EfficientDet-b1	0.3M	45
EfficientDet-b2	1.2M	25
EfficientDet-b3	1.6M	24
EfficientDet-b4	2.4M	16
EfficientDet-b5	3.6M	8
CompNet via BBV	0.8M	18
CompNet via RPN	1.4M	15
O2RNet (ResNet50)	2.0M	18
O2RNet (ResNet101)	3.6M	10

4.4 Experimental Results

4.4.1 Performance Metrics

For model development and evaluation, conventionally the apple dataset is randomly partitioned into training, validation, and test sets for model training and evaluation, respectively. To quantitatively evaluate the detection performance, I use performance metrics including precision, recall, and F1-score for algorithm evaluation. All detection outcomes are divided into four types: true positive (TP), false positive (FP), true negative (TN), and false negative (FN), based on the relation between the true class and predicted class. The precision (P) and recall (R) are defined as follows:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}. \tag{4.4}$$

The F1-score is then subsequently defined as:

$$F1 = \frac{2 \cdot P \cdot R}{P + R}. \tag{4.5}$$

To better evaluate the precision between the prediction and the ground truth, I also employ Microsoft Common Objects in Context (COCO) dataset [67] evaluation metrics. Specifically, after the calculation of precision and recall, I calculate the average precision (AP) and average recall (AR) based on different Intersection over Union (IoU) between the prediction and the ground truth. For example, $AP_{IoU=.50}$ or AP_{50} denotes that AP is averaged over $IoU = 0.50$ values, which belongs to PASCAL VOC metric [95]. I also use $AP_{IoU=.75}$ or AP_{75}, which is a stricter metric for model evaluations. In my study, I use a spectrum of 10 IoU thresholds ranging $0.50 : 0.05 : 0.95$ to average over multiple IoUs to obtain a comprehensive set of results.

4.4.2 Experimental Setup

In this section, I evaluate the efficacy of the proposed O2RNet on the processed data as discussed in Section 3.2. The network hyper-parameters, including the momentum, learning rate, decay factor, training steps, and batch size, are set as 0.9, 0.001, 0.0005, 934, and 1, respectively, through cross-validation. The input image size is 1280×720, which is aligned with the resolution of the camera used in our data collection. To better analyze the training process, I set up 80 epochs for training. I exploit a pre-trained model on the COCO dataset [67], where I train on 2017train ($115k$ images) and evaluate results on both 2017val and 2017test-dev to pre-train model parameters. This pre-trained model generally only takes 50 epochs to converge. By tuning the steps t in FES, different results are obtained and listed in Table 4.1, which shows that O2RNet with $t = 1$ leads to the best performance.

4.4.3 Performance Comparison and Analysis

To accelerate the model training on our customized dataset, I initialize parameters by transfer learning from ImageNet [26]. ImageNet provides large-scale images in different fields (including apples) and large-scale ground truth annotation. During the transfer learning process, my model learns specific characteristics with an effective transfer of features from ImageNet. Compared to randomized parameters, the results (see Figure 4.5) shows that my model converges faster as benefited from the pretraining on a large-scale database.

Table 4.3 Performance of O2RNet on the augmented dataset. The geometric transformations consist of rotation, flipping and scaling. The color space transformations consist of brightness and contrast shifting. Finally, all of the augmentation methods are integrated to evaluate the O2RNet.

Augmentation	AP	AP_{50}	AP_{75}	AR	AR_{50}	AR_{75}	F1-Score
Base	0.51	0.92	0.90	0.35	0.91	0.80	0.84
Geometric transformations (GTs)	0.52	0.93	0.91	0.35	0.91	0.80	0.85
Color space transformations (CSTs)	0.52	0.93	0.91	0.35	0.91	0.81	0.85
Gausian noise	0.48	0.91	0.90	0.34	0.91	0.80	0.83
Mixup	0.52	0.93	0.92	0.35	0.92	0.81	0.85
Sharpening	0.52	0.92	0.90	0.35	0.91	0.80	0.84
GTs+CSTs+Mixup	**0.52**	**0.96**	**0.94**	**0.36**	**0.94**	**0.83**	**0.88**
All	0.52	0.94	0.92	0.36	0.92	0.83	0.86

Table 4.4 Performance comparison of our own models and other 12 state-of-the-art deep learning models on the customized apple dataset.

Models		AP	AP_{50}	AP_{75}	AR	AR_{50}	AR_{75}	F1-score
FCOS [1]		0.48	0.89	0.87	0.34	0.87	0.78	0.80
YOLOv4 [84]		0.45	0.87	0.84	0.29	0.84	0.73	0.76
Faster R-CNN	ResNet50 [82]	0.48	0.89	0.87	0.32	0.87	0.78	0.81
	ResNet101 [82]	0.49	0.94	0.93	0.31	0.84	0.75	0.82
EfficientDet	EfficientDet-b0 [83]	0.45	0.89	0.85	0.30	0.82	0.71	0.77
	EfficientDet-b1 [83]	0.45	0.89	0.86	0.30	0.82	0.72	0.77
	EfficientDet-b2 [83]	0.46	0.89	0.87	0.30	0.82	0.73	0.78
	EfficientDet-b3 [83]	0.49	0.93	0.91	0.32	0.84	0.75	0.81
	EfficientDet-b4 [83]	0.50	0.94	0.92	0.34	0.88	0.78	0.82
	EfficientDet-b5 [83]	0.50	0.95	0.93	0.34	0.88	0.78	0.83
CompNet	CompNet via BBV [127]	0.50	0.94	0.92	0.36	0.94	0.80	0.85
	CompNet via RPN [34]	0.51	0.95	0.94	0.35	0.94	0.80	0.86
O2RNet	O2RNet-ResNet50	0.50	0.93	0.91	0.35	0.91	0.80	0.84
	O2RNet-ResNet101	**0.52**	**0.96**	**0.94**	**0.36**	**0.94**	**0.83**	**0.88**

Furthermore, data augmentation is another useful technique to optimize detection performance without increasing inference complexity. I applied five augmentation strategies, including geometric transformations (GTs), color space transformations (CSTs), Gaussian noise injection, mixup and sharpening data augmentation, to extend our dataset. The results are summarized in Table 4.3. It shows that GTs such as rotation, flipping and scaling – by changing the pixel position of the image and reordering apples in the image – improve the accuracy performance by around 1%. Through changing color illumination and intensity of an image, CSTs also roughly increases the performance by 1%. Due to the sparsity of

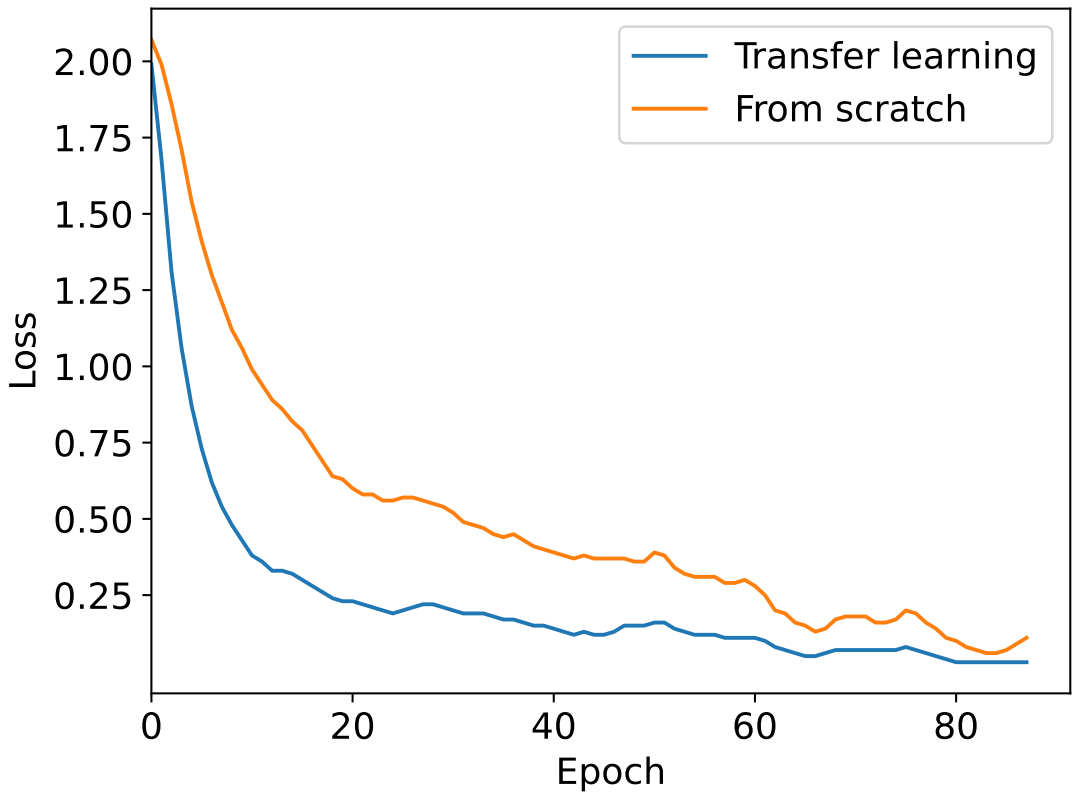

Figure 4.5 Training loss comparison between transfer learning and training from scratch on my model (O2RNet). The training loss with transfer learning from ImageNet apparently decreases and converges faster as compared with training from scratch.

apples on some images, mixup helps enlarge apple density on the image and enhances the accuracy by 2%. It turns out that Gausian noise and sharpening do not help much, as they try to change textures and increase complexities on the dataset, which generate confusing data and is not suitable for my model. Finally, the augmentation combination of GTs, CSTs and Mixup offers the best enhancement by increasing the accuracy of 4% on our dataset.

To better evaluate the performance of my model, I compare our O2RNet with the-state-of-art object detection methods on our customized apple dataset (see Table 4.2 for a list of benchmark models and their number of parameters). In particular, FCOS and YOLOv4 are representatives of one-stage detectors, achieving consistent improvement and demonstrating their effectiveness by outperforming the SSD method [69] on several public datasets [110, 11]. I also evaluate Faster R-CNN and EfficientDet since they are state-of-the-art models with promising performance demonstrated in fruit harvesting-related works [78, 125]. I also

compare O2RNet with the state-of-the-art occlusion-aware network CompNet [34].

I then use the same experimental setup to train each model and evaluate them on the same apple test dataset. The results are shown in Table 4.4, which compares the detection precision and recall over different IoUs among the 14 selected models (including our O2RNet). Notably, in addition to FCOS, EfficientDet-b5 and Faster R-CNN achieved decent F1-scores of 0.83 and 0.82, respectively. Two occlusion-aware networks, CompNet and our O2RNet clearly outperform all traditional models with F1-scores of 0.86 and 0.88, respectively, and O2RNet clearly shows superior performance over CompNet. Some representative inference results are shown in Figure 4.6. It can be seen that our O2RNet can effectively separate clustered apples and thereby improves the precision and recall and subsequently the F1-score.

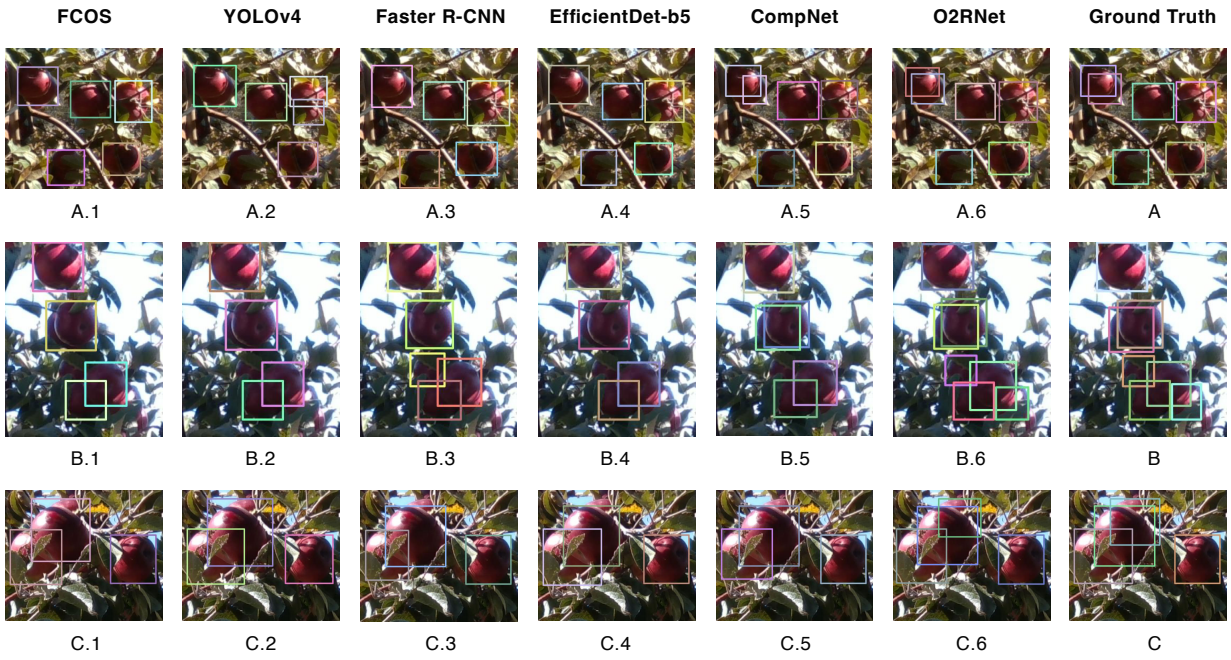

Figure 4.6 Results from six models on the various lighting conditions and occlusions.

4.5 Summary of the Chapter

I collected a comprehensive apple dataset under different lighting conditions and at various occlusion levels from two real orchards. A novel Occluder-Occludee Relational Network (O2RNet) was developed to robustly detect clustered apples from the dataset. Our de-

veloped O2RNet significantly reduced false detection and improved the detection rate by embedding relationships between the occluder and the occludee. State-of-art performance was demonstrated in comprehensive experiments. I also found that transfer learning and data augmentation techniques were useful tools to enhance learning efficiency and model performance.

CHAPTER 5

ALACS: ACTIVE LASER-CAMERA SCANNING FOR 3D APPLE LOCALIZATION

In this chapter, I review several consumer RGB-D cameras and depict our proposed localization technique, called Active Laser-Camera Scanner (ALACS). I propose a feature-matching algorithm, called Laser Line Extraction (LLE), to help ALACS transform the 2D fruit positions to 3D fruit positions, thus achieving accurate mapping even under complex fruit morphology, variable lighting conditions and occlusions.

5.1 Introduction

Three-dimensional localization is the other crucial aspect of fruit perception. Accurate and reliable localization of objects such as fruits in the 3D space is essential for automated agricultural systems to optimize crop management and harvesting strategies, ultimately improving productivity, efficiency, and sustainability [40]. Several types of 3D sensors are currently available, including Time-of-Flight (ToF) cameras, LiDAR (Light Detection and Ranging), stereo-vision cameras, and structure light systems. Specifically, ToF cameras [61] measure depth by emitting a light signal (usually IR) and measuring the time it takes for the signal to bounce back to the sensor. This time difference allows the camera to calculate the distance to objects in the scene. ToF cameras, such as the PMD CamBoard pico [5] and Microsoft Kinectv2 [80], are known for their accuracy and high-resolution depth data. LiDAR systems [92] use lasers to send out light pulses and measure the time it takes for the pulses to return after reflecting off objects. This time difference is used to calculate distances and generate a 3D point cloud of the scene. LiDAR systems can provide accurate depth data but are often more expensive than other methods and often have limited spatial resolution. Stereo vision systems, on the other hand, use two cameras, typically placed side-by-side at a known distance apart, to capture images of the same scene. By comparing the images and identifying corresponding points in each image, depth information can be estimated using triangulation. Stereo vision systems [62] often require more computational power and

can be sensitive to lighting conditions, but they do not rely on active IR illumination. In contrast, structured light systems [112] project a known pattern of light (usually infrared) onto the scene and then capture the deformed pattern with a camera. The deformation of the pattern allows the system to reconstruct the 3D geometry of the scene. It usually works well in low-light conditions (since it uses active illumination) but it is sensitive to ambient light and surface properties (e.g., reflectivity, transparency).

