Pontifícia Universidade Católica
do Rio de Janeiro

Gabriel Lins Tenório

Strawberry Monitoring: Detection, Classification, and Visual Servoing

Tese de Doutorado

Thesis presented to the Programa de Pós–graduação em Engenharia Elétrica of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Engenharia Elétrica.

Advisor: Prof. Wouter Caarls

Rio de Janeiro
April 2024

Pontifícia Universidade Católica
do Rio de Janeiro

Gabriel Lins Tenório

Strawberry Monitoring: Detection, Classification, and Visual Servoing

Thesis presented to the Programa de Pós–graduação em Engenharia Elétrica of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Engenharia Elétrica. Approved by the Examination Committee.

Prof. Wouter Caarls
Advisor
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Eduardo Costa da Silva
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Raul Queiroz Feitosa
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Antonio Candea Leite
NMBU

Prof. Weria Khaksar
NMBU

Rio de Janeiro, April 25th, 2024

Gabriel Lins Tenório

Master in Electrical Engineering and Majored in Control and Automation Engineering at the Pontifical University Catholic of Rio de Janeiro.

Bibliographic data

Lins Tenório, Gabriel

Strawberry Monitoring: Detection, Classification, and Visual Servoing / Gabriel Lins Tenório; advisor: Wouter Caarls. – Rio de janeiro: PUC-Rio , Departamento de Engenharia Elétrica, 2024.

v., 105 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica.

Inclui bibliografia

1. Engenharia Elétrica – Teses. 2. Servo Visão Baseado em Aprendizado Profundo;. 3. Segmentação de Instâncias em 3D;. 4. Detecção de Morangos;. 5. Agricultura de Precisão;. I. Caarls, Wouter. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. III. Título.

CDD: 621.3

In memoriam of my parents for their support, love, and encouragement.

Acknowledgments

I would like to thank my advisor, Prof. Wouter Caarls, for his wise and patient guidance and his enthusiasm in this work. Additionally, I extend my gratitude to Prof. Weria Khaksar for motivating me, lending an ear, and sharing ideas throughout this work.

I thank the National Council for Scientific and Technological Development (CNPq), the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), and the Norwegian University of Life Sciences (NMBU) for their financial support. This thesis was partially supported by the Scientific and Technological Development (CNPq) under a Doctoral Scholarship and project number 314121/2021-8. This work was partially funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This research was partially financed by the Research Council of Norway as a part of the DigiFoods SFI, under grant agreement 309259.

I would also like to extend my sincere thanks to Antonio Candeia Leite for his friendship, for recognizing my skills, and for opening doors to opportunities in Norway since 2017, as well as to the friends I made in Norway. A special thanks to Igor Costa, Lucas Vares, and Michael Angelo for their brainstorming help and for being a shoulder to lean on.

I want to express my gratitude to my fiancée, Lorraine Vargas, her family (Rayanne Vargas, Amilton Vargas, and Venância Vargas), my aunt Renilde Tenório and my friend Thiago Torres for their affection, belief in, and support for my ability to overcome life's challenges.

I wish to honor the memory of my beloved parents for their support, love, and encouragement. I know they are proud of me, as they always have been.

Abstract

Lins Tenório, Gabriel; Caarls, Wouter (Advisor). **Strawberry Monitoring: Detection, Classification, and Visual Servoing**. Rio de Janeiro, 2024. 105p. Tese de doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

The present work begins with an investigation into the use of 3D Deep Learning models for enhanced strawberry detection in polytunnels. We focus on two main tasks: firstly, fruit detection, comparing the standard MaskRCNN with an adapted version that integrates depth information (MaskRCNN-D). Both models are capable of classifying strawberries based on their maturity (ripe, unripe) and health status (affected by disease or fungus). Secondly, we focus on identifying the widest region of strawberries, fulfilling a requirement for a spectrometer system capable of measuring their sugar content. In this task, we compare a contour-based algorithm with an enhanced version of the VGG-16 model. Our findings demonstrate that integrating depth data into the MaskRCNN-D results in up to a 13.7% improvement in mAP across various strawberry test sets, including simulated ones, emphasizing the model's effectiveness in both real-world and simulated agricultural scenarios. Furthermore, our end-to-end pipeline approach, which combines the fruit detection (MaskRCNN-D) and widest region identification models (enhanced VGG-16), shows a remarkably low localization error, achieving down to 11.3 pixels of RMSE in a 224×224 strawberry cropped image. Finally, we explore the challenge of enhancing the quality of the data readings from the spectrometer through automatic sensor positioning. To this end, we designed and trained a Deep Learning model with simulated data, capable of predicting the sensor accuracy based on a given image of the strawberry and the subsequent displacement of the sensor's position. Using this model, we calculate the gradient of the accuracy output with respect to the displacement input. This results in a vector indicating the direction and magnitude with which the sensor should be moved to improve the sensor signal accuracy. A Visual Servoing solution based on this vector provided a significant increase in the average sensor accuracy and improvement in consistency across new simulated iterations.

Keywords

Deep Learning-Based Visual Servoing; 3D Instance Segmentation; Strawberry Detection; Precision Agriculture;

Resumo

Lins Tenório, Gabriel; Caarls, Wouter. **Monitoramento de Morangos: Detecção, Classificação e Servovisão**. Rio de Janeiro, 2024. 105p. Tese de Doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

O presente trabalho inicia com uma investigação sobre o uso de modelos de Aprendizado Profundo 3D para a detecção aprimorada de morangos em túneis de cultivo. Focou-se em duas tarefas principais: primeiramente, a detecção de frutas, comparando o modelo original MaskRCNN com uma versão adaptada que integra informações de profundidade (MaskRCNN-D). Ambos os modelos são capazes de classificar morangos baseados em sua maturidade (maduro, não maduro) e estado de saúde (afetados por doença ou fungo). Em segundo lugar, focou-se em identificar a região mais ampla dos morangos, cumprindo um requisito para um sistema de espectrômetro capaz de medir o conteúdo de açúcar das frutas. Nesta tarefa, comparou-se um algoritmo baseado em contorno com uma versão aprimorada do modelo VGG-16. Os resultados demonstram que a integração de dados de profundidade no MaskRCNN-D resulta em até 13.7% de melhoria no mAP através de diversos conjuntos de teste de morangos, incluindo os simulados, enfatizando a eficácia do modelo em cenários agrícolas reais e simulados. Além disso, nossa abordagem de solução ponta-a-ponta, que combina a detecção de frutas (MaskRCNN-D) e os modelos de identificação da região mais ampla (VGG-16 aprimorado), mostra um erro de localização notavelmente baixo, alcançando até 11.3 pixels de RMSE em uma imagem de morango cortada de 224×224. Finalmente, explorou-se o desafio de aprimorar a qualidade das leituras de dados do espectrômetro através do posicionamento automático do sensor. Para tal, projetou-se e treinou-se um modelo de Aprendizado Profundo com dados simulados, capaz de prever a acurácia do sensor com base em uma imagem dada de um morango e o deslocamento desejado da posição do sensor. Usando este modelo, calcula-se o gradiente da saída de acurácia em relação à entrada de deslocamento. Isso resulta em um vetor indicando a direção e magnitude com que o sensor deve ser movido para melhorar a acurácia do sinal do sensor. Propôs-se então uma solução de Servo Visão baseada neste vetor, obtendo um aumento significativo na acurácia média do sensor e melhoria na consistência em novas iterações simuladas.