Over the years, numerous techniques have been attempted for fruit 3D localization based on the aforementioned sensors and advanced computer vision methods [77, 45, 128, 68]. Specifically, [76] employed a stereo camera system in conjunction with a tailored fruit-matching algorithm, reporting a localization error of approximately 11 mm in the simplified indoor environment. However, outdoor agricultural environments are significantly more challenging, with exposure to a wide spectrum of lighting conditions and complex tree and fruit structures, which can cause major issues to its fruit matching techniques as they can significantly influence the visual characteristics and discernibility of fruits captured within the images or point clouds. [2] utilized a commercial device, RealSense RGB-D camera, to obtain the positions of apples and reported a localization error of around 9.5 mm in an ideal indoor environment. However, due to the low resolution of the projector in the depth camera, RGB-D cameras have to interpolate based on partial depth measurements. Unlike plane surfaces, fruits can exhibit a wide range of shapes, sizes, colors, and textures, making it difficult to develop a one-size-fits-all approach to 3D fruit localization. To overcome complex fruit morphology, [101] utilized a global time-of-flight camera to obtain fruit point cloud by removing the background, in order to accurately localize each point on the fruit. However, in real-world orchard scenarios, fruits are often surrounded by leaves, branches, and other fruits, which can create occlusions and clusters in the images or point clouds. These issues make it difficult to accurately identify and localize the fruits, particularly when they are partially or fully occluded.

In this chapter, I address the above challenges by designing and developing a novel Active

Laser-Camera Scanner (ALACS) system. Specifically, an RGB camera is integrated with a line laser to achieve robust and accurate localization using the triangulation principle. I propose a feature-matching algorithm, called Laser Line Extraction (LLE), to help ALACS transform the 2D fruit positions to 3D fruit positions, thus achieving accurate mapping even under complex fruit morphology, variable lighting conditions and occlusions. This research is expected to provide a valuable reference for future research on developing fruit 3D localization systems in fruit harvesting. The main contributions of this chapter are highlighted as follows:

1. A novel Active Laser-Camera Scanning system, consisting of a red line laser, an RGB-D camera, and a linear motion slide, is designed and developed for accurate fruit 3D localization.
2. A Laser Line Extraction (LLE) algorithm is proposed and implemented for robust feature matching to enable stable 2D-3D transformation for ALACS.
3. System evaluation and validation of the ALACS are performed indoors and outdoors in comparison with a conventional 3D sensing technique, i.e., Intel RealSense D435i RGB-Depth camera.

5.2 Active Laser-Camera Scanning (ALACS)

The ALACS is designed to provide accurate 3D localization of apples in orchards by combining the advantages of a depth camera and laser scanning. The major hardware components include a red line laser (Laserglow Technologies, North York, ON, Canada), a FLIR RGB camera (Teledyne FLIR, Wilsonville, OR, USA), a linear motion slide, and an Intel RealSense D435i RGB-D camera, which is used for providing rough initial global estimates of fruits. As shown in Figure 5.1, the RGB-D camera is mounted on a horizontal frame that is above the manipulator to provide a global view of the scene and initialize the laser position. The line laser is mounted on the linear motion slide that enables the laser to move horizontally with a full stroke of 20 cm. Meanwhile, the FLIR RGB camera is installed at the left end of the linear motion slide with a relative angle to the laser to

capture laser patterns on apples. The hardware configuration of the ALACS is designed to facilitate depth measurements based on the principle of laser triangulation [30]. Specifically, the laser triangulation-based technique is a classical high-precision localization scheme that captures depth measurements by pairing a laser illumination source with a camera. It is worth noting that with the conventional laser triangulation sensors, the relative position and pose between the laser and the camera is fixed (i.e., both of them are static or moving simultaneously), whereas in ALACS the camera is fixed while the laser position is actively adjusted with the linear motion slide to seek the target fruit (see subsequent discussions for more details). Specifically, ALACS performs fruit 3D localization in three steps: laser scanning, target position determination, and 2D-3D position transformation.

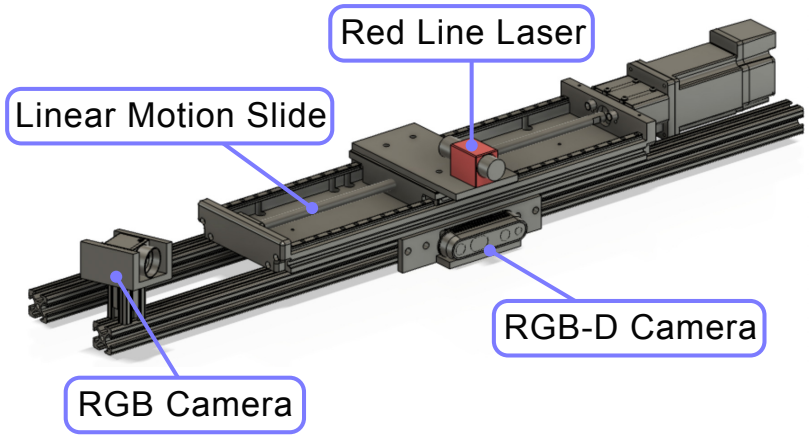

Figure 5.1 CAD model of ALACS.

As shown in Figure 5.2, the first step in the ALACS workflow is to capture a global view of the scene using the RGB-D camera. This camera provides both color and depth information, which can be used to segment and identify bounding boxes containing apples based on an apple detection approach (see [24]). Then, a rough 3D position of each detected apple is estimated. Based on a planning strategy [134], ALACS selects a target apple and uses the rough 3D position provided by RealSense D435i to guide the laser to the initialized position related to the target apple. The laser is then moved horizontally from the left to the right side of the apple in five 2-cm increments; it illuminates the target apple and creates

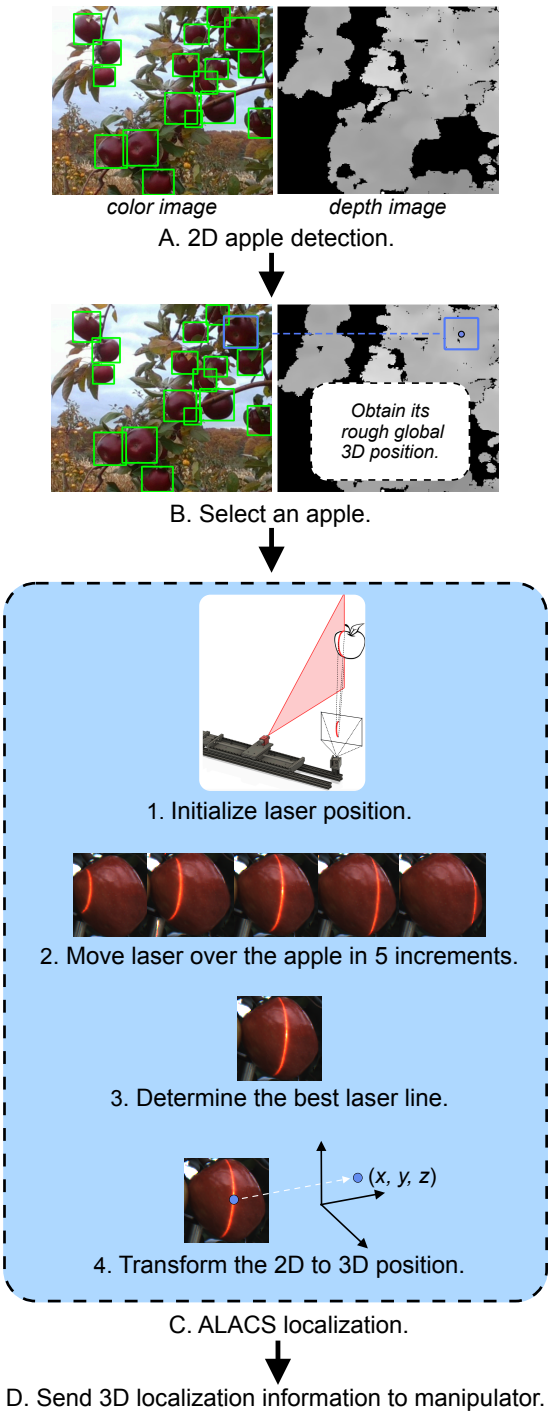

Figure 5.2 Schematics of the ALACS workflow: In step A, the RGB-D camera provides an initial global view of detected apples. In step B, a target apple is determined based on a planning strategy and its rough 3D location is sent to ALACS. In step C, the fruit is scanned and the high-precision position is obtained. Finally in step D, the 3D localization information is sent to the manipulator for fruit picking.

visible laser lines on the surface of the fruit (see Figure 5.3).

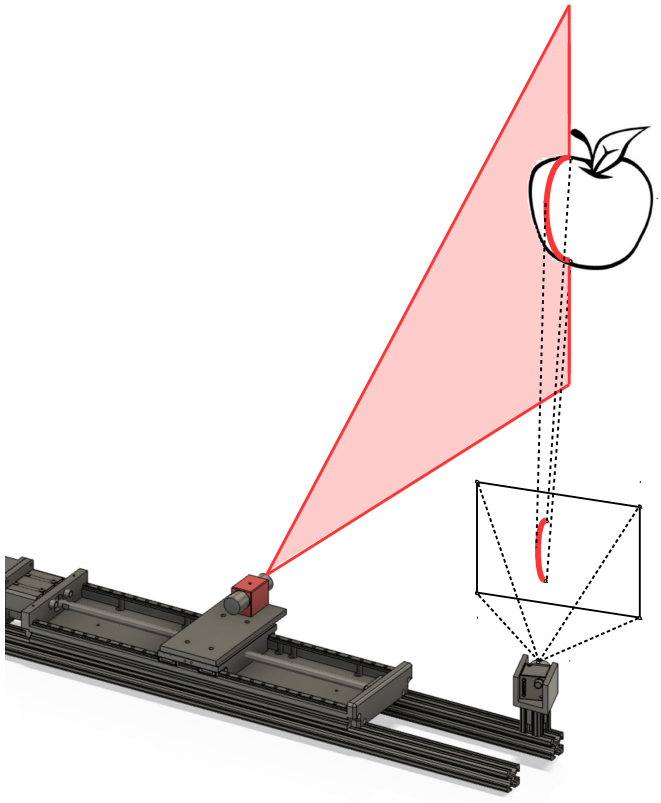

Figure 5.3 Schematic of laser scanning on a target fruit in ALACS to obtain one laser line from the fruit.

During the laser scanning process, at each stop an image is obtained at each stop with the illuminated apple being captured by a high-resolution FLIR camera. ALACS uses a laser line extraction algorithm (see Section 5.3) to extract laser patterns on the target apple for each stop. These candidates resulting from the five different laser projections would cover the target apple. The most reliable candidate is then selected to determine the apple's centroid position based on a confidence evaluation for each candidate, which is calculated using two key factors: the distance to the estimated center and the number of extracted laser line pixels.

Distance to the Estimated Center: The distance factor in the confidence calculation quantifies how close the candidate laser line is to the apple's estimated center (obtained from our apple detection algorithm [24]). Candidates that are closer to the center are considered more

reliable and are assigned higher confidence scores. This distance factor Δd is calculated using the Euclidean distance between the candidate's position and the estimated center.

Number of Extracted Laser Line Pixels: The second factor contributing to the confidence calculation is the number of extracted laser line pixels, $\mathcal{N}$, for each candidate. Candidates with more extracted laser line pixels are considered to provide a more complete representation of the apple's surface geometry and are thus deemed more reliable. Consequently, these candidates are assigned higher confidence scores. Then the confidence $\mathcal{P}$ is calculated by $\mathcal{P} = \omega_1 \cdot \mathcal{N} - \omega_2 \cdot \Delta d$, where ω_1 and ω_2 are weights for these two factors and are obtained through cross-validation.

After selecting the most reliable candidate based on the calculated confidence scores, the 2D position of the center of this laser line is obtained and the apple's center position is determined using the laser triangulation scheme [30]. The basic idea of this technique is to capture depth measurements by pairing a laser illumination source with a camera. Both the laser beam and the camera are aimed at the target object, and based on the extrinsic parameters between the laser source and the camera sensor, the depth information can be computed using trigonometry. The transformation rule for 2D position (u_i, v_i) to 3D position (x_i, y_i, z_i) follows

$$\begin{aligned} z_i &= \frac{L}{\sin(\alpha) - u_i \cos(\alpha) - v_i \tan(\beta)}, \\ x_i &= \frac{L u_i}{\sin(\alpha) - u_i \cos(\alpha) - v_i \tan(\beta)}, \\ y_i &= \frac{L v_i}{\sin(\alpha) - u_i \cos(\alpha) - v_i \tan(\beta)}, \end{aligned} \tag{5.1}$$

where extrinsic parameters L, α, and β are, respectively, the baseline (i.e., the distance between the camera and the line laser), horizontal angle, and vertical angle between the laser illumination source and the camera. The details of parameter estimation are discussed in [130]. Therefore, ALACS finally localizes the target apple. After the apple has been picked, the process repeats for the next target fruit.

5.3 Laser Line Extraction (LLE)

In this section, I present more details on the laser line extraction steps that are of paramount importance in the ALACS system since it serves as the crucial link between the laser scanning process and the final triangulation-based localization. In ALACS, images of the illuminated apple are captured using a high-resolution camera. These images are then processed to extract the laser lines on the apple's surface. Extracting accurate and well-defined laser lines from the captured images provides essential geometric information about the apple's surface. Furthermore, effective laser line extraction techniques can help mitigate the impact of noise, occlusions, and illumination variations, which can significantly improve the overall performance and reliability of the ALACS method. I next discuss the specific algorithms and techniques employed for laser line extraction and their role in enhancing the accuracy of the ALACS-based apple localization process.

I have chosen a red laser as the domain laser color, as opposed to blue or green lasers. This choice was made based on our preliminary evaluations, which found that red laser lines appear more intense and distinct in the captured images (see Figure 5.4 for comparisons among three lasers of different colors), thus facilitating more effective extraction of laser lines. More specifically, LLE is designed with 4 steps: laser pattern detection, noise removal, line focus, and curve fitting, as illustrated in Figure 5.5.

Laser Pattern Detection: Based on the utilized laser line pattern, various image processing techniques, such as edge detection, filtering, and thresholding, can be employed to identify and isolate the laser lines in the captured images. Thresholding is the simplest way to extract highlighted patterns but it is usually affected by strong external lighting like sunlight. Edge detection using Sobel or Robert kernel [27] can find line patterns accurately but they need more computation time compared to thresholding. Hence, I designed a novel algorithm, called bidirectional Relative Color Enhancement (bRCE), to detect the laser line pattern from the selected apple image $\mathcal{I}$ efficiently. The bRCE (see Algorithm 5.1) gets bi-directional horizontal gradient matrix$\mathcal{G}$, which is calculated through shifted differences of the image

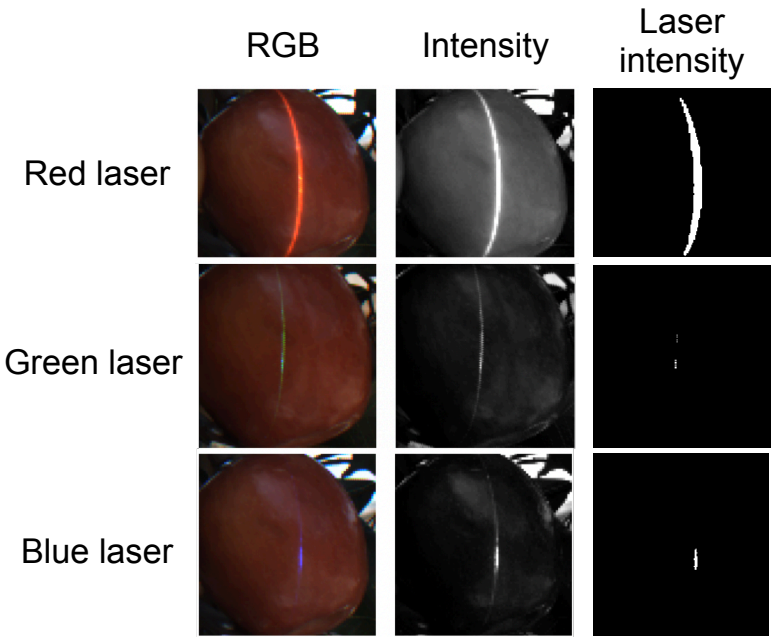

Figure 5.4 Comparison of laser projections on a red apple by a red laser (635 nm), green laser (515 nm) and blue laser (447 nm). Laser intensity is obtained using a thresholding of 140.

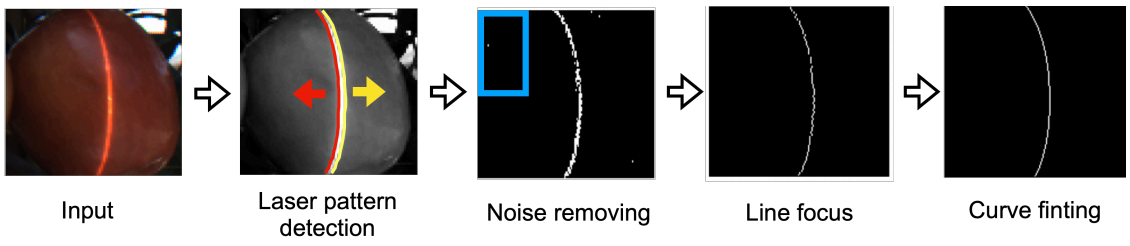

Figure 5.5 Schematic of the laser line extraction (LLE) workflow.

in both left-to-right and right-to-left directions. The bRCE can effectively highlight the boundaries of the laser lines, help enhance the contrast of the laser lines and make them more distinguishable from the background. The bRCE outperforms the thresholding-based method especially in over-exposure situations, as shown by an example in Figure 5.6.