Palavras-chave

Servo Visão Baseado em Aprendizado Profundo; Segmentação de Instâncias em 3D; Detecção de Morangos; Agricultura de Precisão;

Table of contents

List of figures

Figure 6.15 Fourth set of examples of the NGVS Performance in real-world situations. The red arrow on each case represents the orientation and magnitude in which the controller should follow in order to improve the sensor accuracy. 95

List of tables

List of Abreviations

ANN – Artificial Neural Networks
AP – Average Precision
CNNs – Convolutional Neuronal Networks
DL – Deep Learning
DoF – Degrees of Freedom
ES – Early Stopping
FoV – Field of View
GBS – Grid-Based Solution
IBVS – Image-Based Visual Servoing
IoU – Intersection over Union
LRF – Laser Rangefinder
mAP – Mean Average Precision
MLP – Multilayer perceptron
MSE – Mean Squared Error
NGVS – Network Gradient Visual Servoing
NO – Norway
PBVS – Position-Based Visual Servoing
RAS – Robotic and Automous Systems
RCNN – Region-based Convolutional Neural Network
RMSE – Root Mean Squared Error
ROI – Region Of Interest
UGV – Unmanned Ground Vehicle
UK – United Kingdom
VGG – Visual Geometry Group
VS – Visual Servoing
WSCNNs – Weight Standardized Convolutional Neural Networks

List of Symbols

ΔD	deviation (cm) of the sensor from the required distance to a strawberry
X, Y	pixel coordinates
p	camera position (m) relative to the robotic arm's base, $p = [x, y, z]$
x	horizontal position (m)
y	depth position (m)
z	vertical position (m)
$\bar{p}$	camera position (m) relative to the current strawberry's widest region, $\bar{p} = [\bar{x}, \bar{z}]$
$\bar{x}$	horizontal position (m)
$\bar{z}$	vertical position (m)
d	displacement (m) vector between two camera positions, $d = [\Delta x, \Delta z]$
Δx	horizontal displacement (m)
Δz	vertical displacement (m)
Acc_i	sensor's accuracy corresponding to an image I_i
R_{LRF}	scalar reading (cm) from a 1D LRF
μ_{LRF}	average (cm) of the readings from a 2D LRF
σ_{LRF}	standard deviation (cm) of the readings from a 2D LRF

1
Introduction

The importance of inspecting strawberries for signs of adequate ripening, nutrient absorption, and absence of diseases is vital for ensuring their overall quality. Quality maintenance also reduces waste due to spoilage and strengthens brand integrity, which is crucial in the competitive strawberry market. As research continues to evolve, the direct assessment of strawberry quality in outdoor as well as indoor fields has become a significant area of research [1, 2]. However, given the scarcity of labor, as is the case in Europe, and the time-consuming nature of manually operating sophisticated equipment, the adoption of autonomous systems stands out. Autonomous inspection systems can offer valuable agricultural information to farmers [3] and drastically reduce the manual labor involved in monitoring strawberries. An example of such a system is mobile manipulators, which commonly use an Unmanned Ground Vehicle (UGV) equipped with one or more manipulators. By integrating manipulators with UGVs, mobile manipulators provide enhanced versatility and adaptability, allowing for the performance of a variety of complex tasks. This integration enables addressing key agricultural challenges more effectively than UGVs alone, which are typically limited to basic mobility functions. These manipulators, integral components of mobile manipulators, are outfitted with sensors, cameras, or grippers, enabling them to perform tasks ranging from monitoring to the harvesting of strawberries. This application is exemplified in the work of Ge et al. [4], which employs mobile manipulators for harvesting strawberries.

In terms of quality perception, the literature contains numerous studies that employ simple cameras integrated with advanced Deep Learning algorithms for agricultural tasks. These include estimating the maturation of tomato clusters [5], detecting diseases and pests in strawberries [2], and assessing the quality of various fruits [6], all of which have shown promising results. On the other hand, there are advancements and innovations directly in sensor technology, as exemplified by a novel NIR Interaction Spectroscopy prototype, which is capable of estimating the dry matter content in potatoes without physical contact [7]. In a later development, Wold et al. [8] redesigned and recalibrated the spectrometer system for use in measuring the sugar con-

tent in strawberries.

1.1 Specific Challenges

Beyond quality sensing, a significant challenge in precision agriculture lies in the accurate autonomous location of fruits within complex agricultural environments. This has led to innovative efforts in developing technologies capable of automatically locating fruits in the field. One example is the work by Lins Tenorio et al. [5], which developed a system for automatic detection, tracking, and counting of tomato clusters using object detection techniques in continuous scenes of plant rows. Another example relates to strawberry detection for automated harvesting, as demonstrated by Ge et al. [4], who employed an instance segmentation algorithm to locate as well as classify ripe and unripe strawberries. Furthering this field, Le Louëdec and Cielniak [9] proposed a 3D semantic segmentation model to locate strawberries in polytunnels.

Regarding the use of spectrometers for autonomous inspection, specifically the one that measures sugar content on strawberries, there is a requirement for an experimentally ideal sensor position for accurate readings, preferably on the area of the strawberry with the largest horizontal surface [8]. Consequently, a system that can accurately position the sensor at this specific location through autonomous robotic arm positioning becomes crucial. Moreover, even with knowledge of the ideal experimental position, there are still variables under investigation, including occlusions by vegetation and technical uncertainties like specular occlusion and photon shot noise. Theoretically, these challenges can be addressed by implementing advancements that enable the robotic arm to precisely position the sensor in a way that minimizes the influence of these variables, thereby directly optimizing the signal quality from the spectrometer.

1.2 Proposed Solution

The present work initially introduces a vision system, based on Deep Learning algorithms, tailored for accurately positioning a NIR Spectrometer for the non-invasive sugar content estimation of strawberries in polytunnels. Our vision system is based on two primary components: fruit and widest region detectors. The fruit detector is responsible for locating strawberries in instances and classifying them into ripe, unripe, and affected by disease or fungus categories. Once the strawberries are identified by the fruit detector,

the vision system smoothly transitions to the widest region detector, aiming to pinpoint the widest part of the strawberry, which is a crucial requirement for precise quality assessment. Strawberry scenes in polytunnel environments are naturally complex, influenced by variations in lighting, occlusion, and the diversity of the strawberries. This complexity necessitates the application of an advanced technique such as Deep Learning, which offers sophisticated pattern recognition capabilities essential for adapting to the complexity of agricultural scenes. This approach significantly outperforms simpler computer vision methods. Our Deep Learning-based vision system undergoes training and validation using real-world data, ensuring it is well-prepared for practical applications. Subsequently, its performance is also validated in simulations to encompass a diverse range of scenarios. With the vision system evaluated, we conduct both simulated and real tests on a manipulator to assess the sensor's movement towards the desired positions. Figure 1.1 shows the mobile manipulator employed in this work and the detection of strawberries in a polytunnel environment using our Deep Learning-based vision system.

Figure 1.1: The left image shows the mobile manipulator employed in this work, comprising a UGV and a manipulator. The manipulator is equipped with a 3D camera for sensing and detecting strawberries, and a NIR spectrometer for the non-invasive sugar content estimation. The right image shows an example of strawberry detection in a polytunnel environment using the Deep Learning-based vision system.

In the subsequent phase, we explore our novel solution to optimize the signal quality obtained by the spectrometer through the precise positioning of the sensor in front of strawberries. Initially, we design a reward function capable of simulating the spectrometer signal accuracy, which penalizes, with continuous values, the sensor positioning. We gather data from each point of a set of strawberries through grid scanning, saving the camera's image and the sensor accuracy for each position on each strawberry. Secondly, we designed a Deep Learning Model and trained it using the dataset developed, which we have named the 'Accuracy Estimation Model'. The trained model is capable

of predicting the sensor accuracy that the sensor will return when moving the sensor to another position. Thirdly, we propose a novel approach based on the gradient of this model, capable of pointing to the direction that will increase the sensor measured accuracy using just an image as input. We apply this approach iteratively in a Visual Servoing control loop so that at the end of the iterations, the sensor accuracy for a given strawberry is optimized. Our proposed visual servoing method is a type of Extremum Seeking Control [10] for function maximization. However, instead of using visual servoing to a directly visible feature, it uses the gradient of the model to improve sensor positioning towards optimizing the predicted sensor accuracy. To the best of our knowledge, this is the first time such an approach has been implemented and documented in the literature. Additionally, we present a more simplified solution that utilizes predictions of sensor accuracy from the Accuracy Estimation Model for different displacements, thereby enabling the direct positioning of the sensor to the location of maximum sensor accuracy.

Next, this work explores sections on the Contributions and the Organization of this Thesis.

1.3 Contributions of this Thesis

The key contributions of this thesis are detailed below:

- Implemented a strawberry instance detector and developed a sensing approach for safe manipulation in autonomous strawberry picking, identifying safe regions for manipulation based on rectified instance segmentation masks [4];
- Demonstrated the relevance of employing Deep Learning models with 3D data inputs for enhanced performance in strawberry instance detection, additionally improving the identification of clustered strawberries [11];
- Implemented a strawberry detection pipeline designed to solve some of the positioning constraints of a Spectroscopic sensor able to provide sugar content data on strawberries [11];
- Assembled and enhanced a simulation environment to closely mirror real-world conditions by integrating an Unmanned Ground Vehicle (UGV), robot arm, and 3D camera, along with polytunnel and a strawberry generator. Custom enhancements included a halogen projector setup for the spectrometer system and sensing capabilities for simulating the Spectroscopic sensor accuracy, along with improvements to strawberry generation to accommodate various heights;

- Proposed a novel Visual Servoing methodology leveraging the gradient of a Deep Learning model to iteratively optimize the signal accuracy of the Spectroscopic sensor through autonomous sensor positioning.
- Designed a Deep Learning model trained to predict the accuracy of a spectrometer for a given image and desired displacement. This model is also tailored for use with the proposed gradient method.
- Developed a specific dataset for the designed model, meeting the requirements for sensor accuracy prediction and gradient computation for sensor accuracy optimization.

1.4 Organization of the Thesis

This Doctoral thesis is organized by first presenting the background in Chapter 2, which initially suggests some key readings, then proceeds to provide basic concepts and discusses the problems in precision agriculture on which this thesis is based. Following, Chapter 3 delves into the datasets employed in this thesis, and Chapter 4 discusses the models and methods used. Next, Chapter 5 provides details on our experiments, while Chapter 6 evaluates the results. Continuing, Chapter 7 wraps up the study with conclusions and Chapter 8 provides a foundation for future work. Lastly, Chapter 9 lists the publications made during the PhD journey.

2
Background

In order to better understand the concepts explained in this project, we recommend some Deep Learning books for beginners and intermediate readers in this area. The first recommendation is available online [12]. The second and third recommendations are, respectively, the classic [13] and the current [14] Deep Learning books. For the integration of computer vision with Robotics, especially in the context of visual control or visual servoing, we recommend a comprehensive source [15].

2.1
Deep Learning

The field of image analysis has advanced significantly in the past decade, moving away from traditional methods that largely relied on computer vision techniques. These traditional techniques often required handcrafted features and time-consuming manual adjustments tailored to specific datasets. For instance, Belhumeur et al. [16] demonstrated the use of eigenfaces in facial recognition, highlighting the necessity for carefully designed features in early computer vision. Similarly, Viola and Jones [17] developed a rapid object detection framework that became foundational due to its effective feature design. Lowe's SIFT algorithm [18] further exemplifies this by providing robust feature detection in varied image contexts. Lastly, Dalal and Triggs' work on histogram of oriented gradients [19] showcased how tailored feature extraction can significantly improve the accuracy of human detection algorithms. Later, with the advances in Graphics Processing Units (GPUs) [20], which significantly enhanced computational speed and efficiency, and the emergence of Deep Learning, particularly Convolutional Neural Networks (CNNs) [21, 22], the scenario has been essentially transformed.

CNNs are able to automate the feature extraction process, learning spatial structures directly from image pixels. The process involves the use of convolution, an algebraic operation that is applied in parallel across the image using multiple kernels. These kernels are essentially trainable filters that adapt during the learning process to become specialized in extracting different types of image features. The architecture of these networks typically comprises a

series of convolutional layers, designed to recognize patterns at varying levels of complexity, and pooling layers, which reduce data dimensions while retaining dominant features. The convolutional layers with their kernels, through the training process, enable the network to progress from recognizing generic patterns like edges to identifying more complex, dataset-specific attributes, also known as features.