Noise removing: The laser line patterns are usually highlighted with small outliers, as shown by examples in Figure 5.7, since the over-exposure part caused by strong sunlight can interfere with the laser line. To address the discontinuity of the outliers, I employ a sliding window counting method to remove these outliers. As shown in Algorithm 5.2, a predefined window with size $\mathcal{W}$ is slid horizontally across the image row by row based on the partition size γ,

Algorithm 5.1 bRCE laser pattern detection

Input: $\mathcal{I}$: $n \times m \times 3$ RGB image matrix of the selected apple.
Output: $\mathcal{G}$: $n \times m$ laser prediction matrix.
Params: *step*: gradient step, *th*: gradient threshold.

$\mathcal{R}$ = red channel of $\mathcal{I}$
$\mathcal{G}_1 = \mathcal{R}[:, : -2 \cdot step] - \mathcal{R}[:, step : -step]$
$\mathcal{G}_2 = \mathcal{R}[:, 2 \cdot step :] - \mathcal{R}[:, step : -step]$
/*$\mathcal{G}_1, \mathcal{G}_2$ are two horizontal gradients of $\mathcal{I}$.*/

for $\mathcal{G}_1, \mathcal{G}_2$ **do**
  $\mathcal{G}_i[\mathcal{G}_i \leq th] = 0$
end
$\mathcal{G} = \mathcal{G}_1 \circ \mathcal{G}_2$, ▷ $\circ$ is element-wise product
$\mathcal{G}[\mathcal{G} > 0] = 1$

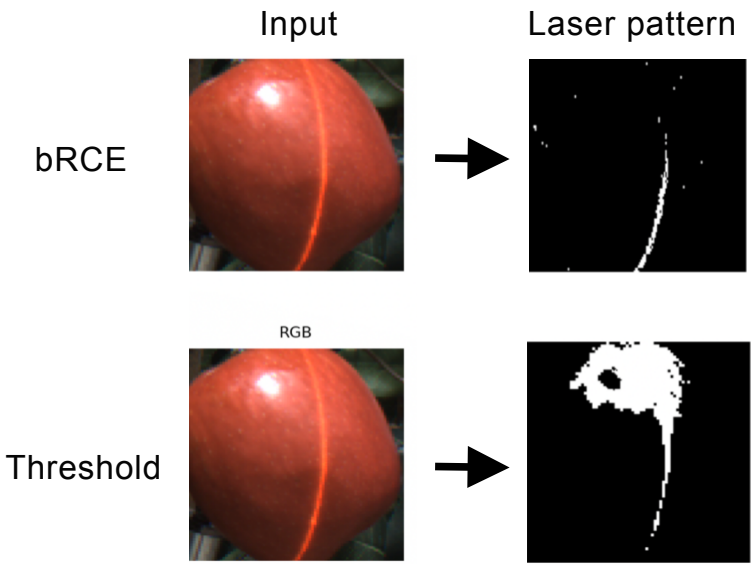

Figure 5.6 Laser pattern detection comparison between bidirectional Relative Color Enhancement (bRCE) with step= 8, th= 40 and thresholding with th= 220 under over-exposure situations.

and the number of strong curve pixels within the window is counted. If the count exceeds a pre-specified threshold θ, the pixels within the window are considered part of the laser line; otherwise, they are categorized as noises and discarded. This filtering process helps retain only the most prominent laser lines $\mathcal{C}$ in the image while eliminating undesired artifacts and noises.

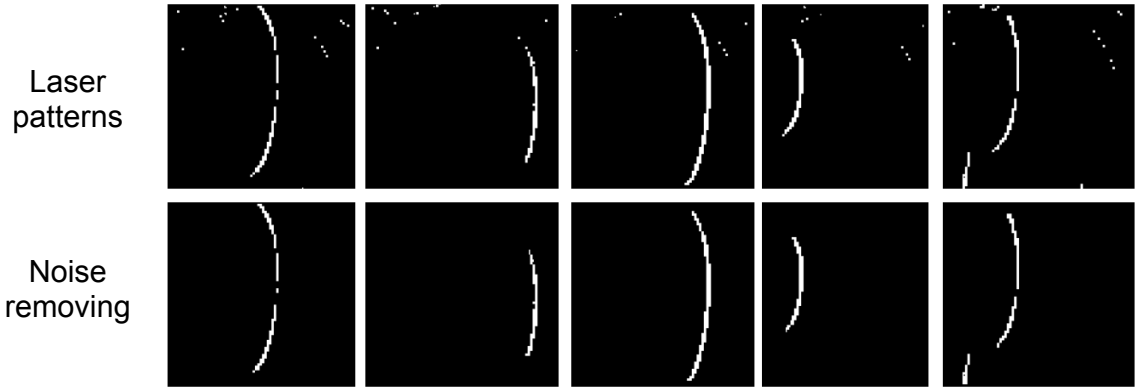

Figure 5.7 Extracted laser patterns with outliers (noises) (first row) and resultant laser patterns after noise removal with the use of a threshold of $\theta = 8$ and the partition size of $\gamma = 3$ (second row).

Algorithm 5.2 Noise removing

Input: $\mathcal{G}$: $n \times m$ laser prediction matrix (from Algo. 1).
Output: $\mathcal{C}$: $n \times m$ noise-free laser matrix.
Params: w_G, h_G: width and height of $\mathcal{G}$,
θ: threshold of noise, γ: partition size of $\mathcal{G}$.

/* $\mathcal{W}_{x,y,w,h}$ is the sliding window with the left-top position (x, y), the width w and the height h. */
$w, h = \frac{1}{4}w_G, \frac{1}{2}h_G$
s.t. $0 < \theta \leq w \times h$, $\frac{h}{w} > 1$
$\mathcal{C} = \mathcal{G}$
$u = \frac{w_G - w}{\gamma}$
$v = \frac{h_G}{h}$
for $i = 1, 2, 3 \ldots u$ **do**
  for $j = 1, 2, 3 \ldots v$ **do**
    $x = (i-1) \cdot \gamma$, $y = (j-1) \cdot h$,
    if *sum of* $\mathcal{W}_{x,y,w,h} \leq \theta$ **then**
      $\mathcal{C}[x : x + w, y : y + h] = 0$,
    end
  end
end

Line focusing: To enable a point-to-point feature matching for laser triangulation, LLE proceeds to extract a centerline for each laser pattern (generally with a width of greater than 2 pixels) by computing the centroids of the remaining strong laser pattern pixels on a row-by-row basis. By averaging the column indices of the strong curve pixels, the algorithm determines the centroid of the laser line in each row, which are then connected to form a

continuous centerline. This focused centerline effectively represents the central path of the laser line in the image, which will be used as the basis for the final continuous curve fitting step discussed next.

Curve fitting: The focused laser line obtained from the last step is not always smooth and continuous. As such, I use a polynomial to fit the rough focused line and thus generate a smooth and continuous curve that accurately represents the laser line in the image. The choice of polynomial order depends on the expected curvature of the laser line on the apple's surface, with higher-order polynomials offering greater flexibility to fit complex shapes. To avoid overfitting problem and through cross-validation, the 4th-order polynomial is used in this study to fit curves, which strikes a good tradeoff between accuracy and simplicity based on our preliminary evaluations.

5.4 Experimental Results

In ALACS, the 3D localization performance is affected by both feature matching accuracy and 3D positions estimation. I thus separately evaluate the performance of LLE and the final apple 3D localization performance.

5.4.1 LLE Evaluation

To evaluate the performance of LLE, I conducted a series of experiments under varying lighting conditions (from 1000 to 6500 lux) in the outdoors environment, specifically overcast and direct lighting scenarios.These tests were intended to assess the robustness and effectiveness of the LLE algorithm in extracting laser lines in challenging environments. To evaluate the laser line extraction performance using bRCE, I first varied the bRCE parameters *step* and th to observe how they affect laser extraction performance (see Figure 5.8). Based on tests at different distances of 0.8 m, 1.0 m, 1.2 m, 1.4 m, and 1.6 m, it was found that the laser lines on the surface of apple are around 4-pixel wide. When I tuned *step* from 2 to 14 with a fixed th, the best results are generated with around step= 4. With *step* increases, the shape of laser pattern becomes slacking. A similar approach was used to tune th from 20 to 70, and the number of laser pixels tends to decline as th increases. After calculating

the extracted line's mean and standard deviation from different combination of parameters on 400 images, the best parameters are chosen to be step= 4 and th= 40.

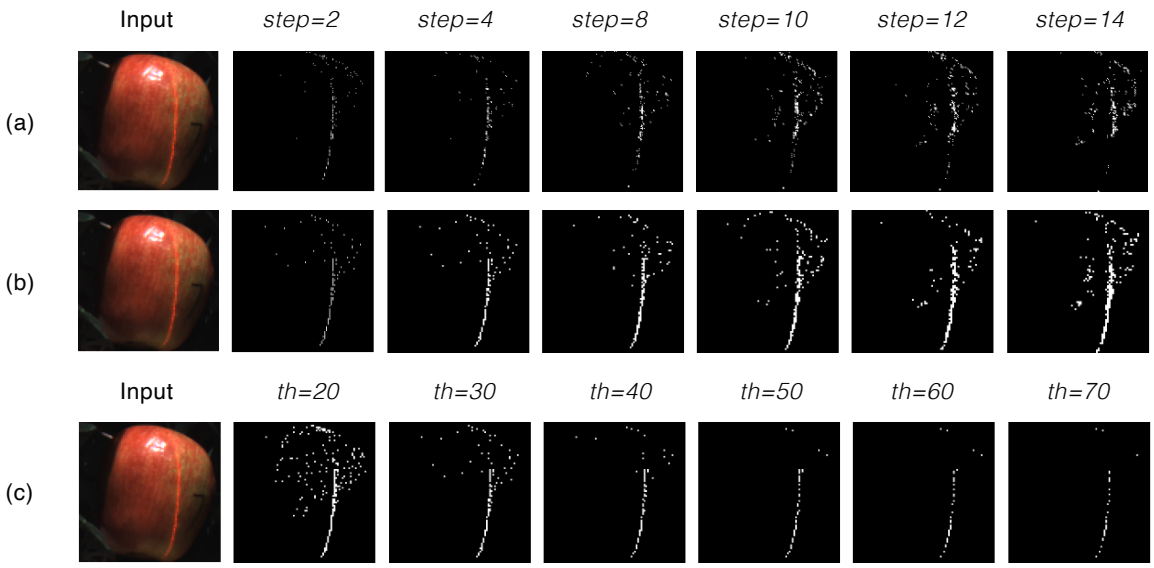

Figure 5.8 An example of the laser pattern detection performance with different bRCE parameters: (a) with a threshlod value th= 60 and the step varying from 2 to 14; (b) with th= 30 and the step varying from 2 to 14; and (c) with step= 4 and th varying from 20 to 70. The best result is obtained using step= 4 and th= 40 for this input.

Furthermore, to perform a quantitative evaluation of the LLE algorithm, I calculate the laser line displacements between the LLE predictions and the ground truth, which are obtained through manual labeling. This evaluation provides a quantitative measure of the accuracy and reliability of the LLE algorithm in extracting laser lines under various conditions. For this evaluation, a set of images with visible laser lines were manually annotated by trained persons, who carefully trace the laser lines and mark their positions as ground truth. These ground-truth annotations serve as a reference for comparing the performance of the LLE algorithm against an ideal extraction. LLE was then applied to the same set of images, and the resulting laser line predictions were compared with the ground truth annotations. The laser line displacements were calculated as the average pixel-wise Euclidean distances between the points on each row between the LLE-predicted laser lines and the ground-truth laser lines. Smaller displacements indicate a higher degree of agreement between the LLE

predictions and the ground truth, reflecting a more accurate and reliable extraction performance. Since I only used central results to localize, I calculated the displacements based on different central segments of each laser line. The results are summarized in Table 5.1. With calculating displacements in 10% central segment of the laser line, the LLE generated 1 pixel displacement in average.

Table 5.1 Performance of the laser line extraction (LLE) algorithm, as measured by average (Avg), minimum (Min) and maximum (Max) displacements (Disp.) in pixels for various central segment ratios (from 10% to 80%) on 300 cases.

	10%	20%	30%	40%	50%	60%	70%	80%
Avg Disp.	1.0	1.2	1.3	1.4	1.5	1.7	1.7	1.9
Min Disp.	0	0.2	0.3	0.4	0.4	0.5	0.5	0.6
Max Disp.	2.9	3.2	3.3	3.5	3.8	3.9	4.0	4.2

By analyzing the laser line displacements across various images and conditions (see Figure 5.9), I can gain insights into the performance and robustness of the LLE algorithm. To avoid image saturation caused by direct lighting, the extracted line by LLE is not always entire and continuous. In the meantime, occlusions also divide laser lines into different parts. These discontinuous challenges are fixed by the polynomial curve fitting (see Figure 5.9) and make the final laser line attach to the ground truth. This quantitative evaluation shows us how accurate LLE identify laser patterns even under direct sunlight, ultimately contributing to the overall performance of the ALACS-based apple localization system.

5.4.2 ALACS Localization Evaluation

To assess the performance of the ALACS-based apple localization, I conducted a series of evaluation experiments including occlusion and cluster cases, in both indoor and outdoor environments. In the indoor environment, I compared the localization results obtained by the ALACS method against the ground truth data acquired using a high-precision Qualisys localization system (Qualisys, Sweden) with an accuracy of 0.11 mm [17].To facilitate the acquisition of ground truth data, markers were placed on the apples in the orchard, and the Qualisys system was employed to accurately determine their 3D positions. These ground

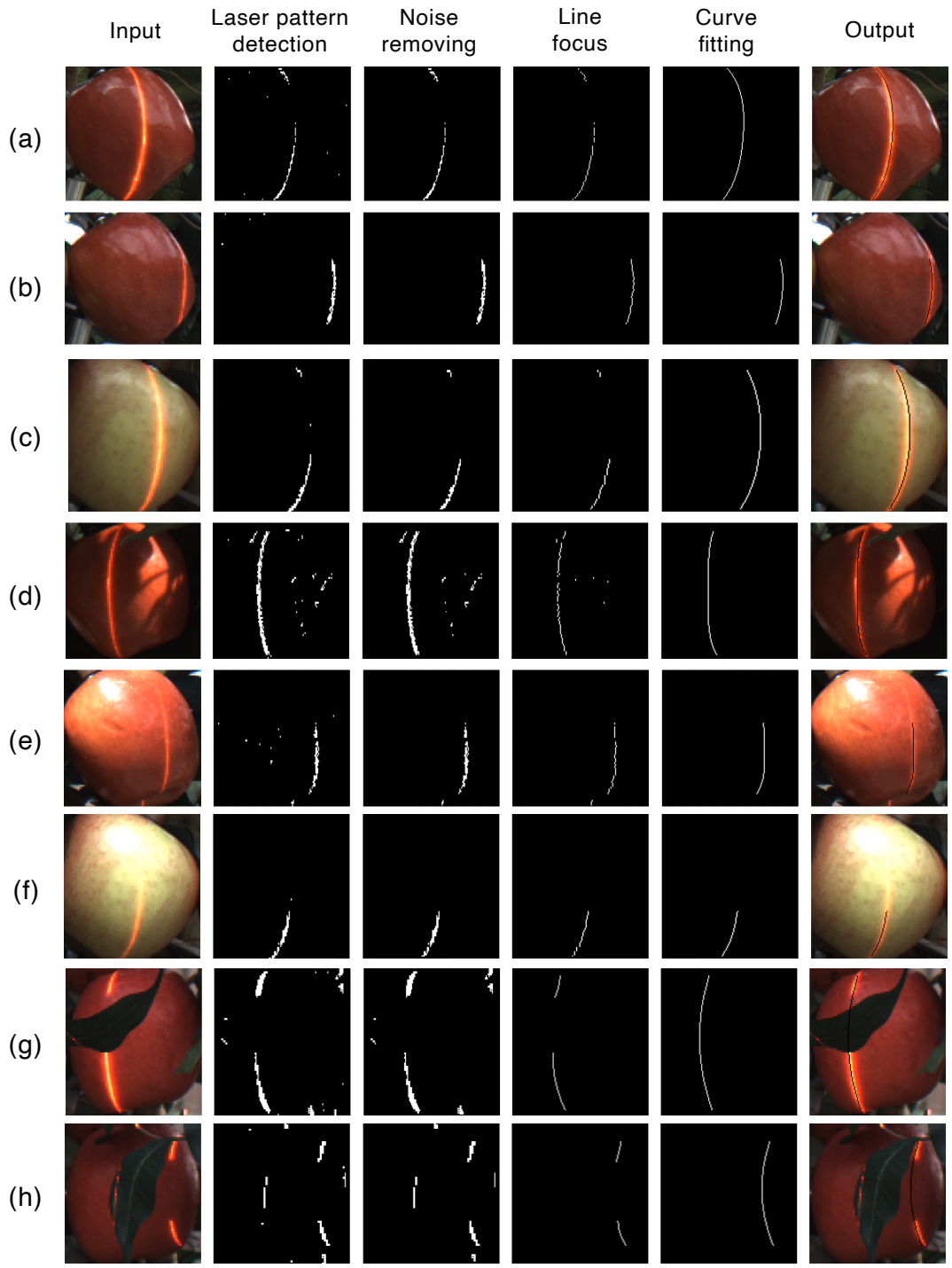

Figure 5.9 The visualization of the LLE process under different lighting conditions and occlusions (step= 4, th= 40): (a)-(c) show the apples with different laser line positions under overcast conditions; (d)-(f) show the apples under direct sunlight; (g) and (h) show the occluded apple cases.

truth positions served as a reference for evaluating the performance of the ALACS-based localization method.