In terms of application, CNNs are frequently used in supervised learning tasks, where images as inputs are matched with various types of labels, such as binary values, continuous values (which could be scalar or vector), bounding box coordinates, masks or a combination of these. During the training phase, these input/label pairs are used by Deep Learning models to effectively accomplish the specified tasks as Image Classification, Image Regression, Semantic and Instance Segmentations which will be explored in the following subsections. Additionally, we explore the theory of the gradient in a Neural Network, which is important for understanding the approach proposed in this work.

2.1.1 Image Classification and Regression

In Image Classification, the objective is to determine the probabilities that an image belongs to certain established categories, such as the classification of strawberries. Models such as VGGNet [23], AlexNet [22], ResNet [24], and EfficientNet [25] have demonstrated remarkable success in such classification tasks. On the other hand, regression tasks require modifications to the output layer of the network aiming for the model to predict and interpolate one or more continuous values associated with an image. An example of this is determining the coordinates of the widest regions of the strawberries.

2.1.2 Semantic and Instance Segmentations

Semantic segmentation is a technique that divides an image into regions that are semantically comparable, classifying each pixel of the image according to its respective class. For instance, in agricultural applications, this method can be used to categorize pixels related to different strawberry classes. Architectures such as FCN [26], U-Net [27] and SegNet [28] exemplify the implementation of this technique. Expanding upon this idea, a paper by Le Louëdec et al. [9] introduced a novel Semantic Segmentation architeture to achieve effective 3D segmentation of strawberries in both agricultural and simulated polytunnels using RGB-D data (combining color and depth information). However,

while effective, semantic segmentation alone may encounter limitations, particularly in the precise 3D localization of individual objects, as needed in tasks like automated harvesting.

To address these limitations, Instance Segmentation advances the concepts of Semantic Segmentation by not only classifying each pixel of an image but also distinguishing between different instances of the same class. For instance, in the classification of strawberries, it differentiates individual strawberries from one another, assigning a unique identifier to each one. While MaskRCNN [29] remains notable in this area for its segmentation quality, other models like YOLACT [30] offer efficient real-time instance segmentation and recent advancements in the YOLO family [31] finds the balance between segmentation quality and speed. Illustrating the practical application of these concepts, Ge et al. [4] successfully employed MaskR-CNN for precisely identifying and locating each strawberry as well as classifying their ripeness in polytunnels. This was essential for enabling their robotic system to efficiently and safely pick the ripe strawberries while avoiding unripe ones.

2.1.3 Instance Segmentation Performance Measurement

To evaluate the segmentation performance of an instance segmentation model for a given class, the Intersection over Union (IoU) metric is commonly used. This metric, which provides a value between 0 and 1, quantifies the segmentation overlap by dividing the area of intersection by the area of union between the model's predicted mask and the labeled ground truth mask. We illustrate this calculation visually in Figure 2.1.

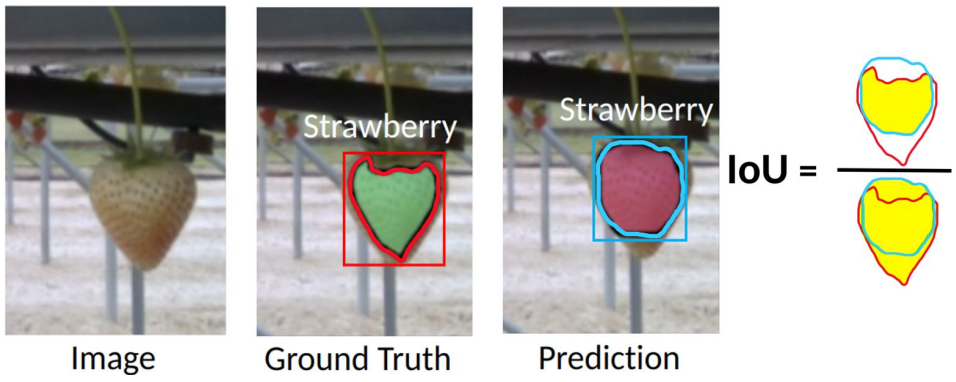

Figure 2.1: Illustration of IoU Calculation. The red contours encompass the labeled ground truth mask, while the light blue contours encompass the predicted mask from an instance segmentation model. The areas shaded in yellow represent the Area of Intersection (above) and the Area of Union (below) in the IoU calculation. The rectangles represent the bounding boxes, which encompass the masks in rectangular form.

An important confidence criterion defined before the calculation of the IoU is the confidence score, typically set between 0.70 and 0.95. The confidence score filters predictions by their reliability, ensuring that only those with high confidence are considered in the evaluation. To measure overall segmentation performance for a given class, the IoU is calculated for each instance in the test set for that class, and then the average of these values is taken.

Another metric that provides a more comprehensive assessment of model performance for a given class is the Average Precision (AP), which essentially combines some metrics in its calculation, providing a value between 0 and 1. One of the metrics included in AP is the 'IoU threshold,' which is the application of a threshold to the IoU to determine the True Positive detections (TPs) for IoU values above that threshold, and False Positive detections (FPs) for values below that threshold. The other two metrics included in the AP calculation are Precision and Recall. Both metrics utilize the TPs and FPs in their calculations, but Recall also incorporates the Ground Truths (GTs). For the calculation of Average Precision (AP), some additional steps are necessary, since Precision and Recall individually provide only a scalar value, and AP is derived from the precision-recall curve. Therefore, it is essential to establish multiple values (i.e., a range) for the 'IoU threshold,' as well as define a confidence score for the detections, in order to produce a series of values for these metrics. For example, by choosing a confidence score of 0.8 for the detections and employing an IoU threshold range from 0.5 to 0.75, we can calculate a series of Precision and Recall values and then plot the corresponding precision-recall curve as shown in Figure 2.2. The AP is then calculated as the area under this curve.

To calculate the overall AP for a given class, the AP is calculated for each instance in the test set for that class, and then the average of these values is taken.

2.1.4 Gradient Computation in Deep Learning

Understanding the fundamental concept of the gradient within a Neural Network is fundamental as it forms the basis of our proposed solution presented later in this work. To clarify this concept, we examine a Multi-Layer Perceptron (MLP) network, commonly referred to as a Dense Network. Consider an MLP consisting of two inputs (x_1 and x_2), two hidden layers, and a single output neuron, which together produce one output (y). The configuration of the network and terminology used are illustrated in Figure 2.3.