In the first evaluation experiment, I allowed the ALACS system to project a single laser line to the center of the marker placed on the apple, under both occlusion-free and occlusion-present situations. To mimic occlusion situations, I used artificial foliage to partially cover apples. The ALACS system then estimated the apple's position using the extracted laser line and the target position estimation process. The resulting ALACS localization results were compared against the ground truth positions acquired using the Qualisys system. Results in Figure 5.10 show that ALACS achieved superior localization performance for occlusion-free situations; the average distance error ranges from 2.5 mm to 5.8 mm at distances from 1.0 m to 1.6 m. When the apples were occluded by leaves, the average localization errors were significantly larger than those without occlusions. The average distance errors, under the occlusion situations, are 6.9 mm, 7.2 mm, 9.0 mm and 11.2 mm respectively at a distance of 1.0 m, 1.2 m, 1.4 m and 1.6 m. Our harvesting robot uses a vacuum-based end effector to pick fruits, which can tolerate localization errors within 20 mm [131]. Hence, the ALACS system can still meet the localization accuracy requirements when apples are occluded by leaves.

In the second indoor evaluation experiment, I tested the ALACS system's ability to estimate the apple's position using multiple laser projections, respectively under occlusion-free and occlusion-present situations. The ALACS system acquired the laser lines at a 2-cm increment for five times over the apple, projecting different laser lines at various positions on the apple's surface. The LLE algorithm was then used to extract the laser lines to determine the apple's center position based on these five laser projections. As shown in Figure 5.11), under the occlusion-free conditions, the average distance error range from 5.5 mm to 9.1 mm for distances ranging from 1.0 m to 1.6 m, compared with the results obtained in the first experiment for a single laser line for the occlusion free condition. Under the occlusion situation, the average distance errors were 9.2 mm, 11.6 mm, 13.9 mm, and 17.5 mm at

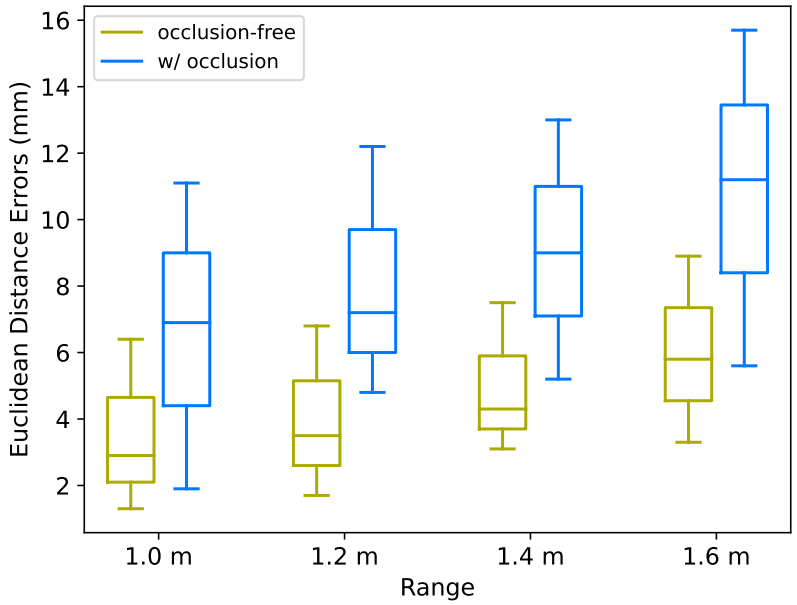

Figure 5.10 Indoor performance evaluation of the active laser-camera scanning (ALACS) system using single laser projections to the Qualisys marker center from different distances for occlusion-free (120 cases) and occlusion-present (120 cases) situations.

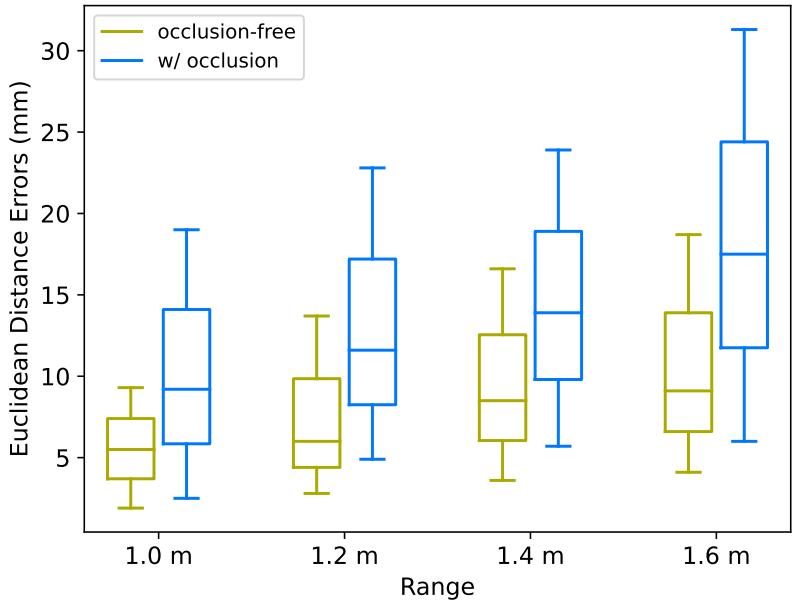

Figure 5.11 Indoor performance evaluation of the active laser-camera scanning (ALACS) system using multiple laser line projections to the Qualisys marker from different distances under the occlusion-free (120 cases) and occlusion-present (120 cases) situations.

1.0 m, 1.2 m, 1.4 m, and 1.6 m, respectively. While these errors are significantly larger compared to those obtained for single laser lines in the first experiment (see Figure 5.10),

the ALACS is still expected to meet our robot localization accuracy requirements of 20 mm when the distance is less than 1.6 m, the maximum working distance designed for our harvesting robot [131].

Furthermore, I also performed a 3D localization comparison between ALACS and the commercial RGB-D camera RealSense D435i, where the latter is commonly used in the harvesting robots developed by other researchers. In the indoor environment, I still used the Qualysis Motion Tracking System to benchmark the results and test the localization from ALACS with multiple laser projections and RealSense D435i with occlusions at different distances. Table 5.2 shows the ALACS system significantly outperformed the RealSense benchmark for different distances among 120 cases.

Since the Qualysis system cannot provide accurate measurements in the outdoor environment due to the varying light condition, I used the positions generated from ALACS and D435i to operate our robotic system [134] to to determine whether fruits would be attached to vacuum-based end-effector. The attachment rate was used as an indirect metric for evaluating the localization results of both ALACS and RealSense D435i. Since our vacuum-based end-effector can tolerate localization errors up to 20 mm, a measured localization error larger than 20 mm would be considered a failed detachment. I tested 100 apples under cloudy and 50% occlusion rate in a research orchard at Michigan State University's Horticultural Teaching and Research Center in Holt, MI. Our results showed that ALACS achieved a 95% detachment rate, whereas the RealSense D435i only had a 71% success rate.

Table 5.2 Comparison of average localization errors (mm) between ALACS and RealSense D435i at different distances.

Sensor \ Range	1.0m	1.2m	1.4m	1.6m
RS D435i	16.0(±7.1)	17.3(±8.0)	19.5(±9.3)	21.5(±11.2)
ALACS	**6.9**(±5.2)	**7.2**(±5.8)	**9.0**(±6.5)	**11.2**(±7.8)

Results from the three indoor and outdoor experiments have demonstrated that the ALACS system has had significantly enhanced performance for apple localization, compared to RealSense D435i.

5.5 Summary of the Chapter

In this chapter, a novel Active Laser-Camera Scanning (ALACS) system was developed for robust apple 3D localization. The proposed LLE method provided precise laser line pattern extractions with an average displacement of 1 pixel under complex fruit morphology, over-exposure, and occlusion conditions. For the apple 3D localization, ALACS was able to achieve average errors of $9.2 - 17.5$ mm at distances ranging from 1.0m to 1.6m, which are significantly better, compared to the widely adopted commercial RGB-D camera RealSense D435i. ALACS also demonstrated superior performance for fruit detachment in an apple orchard, when it was tested with our harvesting robot equipped with a vacuum-based end effector.

CHAPTER 6

SKESEGNET: SKELETON-LEAD SEGMENTATION NETWORK FOR BRANCH SEGMENTATION

In this chapter, I delve into the realm of panoptic segmentation, presenting a comprehensive overview of the field. I introduce our novel approach, SkeSegNet, aimed at advancing branch segmentation by leveraging skeletal information for enhanced accuracy and robustness. Furthermore, I explore the by-product of SkeSegNet to generate 3D branch representations.

6.1 Introduction

In fruit detection and robotic harvesting applications, it is crucial to have a comprehensive understanding of the orchard environment, including not only the fruits but also other elements, such as branches, leaves, and the overall tree structure. Panoptic segmentation, a computer vision technique that combines instance segmentation and semantic segmentation, can provide valuable information about the orchard scene by simultaneously segmenting and classifying individual objects and background regions. By integrating panoptic segmentation into fruit detection systems, I can obtain richer visual details to guide robotic systems more effectively, improving the efficiency and safety of the harvesting process.

Panoptic segmentation aims to provide a holistic understanding of the scene by segmenting and classifying every pixel in the image into various categories, such as fruits, branches, leaves, and background. This level of detail can be particularly useful in fruit detection applications for several reasons. 1) Improved Fruit Detection: Panoptic segmentation can help distinguish fruits from other scene elements, such as leaves and branches, reducing false positives and improving overall detection accuracy; 2)Robotic Navigation and Manipulation: By providing detailed information about the orchard structure, panoptic segmentation can enable robots to navigate and manipulate their environment more effectively, avoiding obstacles and minimizing damage to the trees and fruits; 3) Scene Understanding for fruit status: Panoptic segmentation can contribute to a better understanding of the entire orchard scene,

facilitating more accurate fruit ripeness checking [66].

Several techniques and algorithms have been proposed for panoptic segmentation in computer vision, leveraging advances in deep learning and convolutional neural networks (CNNs). Panoptic Feature Pyramid Networks (Panoptic-FPN) [58] is a unified, end-to-end trainable architecture that combines instance segmentation and semantic segmentation tasks, producing panoptic segmentation results in a single forward pass through the network. Panoptic-DeepLab [20] is another end-to-end trainable architecture for panoptic segmentation, which employs an efficient dual-path architecture to handle both instance and semantic segmentation tasks. Mask R-CNN: Mask R-CNN is a popular instance segmentation method that can be extended to perform panoptic segmentation by combining it with semantic segmentation techniques. Adapting these methods for fruit detection applications in orchards may involve fine-tuning the segmentation models on annotated datasets containing fruit, branch, leaf, and background classes, ensuring that the models can accurately segment and classify these elements in real-world orchard environments.

I adapt the Panoptic-DeepLab architecture for orchard environments and trained it on our customized dataset, which consists of images with five distinct labeled classes, including apples, branches, foliage, sky, and ground. I propose a Skeleton Segmentation Network to improve the accuracy of branch segmentation since branches are the main obstacles against apples. Using the skeleton branches, 3D branches are quickly generated combined with the depth map, which provide obstacle data for our robot's planning algorithm. Then, I evaluate the performance against state-of-the-art panoptic segmentation models and demonstrate superior performances. The contributions of this work are highlighted as follows:

1. An image annotation tool, PicA, is developed and open-sourced to alleviate the burden of manual annotation by leveraging the concept of superpixels and pre-trained models associated with panoptic segmentation annotation.
2. Skeleton-lead Segmentation Network (SkeSegNet) is proposed to address the challenges of segmenting complex branches. SKeSegNet is evaluated and outperformed 4 state-

of-the-art panoptic segmentation models.

3. The SkeSegNet generates 3D branches for efficient obstacle avoidance using depth map.

6.2 Data Annotation

Based on the previous work (see Chapter 3), I select 167 images to compose orchard segmentation dataset. I next processed these collected raw orchard images into formats that can be used to train and evaluate deep networks. Specifically, there are totally five categories, including branch, foliage, ground, sky and apples in the images, to be annotated in pixel level. In panoptic segmentation datasets, instance annotation and semantic annotation are split into different tasks, which are labor-intensive and time-consuming. To address the challenges associated with panoptic segmentation annotation, a novel image annotation tool named PicA[1] has been developed, particularly when dealing with complex scenes. PicA has been designed to alleviate the burden of manual annotation by leveraging the concept of superpixels and pre-trained models to expedite the annotation process.

Superpixels, which are compact and coherent image regions, serve as a fundamental component in PicA's annotation strategy. By segmenting the input images into superpixels, PicA simplifies the annotation process by allowing annotators to work on smaller, more manageable regions of the image at a time. This approach not only enhances the efficiency of the annotation process but also reduces the potential for human error by focusing annotators' attention on local regions. Additionally, PicA harnesses the power of pre-trained models to further streamline the annotation workflow. These models, trained on large and diverse datasets, possess the capability to generate preliminary segmentation results with a high degree of accuracy. PicA integrates these pre-trained results as a starting point for annotators, reducing the amount of manual labor required. This not only accelerates the annotation process but also ensures that annotators are provided with a reliable foundation for further refinement and correction. Figure 6.1 shows the annotation results based on PicA.

[1]The image annotation tool PicA is open-sourced at https://github.com/pengyuchu/picA.

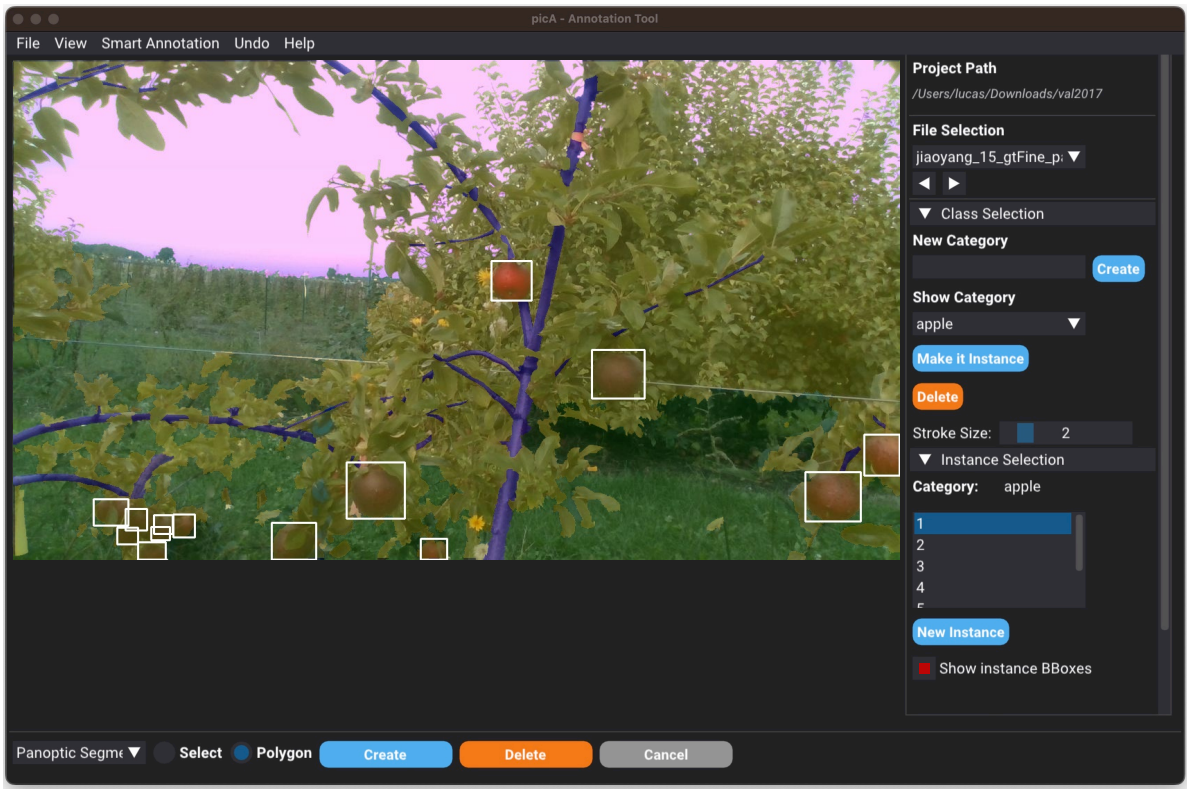

Figure 6.1 PicA: An image annotation tool. In this workspace, a panoptic segmentation project is created and an image is annotated with superpixels. After checking BBoxes, all of instances are shown using white boxes.

6.3 Panoptic-DeepLab: Panoptic Segmentation

Panoptic-DeepLab [20] is conceptually simple, adopting dual-ASPP and dual-decoder modules specific to semantic segmentation and instance segmentation, respectively. Panoptic-DeepLab uses a fast bottom-up baseline (see Figure 6.2) and requires only three loss functions during training, and introduces extra marginal parameters as well as additional slight computation overhead when building on top of a modern semantic segmentation model. In the apple orchard segmentation, the output will be the fusion of scenes (i.e., branch, foliage, ground and sky) and apples.

6.3.1 Architecture of Panoptic-DeepLab

Panoptic-DeepLab (see Figure 6.3) consists of four components: (1) an encoder backbone shared for both semantic segmentation and instance segmentation, (2) decoupled ASPP modules and (3) decoupled decoder modules specific to each task, and (4) task-specific

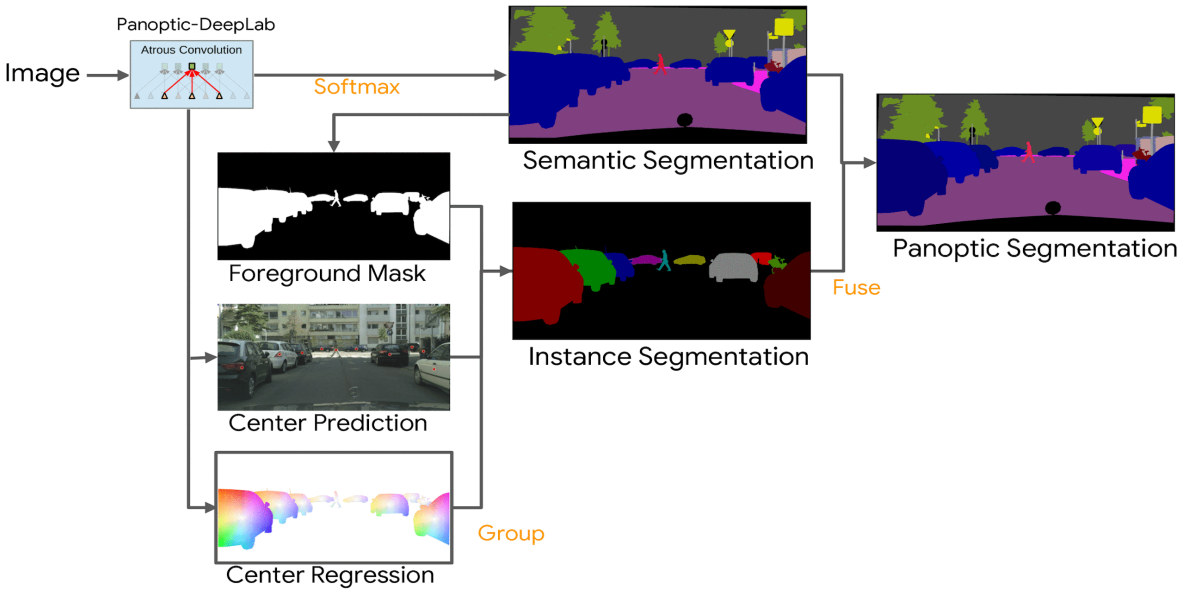

Figure 6.2 Panoptic-DeepLab [20] predicts three outputs: semantic segmentation, instance center prediction and instance center regression. Class-agnostic instance segmentation, obtained by grouping predicted foreground pixels to their closest predicted instance centers, is then fused with semantic segmentation by majority-vote rule to generate final panoptic segmentation.

prediction heads.

Basic architecture: The encoder backbone is adapted from an ImageNet-pretrained neural network paired with atrous convolution for extracting denser feature maps in its last block. Motivated by [21, 81], Panoptic-DeepLab employs separate ASPP and decoder modules for semantic segmentation and instance segmentation, respectively, based on the hypothesis that those two branches requires different contextual and decoding information, which is empirically verified in the following section. The light-weight decoder module follows DeepLabV3+ [18] with two modifications: (1) Panoptic-DeepLab introduces an additional low-level feature with output stride 8 to the decoder, thus the spatial resolution is gradually recovered by a factor of 2, and (2) in each upsampling stage Panoptic-DeepLab applys a single 5×5 depthwise-separable convolution [51].

Semantic segmentation head: Panoptic-DeepLab employs the weighted bootstrapped cross entropy loss, proposed in [126], for semantic segmentation, predicting both 'thing' and 'stuff' classes. The loss improves over bootstrapped cross entropy loss [121, 96, 88] by

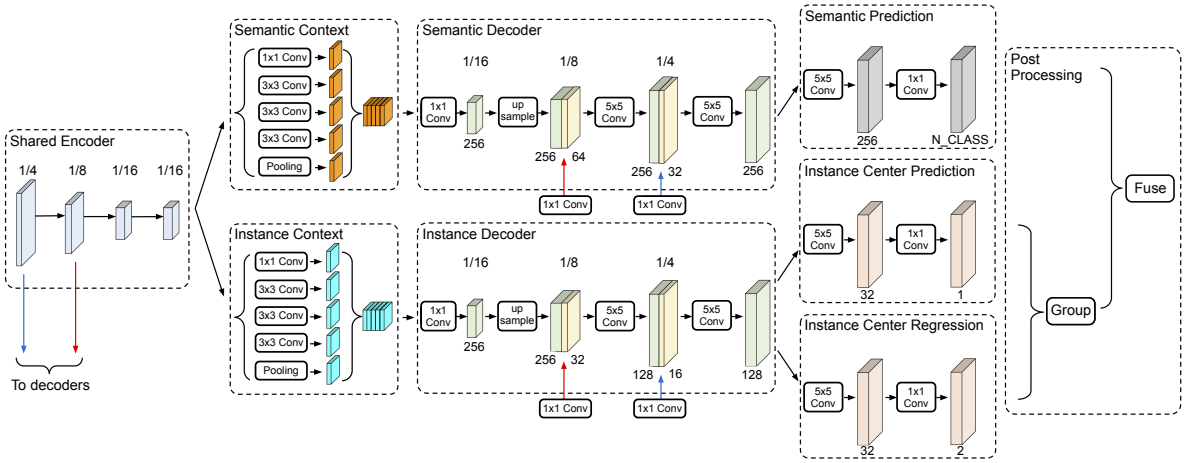

Figure 6.3 Panoptic-DeepLab [20] adopts dual-context and dual-decoder modules for semantic segmentation and instance segmentation predictions. They applys atrous convolution in the last block of a network backbone to extract denser feature map. The Atrous Spatial Pyramid Pooling (ASPP) is employed in the context module as well as a light-weight decoder module consisting of a single convolution during each upsampling stage. The instance segmentation prediction is obtained by predicting the object centers and regressing every foreground pixel (i.e., pixels with predicted 'thing' class) to their corresponding center. The predicted semantic segmentation and class-agnostic instance segmentation are then fused to generate the final panoptic segmentation result by the "majority vote" proposed by DeeperLab.

weighting each pixel differently.

Class-agnostic instance segmentation head: Motivated by Hough Voting [7], Panoptic-DeepLab represents each object instance by its center of mass. For every foreground pixel (i.e., pixel whose class is a 'thing'), they further predicts the offset to its corresponding mass center. During training, groundtruth instance centers are encoded by a 2-D Gaussian with standard deviation of 8 pixels. In particular, they adopts the Mean Squared Error (MSE) loss to minimize the distance between predicted heatmaps and 2D Gaussianencoded groundtruth heatmaps. They uses $L1$ loss for the offset prediction, which is only activated at pixels belonging to object instances. During inference, predicted foreground pixels (obtained by filtering out background 'stuff' regions from semantic segmentation prediction) are grouped to their closest predicted mass center, forming the class-agnostic instance segmentation results, as detailed below.

6.3.2 Panoptic Segmentation

During inference, Panoptic-DeepLab uses an extremely simple grouping operation to obtain instance masks, and a highly efficient majority voting algorithm to merge semantic and instance segmentation into final panoptic segmentation.

Simple instance representation: Panoptic-DeepLab simply represents each object by its center of mass, $\{C_n : (i_n, j_n)\}$. To obtain the center point prediction, they first performs a keypoint-based non-maximum suppression (NMS) on the instance center heatmap prediction, essentially equivalent to applying max pooling on the heatmap prediction and keeping locations whose values do not change before and after max pooling. Finally, a hard threshold is used to filter out predictions with low confidence, and only locations with top-k highest confidence scores are kept. In experiments, Panoptic-DeepLab uses max-pooling with kernel size 7, threshold 0.1, and $k = 200$.

Simple instance grouping: A simple instance center regression is used to to obtain the instance id for each pixel. For example, consider a predicted 'thing' pixel at location (i, j), an offset vector $\mathcal{O}(i, j)$ is predicted to its instance center. $\mathcal{O}(i, j)$ is a vector with two elements, representing the offset in horizontal and vertical directions, respectively. The instance id for the pixel is thus the index of the closest instance center after moving the pixel location (i, j) by the offset $\mathcal{O}(i, j)$. That is,

$$\hat{k}_{i,j} = \arg\min_{k} ||\mathcal{C}_k - ((i, j) + \mathcal{O}(i, j))||, \tag{6.1}$$

where $\hat{k}_{i,j}$ is the predicted instance id for pixel at (i, j). I use semantic segmentation prediction to filter out 'stuff' pixels whose instance id are always set to 0.

Efficient merging: Given the predicted semantic segmentation and class-agnostic instance segmentation results, I adopt a fast and parallelizable method to merge the results, following the "majority vote" principle proposed in DeeperLab [77]. In particular, the semantic label of a predicted instance mask is inferred by the majority vote of the corresponding predicted semantic labels. This operation is essentially accumulating the class label histograms, and

thus is efficiently implemented in GPU, which takes only 3 ms when operating on a 1025×2049 input.

6.3.3 Instance Segmentation

Panoptic-DeepLab can also generate instance segmentation predictions as a by-product. To properly evaluate the instance segmentation results, one needs to associate a confidence score with each predicted instance mask. Previous bottom-up instance segmentation methods use some heuristics to obtain the confidence scores. For example, DWT [6] and SSAP [39] use an average of semantic segmentation scores for some easy classes and use random scores for other harder classes. Additionally, they remove masks whose areas are below a certain threshold for each class. On the other hand, the Panoptic-DeepLab does not adopt any heuristic or post processing for instance segmentation. Motivated by YOLO [93], they computes the class-specific confidence score for each instance mask as $Score_{Object} \times Score_{Class}$, where $Score_{Object}$ is unnormalized objectness score obtained from the class-agnostic center point heatmap, and $Score_{Class}$ is obtained from the average of semantic segmentation predictions within the predicted mask region.

6.4 Skeleton-lead Segmentation Network for Branch Detection

The segmentation of complex structures such as branches poses a significant challenge due to their inherent characteristics of spanning over a large spatial extent. Traditional segmentation approaches may struggle to delineate such structures effectively, as they tend to rely on strict boundaries and may overlook the interconnected nature of branches. Therefore, I introduce an alternative strategy is to treat these branches as a combination of segments and skeletonize them. This approach leverages the idea that branches can be represented as a network of interconnected segments and a central skeleton. To prepare for the annotation of the data, I only add skeleton annotations based on existing branch annotation. Annotators only draw skeletons represented by two ends V^s and V^e (See Figure 6.4). This preparatory step is essential for training and evaluating segmentation models effectively.

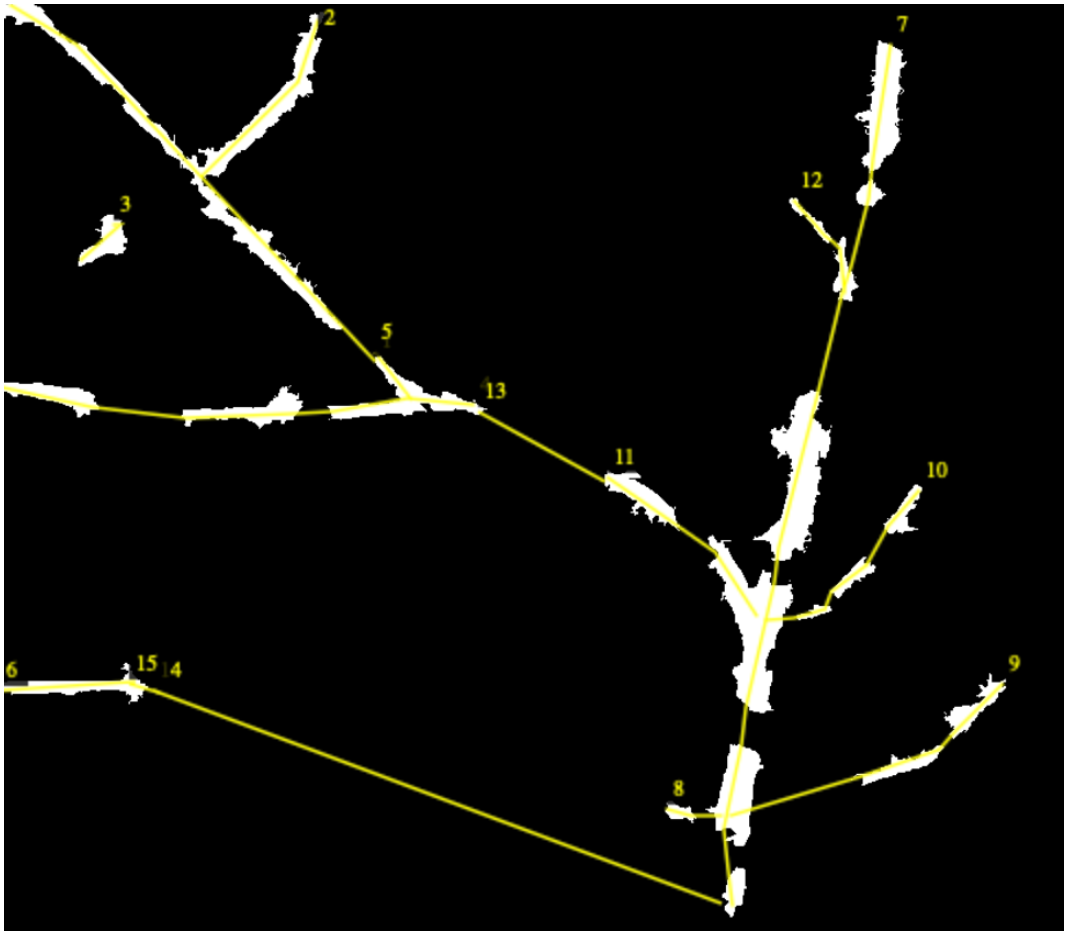

Figure 6.4 A sample of skeleton annotation based on branch annotation. Each line is drawn based on a pair of the skeleton's ends (V^s, V^e).

6.4.1 Architecture

To facilitate the orchard segmentation process, I propose the Skeleton-lead Segmentation Network (SkeSegNet) and introduce it to the Panoptic-Deeplab, in order to address the challenges of segmenting complex branches. SkeSegNet (see Figure 6.5) comprises two key components: the skeleton decoder and the skeleton predictor. The skeleton decoder is responsible for extracting the central skeleton of the branch structure, which serves as a fundamental representation of its topology. The skeleton predictor, on the other hand, generates segment centers and boundaries based on the extracted skeleton, effectively dividing the branch into individual segments. Together, these components work in parallel to achieve a comprehensive and accurate segmentation of complex branching structures, overcoming the limitations of traditional segmentation approaches.