Considering the MLP structure, the computation for the forward pass

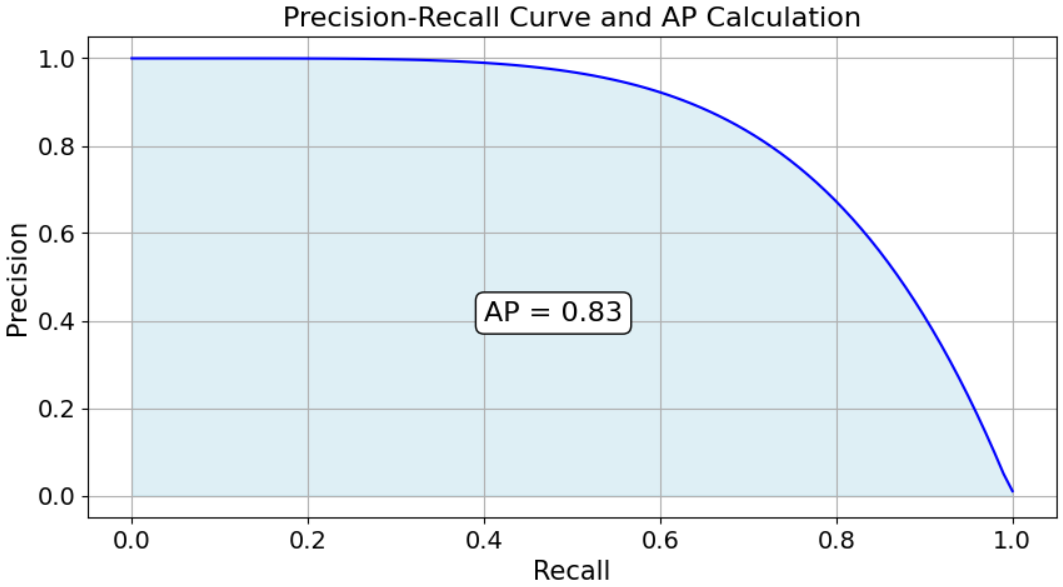

Figure 2.2: Precision-Recall Curve used for AP Calculation. The area under the curve represents the Average Precision (AP), quantifying the model's performance across various IoU threshold levels.

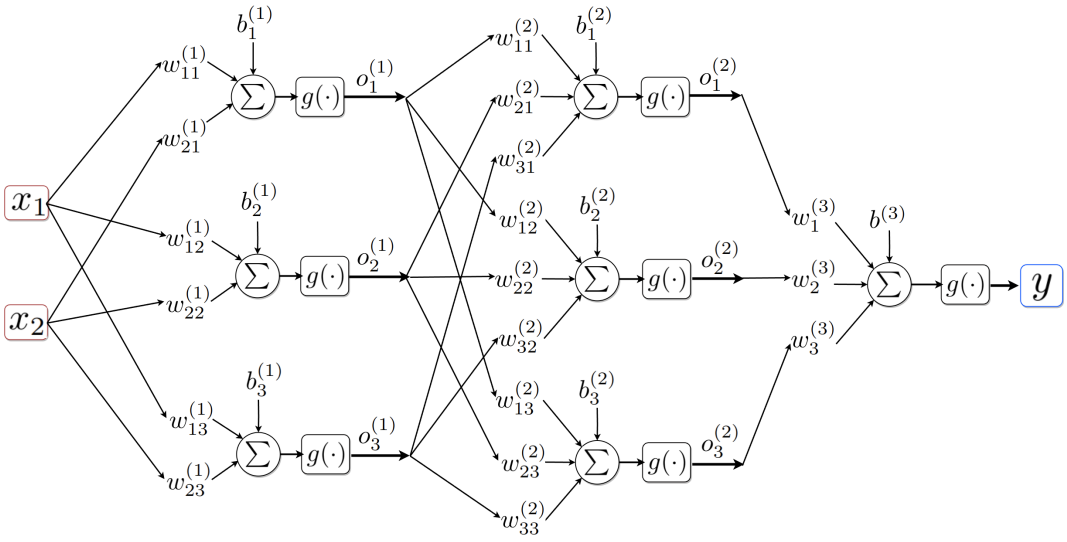

Figure 2.3: Schematic of a Multi-Layer Perceptron and Terminology Used.

begins at the first hidden layer. The net input (i.e. the value before the activation function $g(\cdot)$) to each neuron, denoted as $u_j^{(1)}$, is the weighted sum of the inputs plus the bias term for that neuron (Equation 2-1). The application of the activation function $g(\cdot)$ to these net inputs is presented by Equation 2-2, demonstrating how the inputs x_1 and x_2 are transformed through the first layer's weights and biases to produce the output $O_j^{(1)}$.

$$u_j^{(1)} = w_{1j}^{(1)} x_1 + w_{2j}^{(1)} x_2 + b_j^{(1)}, \tag{2-1}$$

$$O_j^{(1)} = g\left(u_j^{(1)}\right) \tag{2-2}$$

The process continues with the second hidden layer, where the net inputs $u_j^{(2)}$ are calculated as the weighted sum of the outputs from the first layer plus the biases (Equation 2-3), and the outputs $O_j^{(2)}$ are obtained by applying the activation function to $u_j^{(2)}$, as detailed by Equation 2-4.

$$u_j^{(2)} = \sum_{i=1}^{3} w_{ij}^{(2)} O_i^{(1)} + b_j^{(2)}\,, \tag{2-3}$$

$$O_j^{(2)} = g\left(u_j^{(2)}\right) \tag{2-4}$$

Finally, the determination of the network's output y is based on the outputs from the second hidden layer. The net input $u^{(3)}$ to the output neuron is the weighted sum of the second layer's outputs plus the bias (Equation 2-5), and the final output y is obtained by applying the activation function to $u^{(3)}$, as represented by Equation 2-6.

$$u^{(3)} = \sum_{i=1}^{3} w_i^{(3)} O_i^{(2)} + b^{(3)}\,, \tag{2-5}$$

$$y = g\left(u^{(3)}\right) \tag{2-6}$$

With the equations for each layer's outputs now established, we possess the necessary components to calculate the gradient of the network's output y with respect to an input x_k. This calculation, performed through the application of the Chain Rule, is detailed in Equation 2-7.

$$\frac{\partial y}{\partial x_k} = \sum_{i=1}^{3} \left(\frac{\partial y}{\partial O_i^{(2)}} \frac{\partial O_i^{(2)}}{\partial u_i^{(2)}} \left(\sum_{j=1}^{3} \frac{\partial u_i^{(2)}}{\partial O_j^{(1)}} \frac{\partial O_j^{(1)}}{\partial u_j^{(1)}} \frac{\partial u_j^{(1)}}{\partial x_k} \right) \right)\,, \tag{2-7}$$

where:

- $\frac{\partial y}{\partial O_i^{(2)}}$ is the weight $w_i^{(3)}$ from neuron i in the second hidden layer to the output neuron.
- $\frac{\partial O_i^{(2)}}{\partial u_i^{(2)}}$ is the derivative of the activation function $g(\cdot)$ at the second hidden layer.
- $\frac{\partial u_i^{(2)}}{\partial O_j^{(1)}}$ is the weight $w_{ji}^{(2)}$ from neuron j in the first hidden layer to neuron i in the second hidden layer.
- $\frac{\partial O_j^{(1)}}{\partial u_j^{(1)}}$ is the derivative of the activation function $g(\cdot)$ at the first hidden layer.
- $\frac{\partial u_j^{(1)}}{\partial x_k}$ is the weight $w_{kj}^{(1)}$ from the input x_k to neuron j in the first hidden layer.

In the theory of calculus, the gradient of a function at a specific point indicates the direction of the maximum rate of change of the function and also the rate of change in that direction. Analogously, in the context of a Neural

Network, the gradient $\frac{\partial y}{\partial x_k}$ not only demonstrates how a small change in a specific input x_k affects the output y, but also provides information about the direction of this change and the magnitude of the change. This work will investigate and validate the theoretical basis of this principle through implementation and experiments.

2.2 Robotic Arm Positioning and Visual Servoing

Robotic arm positioning relies on its Degrees of Freedom (DoF), joint angles, and link lengths, which determine the workspace of a robot to interact with an environment. An important part of the robotic arm is the end effector, where tools such as grippers and cameras can be applied to perform autonomous tasks. If the camera is attached to the end effector (as is the case in this work), the configuration is called "eye-in-hand", while if the camera is mounted at a fixed point, it is referred to as an "eye-to-hand" configuration.

Some crucial aspects to understand in robotic positioning are direct kinematics and inverse kinematics. Direct kinematics involves determining the end effector's current 3D position from known joint angles through kinematic equations. On the other hand, inverse kinematics calculates the required joint angles in order to move the end efector to desired 3D position. In this case, the calculation can be a bit more challenging depending on the DoF, but in practice it is computed with the assistance of libraries.

Visual Servoing (VS) introduces an additional layer of complexity and capability to robotic positioning, utilizing sensor data to guide the robot's movements. The main subdivisions of Visual Servoing are Image-Based Visual Servoing (IBVS) and Position-Based Visual Servoing (PBVS), each with distinct advantages and challenges. IBVS operates directly with image coordinates, requiring only the visual data and the detected object, which allows for more cost-effective implementations but can be prone to noise and complexity. In contrast, PBVS not only depends on detecting the object within the environment but also significantly relies on the quality of a 3D point cloud of the environment. In the field of agricultural robotics, advanced visual servoing techniques are being tailored to specific environmental and operational needs. Hani et al. [32] developed a visual servoing framework for orchards, employing a robust feature tracking algorithm that allows precise positioning under varying environmental conditions. This method primarily utilizes IBVS, focusing on adapting to dynamic scenarios like strong winds and partial occlusions. Conversely, Shi et al. [33] engineered a 'Global–Local' visual servoing system that combines the 'eye-to-hand' and 'eye-in-hand' configurations, optimizing

both the speed and accuracy required for effective tomato picking. A relevant application of VS to strawberry picking is discussed in the work by Gamba et al. [34], which combines IBVS and PBVS approaches. Additionally, the PBVS technique is explored in the work by Ge et al. [4], with a more detailed discussion to follow in the subsequent section.

2.3 Problem: Strawberry Picking

In this section, we explore the autonomous strawberry picking problem, where the strawberry detection method discussed later in this thesis (i.e., Instance Segmentation) directly relates. This discussion is part of a broader investigation that we have contributed to and have had the opportunity to publish in a journal [4].

The strawberry picking problem is challenging due to the complexity of real-world polytunnel environments. It involves not only detecting the strawberries, but also need to avoid obstacles for safe autonomous robotic manipulation and picking. The obstacles we address include straps used by farmers to support and manage plant growth, ensuring optimal sunlight as well as tables that support the plants and facilitate both drainage and ease of access for maintenance. Safe picking must avoid hitting and getting caught on the straps while also not colliding with the tables (maintaining a safe distance). Both situations can damage the plants, strawberries, and the manipulator. In Figure 2.4, we illustrate an example of safe regions for picking with robotic manipulation.

In order to achieve safe manipulation, we have developed a Machine Vision System that, when provided with a 3D image (i.e., RGB and Depth), is capable of identifying the 3D locations of strawberries within a safe manipulation region. The diagram of this vision system is illustrated in Figure 2.5. Besides the input to the vision system being 3D, the fruit detection model implemented (i.e., Mask R-CNN) solely utilizes the RGB data to determine the pixel locations of the strawberries. The exploration of using 3D data for fruit detection is further discussed later in this thesis.

Additionally, we proposed a refined approach, referred to as 'method 2,' which aims to improve the strap segmentation results through segmentation rectification, enhancing the safe region reliability. In order to achieve this, first, the Canny Edge Detection algorithm proposed by Canny [35] was applied to verify all of the edge points of a segmented strap. Thereafter, we sequentially applied the Probabilistic Hough Transform algorithm proposed by Kiryati et al. [36], which uses a random subset from the edge detector to obtain multiple

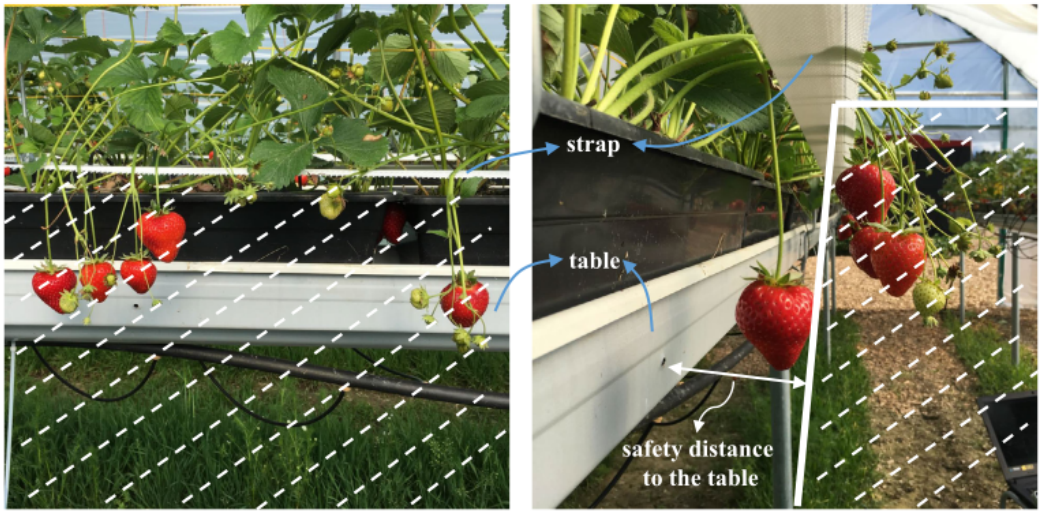

Figure 2.4: The safety manipulation region for the strawberry picking robot. (a) is a front view with the safety region marked by white dash line; (b) is a side view with the safety region marked by white dash line.