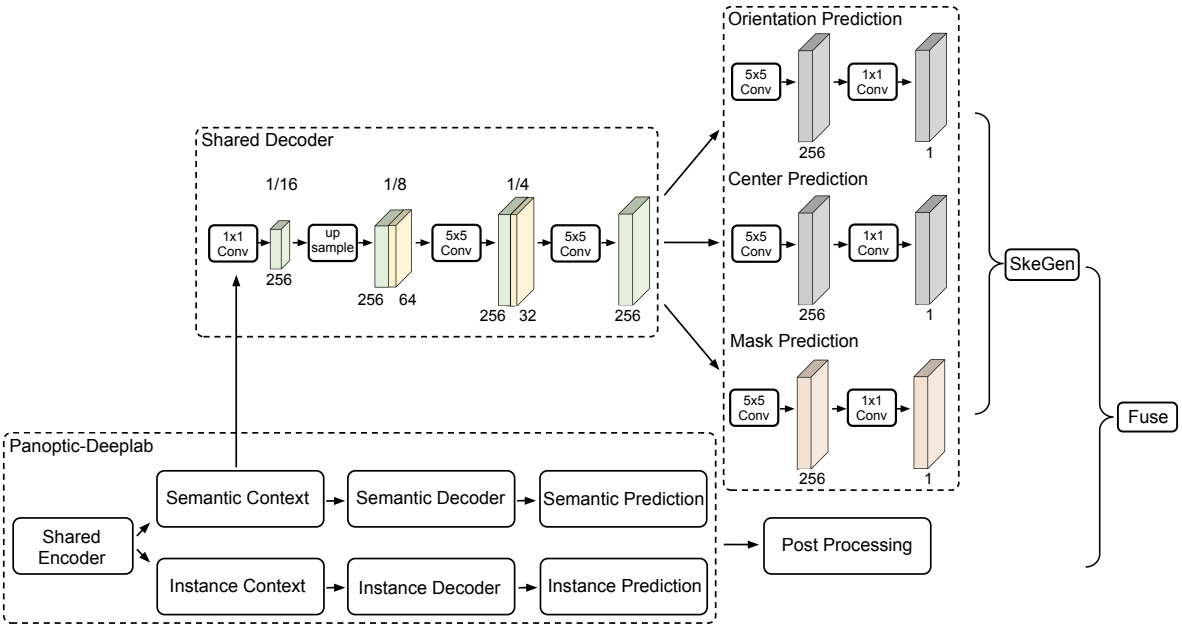

Figure 6.5 Skeleton-lead Segmentation Network (SkeSegNet) consists of Skeleton Decoder and Skeleton Predictor. The decoder uses a light-weight decoder module consisting of a single convolution during each upsampling stage. The skeletons prediction is obtained by predicting the skeleton centers, skeleton masks, and the skeleton orientations. The operation SkeGen converts them to skeletons, and uses our designed fusion methods to improve the branch class probability map.

Skeleton Decoder: After the output from Panoptic-Deeplab's encoder backbone, which is an ImageNet-pretrained neural network paired with atrous convolution for extracting denser feature maps, I applied a light-weight decoder [18] module as the skeleton decoder, to extract skeleton features. The skeleton decoder introduces an additional low-level feature with output stride 8 to the decoder, thus the spatial resolution is gradually recovered by a factor of 2. Additionally, in each upsampling stage the skeleton decoder applys a single 5×5 depthwise-separable convolution [51].

Skeleton Predictor: In the deep convolutional networks, graph data is hard to represent in feature maps. Hence, I split each branch segment into three elements: branch center, branch orientation and branch pixellation, associated with three predictors. The skeleton mask predictor follows a semantic predictor head using a stack of 5×5 and 1×1 depthwise-separable convolution, which predicts skeleton mask. The skeleton center predictor and orientation predictor gives center and orientation (normalized into $(0, 1]$). SkeSegNet repre-

sents each skeleton by its center, its orientation and its pixellation (with the identical width 5). For every pixel, they further predicts the orientaion to its corresponding skeleton center. During training, groundtruth line-obejct centers $((V_x^s + V_x^e)/2, (V_y^s + V_y^e)/2)$ are encoded by a 2-D Gaussian with standard deviation of 5 pixels, and groundtruth line-obejct orientation is calculeted by $\arctan(\frac{V_y^s - V_y^e}{V_x^s - V_x^e})$ for each skeleton pixel. In particular, they adopts the Mean Squared Error (MSE) loss to minimize the distance between predicted heatmaps and 2D Gaussian-encoded groundtruth heatmaps. They uses $L1$ loss for the oriention prediction, which is only activated at pixels belonging to skeleton instances.

SkeGen: SkeGen employs a unique approach to transform each skeleton center into a line segment. This transformation process relies on two key components: the skeleton orientations and the skeleton mask associated with each skeleton. By combining the information from the orientation map and the skeleton mask, SkeGen is able to systematically convert each skeleton center into a precisely defined line segment. For each center, SkeGen follows its orientation in double sides to generate a line until there is no pixel around the end with 10-pixel offsets. Overall, SkeGen transforms centers to skeleton using paris of (V^s, V^e).

After I have skeleton branches, I can fuse the skeleton map into semantic prediction map from the Panoptic-Deeplab to enhance branch probability. I fuse the semantic mask in branch class $\mathcal{P}$ and skeleton map $\mathcal{S}$ with t-width drawing. The fused results $\hat{\mathcal{P}}$ is followed.

$$\hat{\mathcal{P}} = \lambda_1 \cdot \mathcal{P} \cdot \mathcal{S} + \lambda_2, \tag{6.2}$$

where λ_1, λ_2 and t are fused hyperparameters, which will be discussed in Section 6.6.

6.4.2 SkeSegNet Training

In the training process, I convert the skeleton ground truth into three parts: mask, centers and orientations, to feed the SkeSegNet (see Figure 6.6). The skeleton mask is generated by drawing each skeleton with 5 pixels. The skeleton orientation map assign each calculated orientation to its corresponding mask. The skeleton center map assign the position of each skeleton's center as 1.

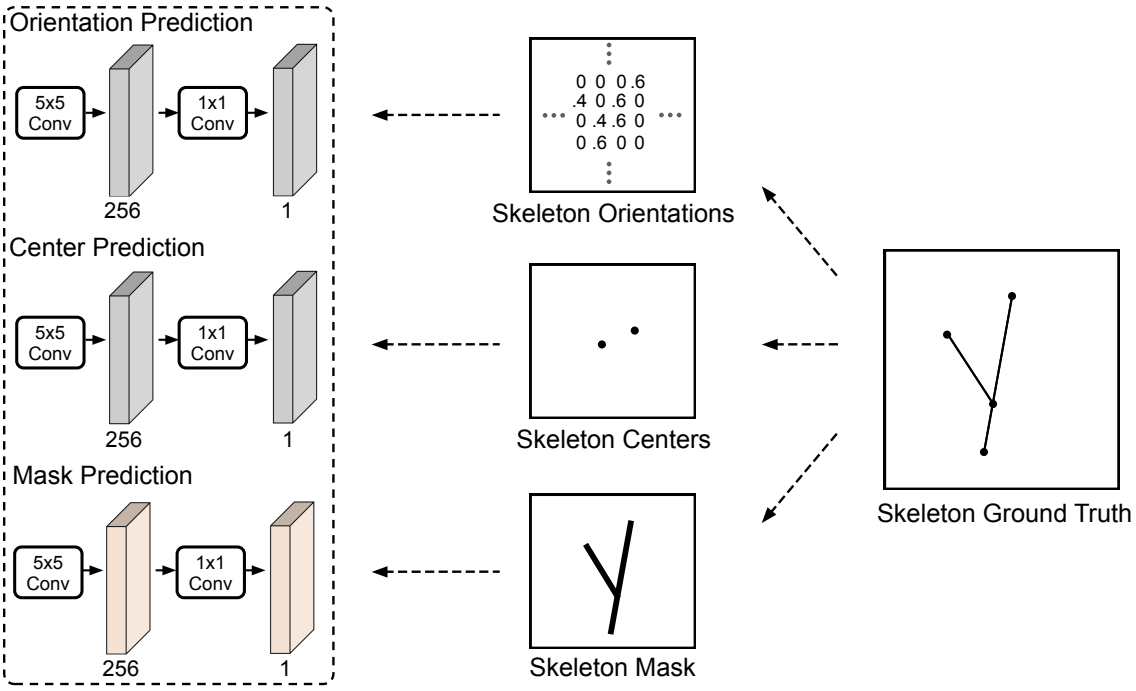

Figure 6.6 During the training period, the skeleton ground truth are converted to skeleton mask, skeleton centers and skeleton orientations for calculating losses with SkeSegNeet predictors. The skeleton mask is generated by drawing each skeleton with t($t = 5$) pixels. The skeleton orientations are composed by assigning each calculated orientation to its corresponding mask. The skeleton centers are composed by assign the position of each skeleton's center as 1.

To avoid affecting panoptic segmentation results, I use light weights on SkeSegNet when training with Panoptic-DeepLab together. Panoptic-DeepLab is trained with three loss functions: weighted bootstrapped cross entropy loss for semantic segmentation head ($\mathcal{L}_{sem}$) [126]; MSE loss for center heatmap head ($\mathcal{L}_{heatmap}$) [111]; and $L1$ loss for center offset head ($\mathcal{L}_{offset}$) [85]. The final loss $\mathcal{L}_{pan}$ is computed as follows:

$$\mathcal{L}_{pan} = \lambda_{sem}\mathcal{L}_{sem} + \lambda_{heatmap}\mathcal{L}_{heatmap} + \lambda_{offset}\mathcal{L}_{offset}. \tag{6.3}$$

Specifically, I set $\lambda_{sem} = 1$, $\lambda_{heatmap} = 200$, and $\lambda_{offset} = 0.01$, to make sure the losses are in the similar magnitude.

On the other hand, SkeSegNet is trained with three loss functions with $L1$ sum loss for each skeleton maps: $\mathcal{L}_{mask}, \mathcal{L}_{center}$ and $\mathcal{L}_{orient}$. the total loss $\mathcal{L}_{ske}$ is computed with as follows:

$$\mathcal{L}_{ske} = \lambda_{mask}\mathcal{L}_{mask} + \lambda_{center}\mathcal{L}_{center} + \lambda_{orient}\mathcal{L}_{orient}, \tag{6.4}$$

where $\lambda_{mask} = 0.1$, $\lambda_{center} = 50$, and $\lambda_{orient} = 1$ are set up to make sure the losses are in the similar magnitude. The final loss function is drawn as follows:

$$\mathcal{L} = \lambda_{ske}\mathcal{L}_{ske} + \mathcal{L}_{pan}, \tag{6.5}$$

where $\lambda_{ske} = 0.2$ is the light weight to ease skeleton's affect on other categories.

6.5 Three-dimensional Structured Branch Generation

The utilization of by-product skeletons generated by SkeSegNet offers a valuable advantage in the process of constructing 3D branches. These by-product skeletons represent significant structural information within the image. Leveraging this information, I can efficiently create 3D representations of branches in combination with depth maps.

To construct 3D branches, the length and width of branches are obtained at first. The length can be calculated based the Euclidean distance between two ends of an skeleton. To estimate the width, I select 10 points for each skeleton and follow the perpendicular direction of skeleton to find out the branch pixels. The maximum of them is treated as the skeleton's width. With the related depth map, I transform the 2D branch to the 3D branch using the following equation,

$$\begin{aligned} z &= d, \\ x &= u * f/d, \\ y &= v * f/d, \\ w &= p * f/d, \end{aligned} \tag{6.6}$$

where (u, v) and (x, y, z) are respectively 2D positions and 3D positions of the branch's ends. The p is the branch's width in pixels and the w is its estimated width in the world coordinate. An example of 3D branch estimation period is shown in Figure 6.7.

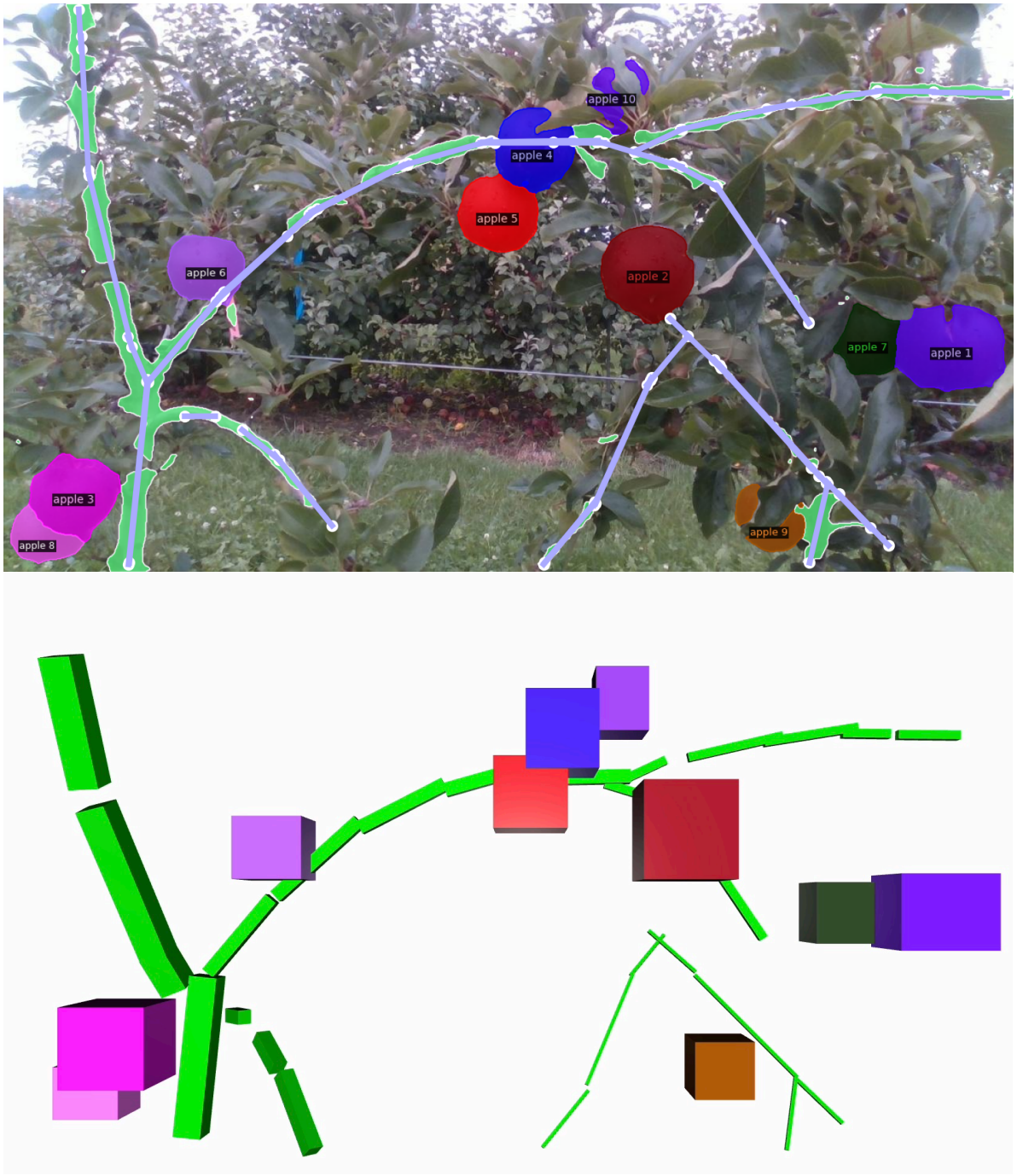

Figure 6.7 Three-dimension representatives of apple and branches. I utilize by-product skeletons from SkeSegNet to construct 3D branches and apples using the related depth map. (a) The predicted skeletons are visualized using blue lines with white circle ends and branch mask is visualized in green. (b) The generated 3D branches are showed in green at the front view and the apples are presented using cuboids with different colors.

The integration of by-product skeletons simplifies and expedites the 3D reconstruction process. Rather than relying solely on depth maps, which may have limitations in accu-

rately capturing fine object boundaries and complex structures, the by-product skeletons act as complementary information sources. The 3D representatives of branches will ease the obstacle avoidance planning.

6.6 Experimental Results

In this section, I present the experimental results of our proposed method, focusing on two main tasks: branch segmentation and panoptic segmentation. Additionally, I compare our approach with the state-of-the-art methods in these domains. Furthermore, I look into the effectiveness of the generated 3D branches in branch avoidance.

I report the Precision, Recall and F1-score to evaluate the branch and panoptic segmentation results. All my models are trained using PyTorch on RTX 2080Ti. I use an initial learning rate of 0.001, fine-tune the batch normalization parameters, perform random scale data augmentation during training, and optimize with Adam without weight decay. I resize the images to 1025 pixels at the longest side and train my models with crop size 1025×1025 with batch size 64. I set training iterations to $200K$.

6.6.1 Branch Segmentation Evaluation

In the evaluation of branch segmentation, I only train branch class and then the task is converted to semantic segmentation.

To get the better results, I use different fusion combination to determine the best fusion parameters (see Equation 6.2). I select t from $0-10$ at a step of 1, λ_1 and λ_2 from $0-5.0$ at a step of 0.1. The best parameters are $t=4$, $\lambda_1=2.5$, $\lambda_2=0.6$ and the results are shown in Table 6.1. Compared with Panoptic-Deeplab, the added SkeSegNet improves a precision of 8.4%, a recall of 8.3%, and a F1-score of 8.5%.

Table 6.1 Performance comparison between the baseline Panoptic-Deeplab and our proposed SkeSegNet for only branch segmentation.

	Precision	Recall	F1-score
Panoptic-Deeplab [20]	62.0	51.2	56.0
SkeSegNet (Ours)	**70.4**	**59.5**	**64.5**

Additionally, I also evaluate other SOTA methods on our own dataset for branch segmentation. Table 6.2 summarizes the performance of my method compared to SOTA methods on the dataset.

Table 6.2 Performance comparison between the-state-of-art and our proposed SkeSegNet for only branch segmentation.

	Precision	Recall	F1-score	FPS
Panoptic-Deeplab [20]	62.0	51.2	56.0	2
Yolov8 [53]	41.2	34.0	37.5	**20**
MaskFormer (ViT) [19]	55.0	48.6	51.8	0.3
MaskFormer (Swin-L) [70]	57.0	50.5	53.7	0.3
SkeSegNet (Ours)	**70.4**	**59.5**	**64.5**	2

Our approach consistently outperforms existing methods in branch segmentation, as demonstrated by the superior scores on F1-score. This indicates the effectiveness of our proposed method in accurately segmenting branches.

6.6.2 Panoptic Segmentation Evaluation

Since our final task to obtain the whole orchard segmentation, I evaluate my method in panoptic segmentation to ensure that SkeSegNet would not affect apples or other stuff (e.g. foliage) segmentation results. I evaluate other SOTA methods on our own dataset for branch segmentation. Table 6.3 summarizes the performance of my method compared to SOTA methods on the dataset. Since the apple and branch segmentation are essential for our work, I focus on the comparison between the baseline model and SkeSegNet. I can see my method still outperform the baseline model with a 5.4, 3.8 and 3.4 increase respectively in precision, recall and F1-score. Besides, the introduced SkeSegNet does not affect other categories' segmentation results.

6.6.3 By-product Skeletons Evaluation

To evaluate the effectiveness of the generated skeleton branches, I conduct experiments using four metrics: center error E_c, length error $Error_l$, orientation error E_o and width error E_w. The metrics are calculated as follows.

Table 6.3 Performance comparison in Precision (P), Recall (R) and F1-score (F1) between the-state-of-art and our proposed SkeSegNet for five categories panoptic segmentation.

	branch			apple			all			FPS
	P	R	F1	P	R	F1	P	R	F1	
Panoptic-Deeplab [20]	62.0	51.2	56.0	**67.0**	60.0	63.3	82.4	**75.4**	**80.9**	2
Yolov8 [53]	41.6	34.2	37.5	63.5	59.2	61.3	65.0	74.9	69.6	**20**
MaskFormer (ViT) [19]	55.0	48.6	51.8	67.0	61.4	**64.0**	79.1	73.0	75.9	0.3
MaskFormer (Swin-L) [70]	57.0	50.5	53.7	66.8	**61.5**	63.9	79.9	73.2	76.3	0.3
SkeSegNet (Ours)	**67.4**	**55.0**	**60.6**	66.8	60.2	63.3	**82.7**	75.1	80.8	2

$$
\begin{aligned}
E_c &= ||\hat{V_c} - V_c||, \\
E_l &= |\hat{L} - L|, \\
E_o &= |\hat{O} - O|, \\
E_w &= |\hat{W} - W|,
\end{aligned}
\tag{6.7}
$$

where V_c, L, O, W are the center, length, orientation and width for groundtruth branches, while $\hat{V_c}, \hat{L}, \hat{O}, \hat{W}$ are predicted ones from SkeSegNet. The Average Error AE is calculated by $\frac{1}{n}E$. The results are shown in Table 6.4.

Table 6.4 Average Errors (AE) for by-product skeleton branches from SkeSegNet. In 2D errors, I calculate errors in pixels. In 3D errors, I convert each point into 3d positions (cm) using the corresponding depth map (from Realsense D435i) over 120 images.

	AE_c	AE_l	AE_o	AE_w
2D	2.6 px	13.2 px	4.0 deg	1.0 px
3D	3.5 cm	18.6 cm	5.2 deg	1.4 cm

This evaluation shows that the predicted skeleton branches have a small error in center error, orientation error and width error with 2.6, 4.0, and 1.2 respectively. The error in length is larger than other metric with an error of 13.2, which is caused by the disagreement between the prediction and ground truth. Due to random of skeleton generation, the skeletons are not always predicted with the identical ones and can be split into different skeletons.

6.7 Summary of the Chapter

SkeSegNet, a novel method, is proposed for improving branch segmentation based on existing panoptic segmentation work Panoptic-Deeplab. Through rigorous experiments, I

demonstrate its effectiveness in accurately segmenting branches, outperforming state-of-the-art methods with a 64.5 F1-score. I also applied this design on a panoptic segmentation task including five categories (i.e., branch, foliage, ground, sky and apples). The results shows that our proposed method still improve 5% branch F1-score compared with the baseline in panoptic segmentation task. Additionally, these skeleton branches generated by SkeSegNet show average errors 2.6 px, 13.2 px, 4.0 degree and 1.2 px in center, length, orientation and width respectively. The generated 3D skeletons combined with the depth map serve as a valuable resource for the design of planning algorithms, opening up new possibilities in the realm of branch avoidance and navigation.

However, the skeleton branch generation still has drawbacks, including that the average precision of 2D skeleton branch is still low, and the generated 3D branches have large errors in the z axis compared with ideal cases. In future studies, I will optimize my methods, e.g., using branch segmentation to improve skeleton prediction, and enhance 2D skeleton generation. Besides, I will utilize high-precision depth sensor to localize branches and evaluate our 3D skeleton branches with more comprehensive experiments.

CHAPTER 7

CONCLUSION AND FUTURE WORK

The importance of accurate fruit detection and localization algorithms in challenging environments cannot be overstated. As I continue to face evolving agricultural and environmental challenges, the need for precise and efficient methods for harvesting fruits becomes increasingly vital. These algorithms not only enhance productivity but also contribute to sustainable farming practices by reducing waste and resource consumption.

Throughout this dissertation, I presented two novel techniques, suppression Mask RCNN and O2RNet, aimed at addressing the challenges of fruit detection. While the suppression Mask RCNN improve the precision for apple detection, O2RNet demonstrates superior performance in detecting and isolating apples in complex orchard environments, effectively handling occlusions and varying lighting conditions. By utilizing a robust deep learning architecture, O2RNet can adapt to different apple morphologies and improve the overall accuracy of fruit detection. ALACS, on the other hand, employs active laser scanning to generate high-resolution 3D data of the apple's surface, overcoming the limitations of passive imaging techniques. With the combination of the bidirectional relative color enhancement (bRCE) algorithm for laser line extraction and the target position estimation process, ALACS achieves accurate and reliable apple localization. The evaluation results further validate the effectiveness of the ALACS system, showcasing its potential for practical applications in fruit orchards. Lastly, a panoptic segmentation work especially for branch segmentation, called SkeSegNet, represented a significant advancement. This innovative approach not only enhances the accuracy of branch detection but also offers the valuable by-product of generating 3D branch representatives for effective branch avoidance strategies.

Our future work will focus on enhancing 3D skeleton branch generation, fruit tracking, and the classification of disease and ripeness. A high-precision depth sensor will be employed to accurately localize branches, significantly reducing errors in 3D branch generation. This advanced technology promises enhanced accuracy and reliability in our branch modeling

processes. The introduction of a fruit tracking method will prevent repetitive picking, thereby accelerating the harvesting time of our robot. Additionally, the enhanced disease and ripeness classification system will not only aid in early detection of plant diseases but also ensure that fruits are harvested at their peak quality. Through these advancements, I hope to contribute substantially to sustainable agricultural practices, reducing waste and increasing productivity in orchards. Our team is committed to continuous innovation, seeking to integrate the latest technologies to revolutionize fruit farming and set new industry standards.

BIBLIOGRAPHY

[1] Aanis Ahmad, Dharmendra Saraswat, Varun Aggarwal, Aaron Etienne, and Benjamin Hancock. Performance of deep learning models for classifying and detecting common weeds in corn and soybean production systems. *Computers and Electronics in Agriculture*, 184:106081, 2021.

[2] Nikita Andriyanov, Ilshat Khasanshin, Daniil Utkin, Timur Gataullin, Stefan Ignar, Vyacheslav Shumaev, and Vladimir Soloviev. Intelligent system for estimation of the spatial position of apples based on yolov3 and real sense depth camera d415. *Symmetry*, 14(1):148, 2022.

[3] Rajkishan Arikapudi and Stavros G Vougioukas. Robotic tree-fruit harvesting with telescoping arms: a study of linear fruit reachability under geometric constraints. *IEEE Access*, 9:17114–17126, 2021.

[4] Johan Baeten, Kevin Donné, Sven Boedrij, Wim Beckers, and Eric Claesen. Autonomous fruit picking machine: A robotic apple harvester. In *Field and service robotics*, pages 531–539. Springer, 2008.

[5] Chris H Bahnsen, Anders S Johansen, Mark P Philipsen, Jesper W Henriksen, Kamal Nasrollahi, and Thomas B Moeslund. 3d sensors for sewer inspection: A quantitative review and analysis. *Sensors*, 21(7):2553, 2021.

[6] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5221–5229, 2017.

[7] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

[8] Suchet Bargoti and James Underwood. Deep fruit detection in orchards. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3626–3633. IEEE, 2017.

[9] Suchet Bargoti and James P Underwood. Image segmentation for fruit detection and yield estimation in apple orchards. *Journal of Field Robotics*, 34(6):1039–1060, 2017.

[10] Meny Benady and Gaines E Miles. Locating melons for robotic harvesting using structured light. *Paper-American Society of Agricultural Engineers (USA)*, 1992.

[11] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[12] D Bulanon and T Kataoka. Fruit detection system and an end effector for robotic

harvesting of fuji apples. *Agricultural Engineering International: CIGR Journal*, 12(1), 2010.

[13] DM Bulanon, TF Burks, and V Alchanatis. Study on temporal variation in citrus canopy using thermal imaging for citrus fruit detection. *Biosystems Engineering*, 101(2):161–171, 2008.

[14] DM Bulanon, TF Burks, and V Alchanatis. Image fusion of visible and thermal images for fruit detection. *Biosystems engineering*, 103(1):12–22, 2009.

[15] Duke M Bulanon, Colton Burr, Marina DeVlieg, Trevor Braddock, and Brice Allen. Development of a visual servo system for robotic fruit harvesting. *AgriEngineering*, 3(4):840–852, 2021.

[16] Martha Cardenas-Weber, Amots Hetzroni, and Gaines E Miles. Machine vision to locate melons and guide robotic harvesting. *Paper-American Society of Agricultural Engineers (USA)*, 1991.

[17] Arianna Carnevale, Ilaria Mannocchi, Mohamed Saifeddine Hadj Sassi, Marco Carli, Giovanna De Luca, Umile Giuseppe Longo, Vincenzo Denaro, and Emiliano Schena. Virtual reality for shoulder rehabilitation: Accuracy evaluation of oculus quest 2. *Sensors*, 22(15):5511, 2022.

[18] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[19] Zhengsu Chen, Lingxi Xie, Jianwei Niu, Xuefeng Liu, Longhui Wei, and Qi Tian. Visformer: The vision-friendly transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 589–598, 2021.

[20] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12475–12485, 2020.

[21] Bowen Cheng, Yunchao Wei, Honghui Shi, Rogerio Feris, Jinjun Xiong, and Thomas Huang. Revisiting rcnn: On awakening the classification power of faster rcnn. In *Proceedings of the European conference on computer vision (ECCV)*, pages 453–468, 2018.

[22] Phillip Chlap, Hang Min, Nym Vandenberg, Jason Dowling, Lois Holloway, and Annette Haworth. A review of medical image data augmentation techniques for deep learning applications. *Journal of Medical Imaging and Radiation Oncology*, 65(5):545–

563, 2021.

[23] Pengyu Chu, Zhaojian Li, Kyle Lammers, Renfu Lu, and Xiaoming Liu. Deep learning-based apple detection using a suppression mask r-cnn. *Pattern Recognition Letters*, 147:206–211, 2021.

[24] Pengyu Chu, Zhaojian Li, Kaixiang Zhang, Dong Chen, Kyle Lammers, and Renfu Lu. O2rnet: Occluder-occludee relational network for robust apple detection in clustered orchard environments. *Smart Agricultural Technology*, 5:100284, 2023. https://doi.org/10.1016/j.atech.2023.100284.

[25] Zhao De-An, Lv Jidong, Ji Wei, Zhang Ying, and Chen Yu. Design and control of an apple harvesting robot. *Biosystems engineering*, 110(2):112–122, 2011.

[26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[27] Vikram Mutneja Dharampal. Methods of image edge detection: A review. *J. Electr. Electron. Syst*, 4(2):2332–0796, 2015.

[28] Philipe A Dias, Amy Tabb, and Henry Medeiros. Apple flower detection using deep convolutional networks. *Computers in Industry*, 99:17–28, 2018.

[29] LG Divyanth, DS Guru, Peeyush Soni, Rajendra Machavaram, Mohammad Nadimi, and Jitendra Paliwal. Image-to-image translation-based data augmentation for improving crop/weed classification models for precision agriculture applications. *Algorithms*, 15(11):401, 2022.

[30] Rainer G Dorsch, Gerd Häusler, and Jürgen M Herrmann. Laser triangulation: fundamental uncertainty in distance measurement. *Applied optics*, 33(7):1306–1314, 1994.

[31] Arun Kumar Dubey and Vanita Jain. Comparative study of convolution neural network's relu and leaky-relu activation functions. In *Applications of Computing, Automation and Wireless Systems in Electrical Engineering: Proceedings of MARC 2018*, pages 873–880. Springer, 2019.

[32] Abhishek Dutta and Andrew Zisserman. The VIA annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, New York, NY, USA, 2019. ACM.

[33] Fadi A. Fathallah. Musculoskeletal disorders in labor-intensive agriculture. *Applied Ergonomics*, 41(6):738 – 743, 2010. Special Section: Selection of papers from IEA 2009.

[34] Steven A Fennimore and Matthew Cutulle. Robotic weeders can improve weed control options for specialty crops. *Pest management science*, 75(7):1767–1774, 2019.

[35] Longsheng Fu, Fangfang Gao, Jingzhu Wu, Rui Li, Manoj Karkee, and Qin Zhang. Application of consumer RGB-D cameras for fruit detection and localization in field: A critical review. *Computers and Electronics in Agriculture*, 177:105687, 2020.

[36] Longsheng Fu, Yaqoob Majeed, Xin Zhang, Manoj Karkee, and Qin Zhang. Faster r–cnn–based apple detection in dense-foliage fruiting-wall trees using rgb and depth features for robotic harvesting. *Biosystems Engineering*, 197:245–256, 2020.

[37] Karina Gallardo and P. Galinato. 2012 cost estimates of establishing, producing, and packing red delicious apples in washington. FS099E, Washington State University Extension Fact Sheet, 2012.

[38] R Karina Gallardo and Suzette P Galinato. *2012 Cost Estimates of Establishing, Producing, and Packing Red Delicious Apples in Washington*. Washington State University Extension, 2012.

[39] Naiyu Gao, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi Huang. Ssap: Single-shot instance segmentation with affinity pyramid. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 642–651, 2019.

[40] Yuanyue Ge, Ya Xiong, Gabriel Lins Tenorio, and Pål Johan From. Fruit localization and environment perception for strawberry harvesting robots. *IEEE Access*, 7:147642–147652, 2019.

[41] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[42] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[43] A Gongal, Suraj Amatya, Manoj Karkee, Q Zhang, and Karen Lewis. Sensors and systems for fruit detection and localization: A review. *Comput. Electron. Agric.*, 116:8–19, Aug. 2015.

[44] A Gongal, Suraj Amatya, Manoj Karkee, Q Zhang, and Karen Lewis. Sensors and systems for fruit detection and localization: A review. *Computers and Electronics in Agriculture*, 116:8–19, 2015.