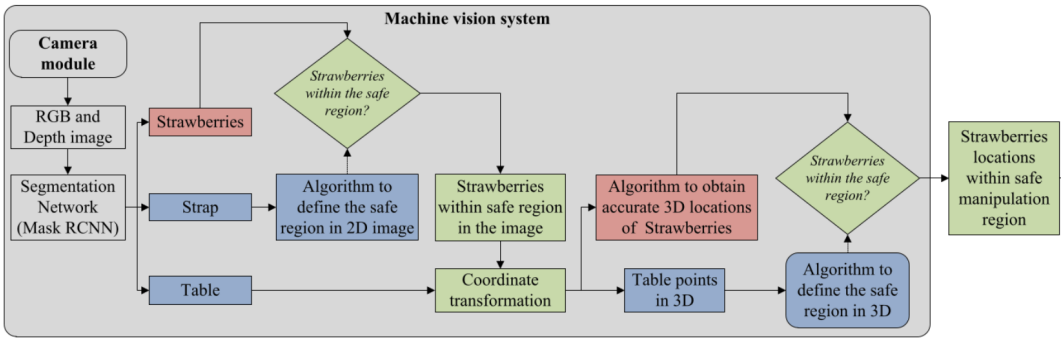

Figure 2.5: Machine Vision System architecture diagram.

lines in the image, including their starting and ending coordinates. All these coordinates were then used to calculate the line equation $y = mx + b$ that best interpolates all the points by using least squares. The bounding box that enclosed all the strap masks, marked by the dash line in Figure 2.6, was determined by the width of the strap and the fitted line. As shown in Figure 2.6, to verify whether strawberries are above or below the straps and assign a warning sign (dangerous or safe) to each fruit, x_i is applied to the line equation to obtain the y and compare it to the $y_{\text{top}} + \tau$. This τ is a value obtained through the original segmented mask to determine the safe manipulation region between the line and the position of the top of the fruit. As shown in Figure 2.6, all cases were defined correctly using the 'method 2'.

Some visual results of this system, showing the location of the ripe strawberries and indicating whether they are in safe regions for manipulation, can be observed in Figure 2.7.

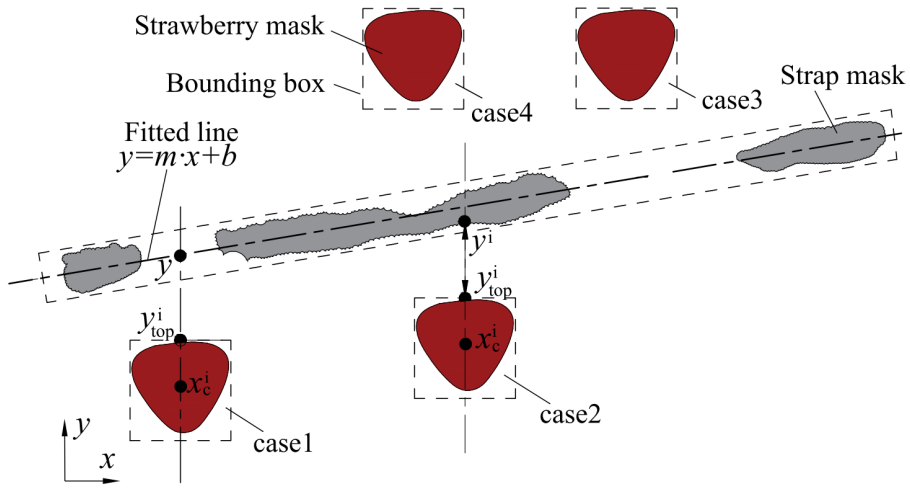

Figure 2.6: Safety assessment for strawberry handling related to strap positioning. In a ground truth scenario, Cases 1 and 2 would be considered safe for handling as they are located below the strap. Conversely, Cases 3 and 4 would be considered dangerous for handling, as they are located above the strap where the robotic arm could get caught during harvesting, potentially causing accidents. Using Method 1 (represented by the fragmented strap mask): Cases 1, 2, and 4 would be correctly classified with respect to the safety condition. However, Case 3 would be incorrectly classified due to the fragmentation of the strap mask in this instance. With the implementation of Method 2 (represented by a line passing through the middle of the fragmented mask): All cases would be correctly classified in relation to the safety condition.

In more detail, this publication can be found in the work of Ge et al. [4], in which we participated in the expansion of the dataset classes, in the implementation of the fruit detection model, in proposing a method for segmentation rectification, in the code to identify the safe regions for manipulation and in writing several chapters of the journal. Specifically, our contributions to the paper are found in the following sections: III.A, IV.A, IV.B, IV.C, IV.D, and in the Visual Results for Figures 3, 10, and 11, as well as in the experiments in Sections V.B and V.C.

2.4 Problem: Sugar Content Estimation on Strawberries

In this section, we explore the problem of estimating the sugar content in strawberries. The research conducted in this thesis is fundamental for developing a strategy for the positioning of a Spectroscopic sensor through autonomous robotic manipulation. We begin by presenting and discussing two conventional sensors, which can be observed in Figure 2.8.