[45] Novian Habibie, Aditya Murda Nugraha, Ahmad Zaki Anshori, M Anwar Ma'sum, and Wisnu Jatmiko. Fruit mapping mobile robot on simulated agricultural area in gazebo simulator using simultaneous localization and mapping (slam). In *2017 International*

Symposium on Micro-NanoMechatronics and Human Science (MHS), pages 1–7. IEEE, 2017.

[46] Michael W Hannan and Thomas F Burks. Current developments in automated citrus harvesting. In *2004 ASAE annual meeting*, page 1. American Society of Agricultural and Biological Engineers, 2004.

[47] Qian Hao, Xin Guo, Feng Yang, et al. Fast recognition method for multiple apple targets in complex occlusion environment based on improved yolov5. *Journal of Sensors*, 2023, 2023.

[48] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[50] Cameron J Hohimer, Heng Wang, Santosh Bhusal, John Miller, Changki Mo, and Manoj Karkee. Design and field evaluation of a robotic apple harvesting system with a 3D-printed soft-robotic end-effector. *Trans. ASABE*, 62(2):405–414, 2019.

[51] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[52] Yuhua Jiao, Rong Luo, Qianwen Li, Xiaobo Deng, Xiang Yin, Chengzhi Ruan, and Weikuan Jia. Detection and localization of overlapped fruits application in an apple harvesting robot. *Electronics*, 9(6):1023, 2020.

[53] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, January 2023.

[54] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, Yonghye Kwon, Kalen Michael, Jiacong Fang, Colin Wong, Zeng Yifu, Diego Montes, et al. ultralytics/yolov5: v6. 2-yolov5 classification models, apple m1, reproducibility, clearml and deci. ai integrations. *Zenodo*, 2022.

[55] Hanwen Kang and Chao Chen. Fruit detection and segmentation for apple harvesting using visual sensor in orchards. *Sensors*, 19(20):4599, 2019.

[56] Hanwen Kang and Chao Chen. Fast implementation of real-time fruit detection in apple orchards using deep learning. *Computers and Electronics in Agriculture*, 168:105108, 2020.

[57] Lei Ke, Yu-Wing Tai, and Chi-Keung Tang. Deep occlusion-aware instance segmentation with overlapping bilayers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4019–4028, 2021.

[58] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6399–6408, 2019.

[59] Adam Kortylewski, Ju He, Qing Liu, and Alan L Yuille. Compositional convolutional neural networks: A deep architecture with innate robustness to partial occlusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8940–8949, 2020.

[60] K Krishna and M Narasimha Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3):433–439, 1999.

[61] Steven Lanzisera, David Zats, and Kristofer SJ Pister. Radio frequency time-of-flight distance measurement for low-cost wireless sensor localization. *IEEE Sensors Journal*, 11(3):837–845, 2011.

[62] Nalpantidis Lazaros, Georgios Christou Sirakoulis, and Antonios Gasteratos. Review of stereo vision algorithms: from software to hardware. *International Journal of Optomechatronics*, 2(4):435–462, 2008.

[63] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[64] Christopher Lehnert, Andrew English, Christopher McCool, Adam W. Tow, and Tristan Perez. Autonomous sweet pepper harvesting for protected cropping systems. 2(2):872–879, 2017.

[65] P Levi, A Falla, and R Pappalardo. Image controlled robotics applied to citrus fruit harvesting. In *7th International Conference on Robot Vision and Sensory Controls, Zurich (Switzerland), 2-4 Feb 1988*. IFS Publications, 1988.

[66] Xinye Li and Ding Chen. A survey on deep learning-based panoptic segmentation. *Digital Signal Processing*, 120:103283, 2022.

[67] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[68] Tianhao Liu, Hanwen Kang, and Chao Chen. Orb-livox: A real-time dynamic system for fruit detection and localization. *Computers and Electronics in Agriculture*, 209:107834, 2023.

[69] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 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, 2016.

[70] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[71] Jun Lu, Nong Sang, Yang Hu, and Huini Fu. Detecting citrus fruits with highlight on tree based on fusion of multi-map. *Optik*, 125(8):1903–1907, 2014.

[72] Renfu Lu, Nathan Dickinson, Kyle Lammers, Kaixiang Zhang, Pengyu Chu, and Zhaojian Li. Design and evaluation of end effectors for a vacuum-based robotic apple harvester. *Journal of the ASABE*, 65(5):963–974, 2022.

[73] Renfu Lu, Zhao Zhang, and Anand K Pothula. Innovative technology for apple harvest and in-field sorting. *Fruit Quarterly*, 25(2), 2017.

[74] Yuzhen Lu, Dong Chen, Ebenezer Olaniyi, and Yanbo Huang. Generative adversarial networks (gans) for image augmentation in agriculture: A systematic review. *Computers and Electronics in Agriculture*, 200:107208, 2022.

[75] Vittorio Mazzia, Aleem Khaliq, Francesco Salvetti, and Marcello Chiaberge. Real-time apple detection system using embedded systems with hardware accelerators: An edge ai application. *IEEE Access*, 8:9102–9114, 2020.

[76] Siddhartha S Mehta, Chau Ton, S Asundi, and Thomas F Burks. Multiple camera fruit localization using a particle filter. *Computers and Electronics in Agriculture*, 142:139–154, 2017.

[77] SS Mehta and TF Burks. Multi-camera fruit localization in robotic harvesting. *IFAC-PapersOnLine*, 49(16):90–95, 2016.

[78] Mohamed Lamine Mekhalfi, Carlo Nicolò, Yakoub Bazi, Mohamad Mahmoud Al Rahhal, Norah A Alsharif, and Eslam Al Maghayreh. Contrasting yolov5, transformer, and efficientdet detectors for crop circle detection in desert. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2021.

[79] Chiranjivi Neupane, Anand Koirala, Zhenglin Wang, and Kerry Brian Walsh. Evaluation of depth cameras for use in fruit localization and sizing: Finding a successor to kinect v2. *Agronomy*, 11(9):1780, 2021.

[80] Chiranjivi Neupane, Anand Koirala, Zhenglin Wang, and Kerry Brian Walsh. Evalu-

ation of depth cameras for use in fruit localization and sizing: Finding a successor to kinect v2. *Agronomy*, 11(9):1780, 2021.

[81] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 8837–8845, 2019.

[82] Jason K Norsworthy, Sarah M Ward, David R Shaw, Rick S Llewellyn, Robert L Nichols, Theodore M Webster, Kevin W Bradley, George Frisvold, Stephen B Powles, Nilda R Burgos, et al. Reducing the risks of herbicide resistance: best management practices and recommendations. *Weed science*, 60(SP1):31–62, 2012.

[83] E-C Oerke. Crop losses to pests. *The Journal of Agricultural Science*, 144(1):31–43, 2006.

[84] Piyush Pandey, Hemanth Narayan Dakshinamurthy, and Sierra N Young. Autonomy in detection, actuation, and planning for robotic weeding systems. *Transactions of the ASABE*, page 0, 2021.

[85] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European conference on computer vision (ECCV)*, pages 269–286, 2018.

[86] Hetal N Patel, RK Jain, Manjunath V Joshi, et al. Fruit detection using improved multiple features based algorithm. *International journal of computer applications*, 13(2):1–5, 2011.

[87] Diego Inácio Patrício and Rafael Rieder. Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review. *Computers and electronics in agriculture*, 153:69–81, 2018.

[88] Tobias Pohlen, Alexander Hermans, Markus Mathias, and Bastian Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4151–4160, 2017.

[89] Feng Qingchun, Zheng Wengang, Qiu Quan, Jiang Kai, and Guo Rui. Study on strawberry robotic harvesting system. In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, volume 1, pages 320–324. IEEE, 2012.

[90] W Qiu and SA Shearer. Maturity assessment of broccoli using the discrete fourier transform. *Transactions of the ASAE*, 35(6):2057–2062, 1992.

[91] Maryam Rahnemoonfar and Clay Sheppard. Deep count: fruit counting based on deep simulated learning. *Sensors*, 17(4):905, 2017.

[92] Thinal Raj, Fazida Hanim Hashim, Aqilah Baseri Huddin, Mohd Faisal Ibrahim, and Aini Hussain. A survey on lidar scanning mechanisms. *Electronics*, 9(5):741, 2020.

[93] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[94] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[95] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019.

[96] Samuel Rota Bulo, Gerhard Neuhold, and Peter Kontschieder. Loss max-pooling for semantic image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2126–2135, 2017.

[97] Inkyu Sa, Zongyuan Ge, Feras Dayoub, Ben Upcroft, Tristan Perez, and Chris McCool. Deepfruits: A fruit detection system using deep neural networks. *Sensors*, 16(8):1222, 2016.

[98] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(60):1–48, 2019.

[99] Yongsheng Si, Gang Liu, and Juan Feng. Location of apples in trees using stereoscopic vision. *Computers and Electronics in Agriculture*, 112:68–74, 2015.

[100] Abhisesh Silwal, Joseph R Davidson, Manoj Karkee, Changki Mo, Qin Zhang, and Karen Lewis. Design, integration, and field evaluation of a robotic apple harvester. *J. Field Robot.*, 34(6):1140–1159, 2017.

[101] Abhisesh Silwal, Joseph R Davidson, Manoj Karkee, Changki Mo, Qin Zhang, and Karen Lewis. Design, integration, and field evaluation of a robotic apple harvester. *Journal of Field Robotics*, 34(6):1140–1159, 2017.

[102] Peter W Sites and Michael J Delwiche. Computer vision to locate fruit on a tree. *Transactions of the ASAE*, 31(1):257–0265, 1988.

[103] David C Slaughter and Roy C Harrell. Color vision in robotic fruit harvesting. *Transactions of the ASAE*, 30(4):1144–1148, 1987.

[104] Denis Stajnko, Miran Lakota, and Marko Hočevar. Estimation of number and diameter of apple fruits in an orchard during the growing season by thermal imaging. *Computers and Electronics in Agriculture*, 42(1):31–42, 2004.

[105] Daobilige Su, He Kong, Yongliang Qiao, and Salah Sukkarieh. Data augmentation for deep learning based semantic segmentation and crop-weed classification in agricultural robotics. *Computers and Electronics in Agriculture*, 190:106418, 2021.

[106] Jun Sun, Bing Lu, HanPing Mao, et al. Fruits recognition in complex background using binocular stereovision. *Journal of Jiangsu University-Natural Science Edition*, 32(4):423–427, 2011.

[107] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[108] Kanae Tanigaki, Tateshi Fujiura, Akira Akase, and Junichi Imagawa. Cherry-harvesting robot. *Computers and electronics in agriculture*, 63(1):65–72, 2008.

[109] Yunong Tian, Guodong Yang, Zhe Wang, Hao Wang, En Li, and Zize Liang. Apple detection during different growth stages in orchards using the improved yolo-v3 model. *Computers and electronics in agriculture*, 157:417–426, 2019.

[110] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.

[111] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. *Advances in neural information processing systems*, 27, 2014.

[112] T Trtík, R Chylík, J Fládr, J Štoller, and I Broukalová. Methods of lighting of concrete structures for high-speed camera measurement. In *IOP Conference Series: Materials Science and Engineering*, volume 596, page 012041. IOP Publishing, 2019.

[113] Nikos Tsoulias, Dimitrios S Paraforos, George Xanthopoulos, and Manuela Zude-Sasse. Apple shape detection based on geometric and radiometric features using a lidar laser scanner. *Remote Sensing*, 12(15):2481, 2020.

[114] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[115] Juan P Wachs, H I Stern, T Burks, and V Alchanatis. Low and high-level visual feature-based apple detection from multi-modal images. *Precision Agriculture*, 11:717–

735, 2010.

[116] Shaohua Wan and Sotirios Goudos. Faster r-cnn for multi-class fruit detection using a robotic vision system. *Computer Networks*, 168:107036, 2020.

[117] Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2606–2615, 2017.

[118] Dale Whittaker, GE Miles, OR Mitchell, and LD Gaultney. Fruit location in a partially occluded image. *Transactions of the ASAE*, 30(3):591–0596, 1987.

[119] Henry Williams, Canaan Ting, Mahla Nejati, Mark Hedley Jones, Nicky Penhall, JongYoon Lim, Matthew Seabright, Jamie Bell, Ho Seok Ahn, Alistair Scarfe, et al. Improvements to and large-scale evaluation of a robotic kiwifruit harvester. *J. Field Robot.*, 37(2):187–201, 2020.

[120] Qiufeng Wu, Yiping Chen, and Jun Meng. Dcgan-based data augmentation for tomato leaf disease identification. *IEEE Access*, 8:98716–98728, 2020.

[121] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Bridging category-level and instance-level semantic image segmentation. *arXiv preprint arXiv:1605.06885*, 2016.

[122] Rong Xiang, Huanyu Jiang, and Yibin Ying. Recognition of clustered tomatoes based on binocular stereo vision. *Computers and Electronics in Agriculture*, 106:75–90, 2014.

[123] Ya Xiong, Yuanyue Ge, Lars Grimstad, and Pål J. From. An autonomous strawberry-harvesting robot: Design, development, integration, and field evaluation. *J. Field Robot.*, 37(2):202–224, 2020.

[124] Guantao Xuan, Chong Gao, Yuanyuan Shao, Meng Zhang, Yongxian Wang, Jingrun Zhong, Qingguo Li, and Hongxing Peng. Apple detection in natural environment using deep learning algorithms. *IEEE Access*, 8:216772–216780, 2020.

[125] Bin Yan, Pan Fan, Xiaoyan Lei, Zhijie Liu, and Fuzeng Yang. A real-time apple targets detection method for picking robot based on improved yolov5. *Remote Sensing*, 13(9):1619, 2021.

[126] Tien-Ju Yang, Maxwell D Collins, Yukun Zhu, Jyh-Jing Hwang, Ting Liu, Xiao Zhang, Vivienne Sze, George Papandreou, and Liang-Chieh Chen. Deeperlab: Single-shot image parser. *arXiv preprint arXiv:1902.05093*, 2019.

[127] Stephen L Young, George E Meyer, and Wayne E Woldt. Future directions for automated weed management in precision agriculture. In *Automation: The future of weed control in cropping systems*, pages 249–259. Springer, 2014.

[128] Tao Yu, Chunhua Hu, Yuning Xie, Jizhan Liu, and Pingping Li. Mature pomegranate fruit detection and location combining improved f-pointnet with 3d point cloud clustering in orchard. *Computers and Electronics in Agriculture*, 200:107233, 2022.

[129] Baohua Zhang, Wenqian Huang, Chaopeng Wang, Liang Gong, Chunjiang Zhao, Chengliang Liu, and Danfeng Huang. Computer vision recognition of stem and calyx in apples using near-infrared linear-array structured light and 3d reconstruction. *Biosystems Engineering*, 139:25–34, 2015.

[130] Kaixiang Zhang, Pengyu Chu, Kyle Lammers, Zhaojian Li, and Renfu Lu. Active laser-camera scanning for high-precision fruit localization in robotic harvesting: System design and calibration. *arXiv preprint arXiv:2311.02500*, 2023.

[131] Kaixiang Zhang, Kyle Lammers, Pengyu Chu, Nathan Dickinson, Zhaojian Li, and Renfu Lu. Algorithm design and integration for a robotic apple harvesting system. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 9217–9224, 2022. https://doi.org/10.1109/IROS47612.2022.9981417.

[132] Kaixiang Zhang, Kyle Lammers, Pengyu Chu, Zhaojian Li, and Renfu Lu. System design and control of an apple harvesting robot. *Mechatronics*, 79:102644, 2021.

[133] Kaixiang Zhang, Kyle Lammers, Pengyu Chu, Zhaojian Li, and Renfu Lu. System design and control of an apple harvesting robot. *Mechatronics*, 79:102644, 2021.

[134] Kaixiang Zhang, Kyle Lammers, Pengyu Chu, Zhaojian Li, and Renfu Lu. An automated apple harvesting robot – from system design to field evaluation. *Journal of Field Robotics*, in press, 2023.

[135] Xin Zhang, Long He, Manoj Karkee, Matthew David Whiting, and Qin Zhang. Field evaluation of targeted shake-and-catch harvesting technologies for fresh market apple. *Trans. ASABE*, 63(6):1759–1771, 2020.

[136] Zhao Zhang, Yuzhen Lu, and Renfu Lu. Development and evaluation of an apple infield grading and sorting system. *Postharvest Biol. Technol.*, 180:111588, 2021.

[137] Jun Zhao, Joel Tow, and Jayantha Katupitiya. On-tree fruit recognition using texture properties and color data. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 263–268. IEEE, 2005.

[138] Yuanshen Zhao, Liang Gong, Yixiang Huang, and Chengliang Liu. A review of key techniques of vision-based control for harvesting robot. *Computers and Electronics in Agriculture*, 127:311–323, 2016.