The first sensor is the 'Handheld Strawberry Brix Meter' by Barry Century®, which operates by using physical samples of strawberries. The

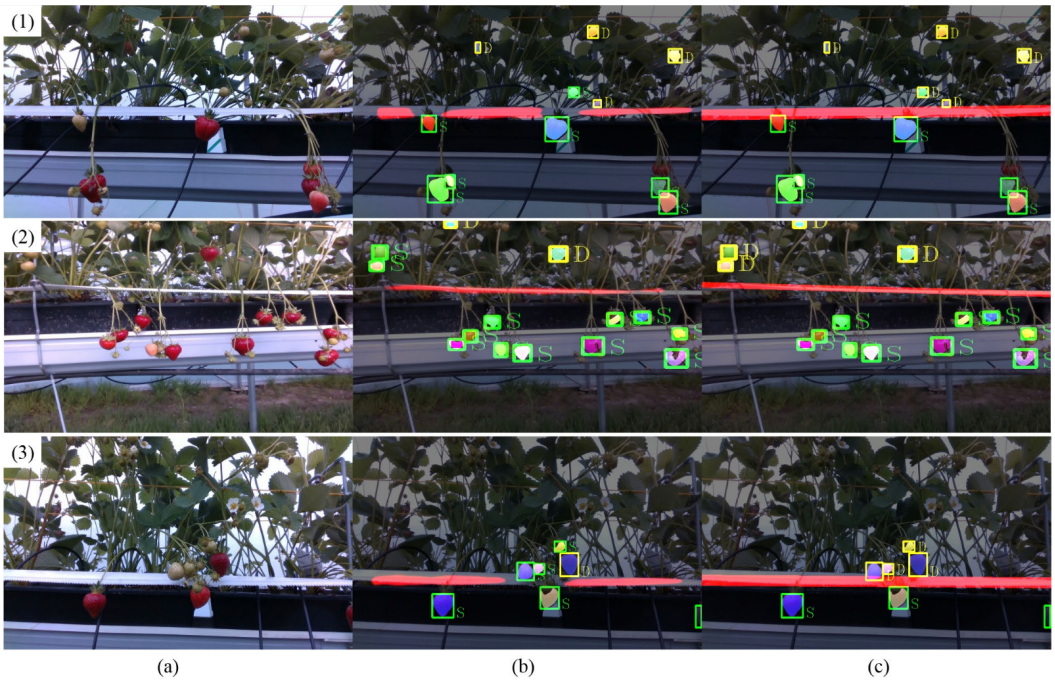

Figure 2.7: Visual results of the safety solution for the original strap segmentation and the rectified strap segmentation: (a) original images (1,2,3); (b) the image results of the instance segmentation method; (c) image results of the instance segmentation method with segmentation rectification ('method 2'); The green and yellow bounding boxes indicate, the safe (S) and the dangerous (D) warning signs.

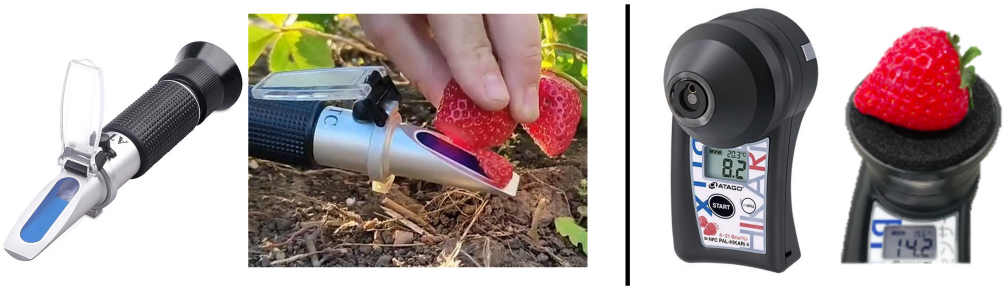

Figure 2.8: Two conventional sensors utilized for sugar content detection in strawberries: Barry Century® Sensor on the left and PAL-HIKARi® Sensor on the right.

fruit is squeezed onto the front part of the sensor, after which the sugar content level is read through an optical viewer on the rear side. This method tends to provide accurate results but necessitates the destruction of the fruit for measurement. The second sensor is the 'Strawberry Infrared Brix Refractometer' by PAL-HIKARi®, which functions by touching the sensor on the strawberry, then displaying the sugar content readings on its built-in screen. The main disadvantage of this sensor is that it requires multi-angle measurements for accurate averaging. Moreover, since it necessitates contact

with the strawberries, there is a risk of contamination and potential damage to the fruits. Due to the operational complexity of both sensors and the need for precise handling, their use for automatic data collection with robotic manipulators, for example, becomes unfeasible.

With advancements in spectroscopy sensors, the work of Wold et al. [8] introduced a state-of-the-art non-contact, non-destructive sensor capable of measuring the sugar content of strawberries from a distance of 20cm. Given these features, this sensor can easily be integrated with autonomous robots for automated data collection. A schematic diagram of this sensor's operation is shown in Figure 2.9.

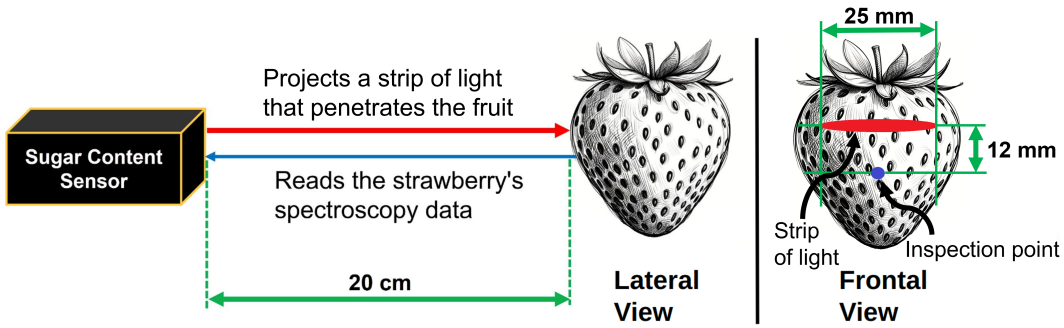

Figure 2.9: Schematic diagram of the operation of the Spectroscopic Sugar Content Sensor. The sugar content information is derived from the spectroscopy data.

The sugar content sensor as described by Wold et al. [8] operates by projecting a rectangular strip of light that penetrates the fruit and achieves accurate sensing with just a single measurement. Directly beneath the central area of this projection, it reads the spectroscopy data at a point referred to as the inspection point. Positioned 20 cm away from the fruit, the strip of light area on the fruit measures approximately 25 mm in width and 3 mm in height. The vertical distance from the center of the projection to the inspection point is about 12 mm. The inspection point itself has a diameter of about 2-3 mm at this same distance. The determination of sugar content, quantified as a scalar value, is derived from the spectroscopy data obtained through processing steps and regression models, focusing on critical wavelengths around 910 nm, 960 nm, and a shoulder at 984 nm, as highlighted by Wold et al [8]. Figure 2.10 illustrates an example of a strawberry spectrum captured using this sensor, with the key wavelengths indicated.

The inspection point is a delicate signal and susceptible to noise, thus there are some problem constraints that need to be considered in order to ensure a correct measurement, as observed from their controlled environment experiments. According to the sensor specialists, the main concerns for correct

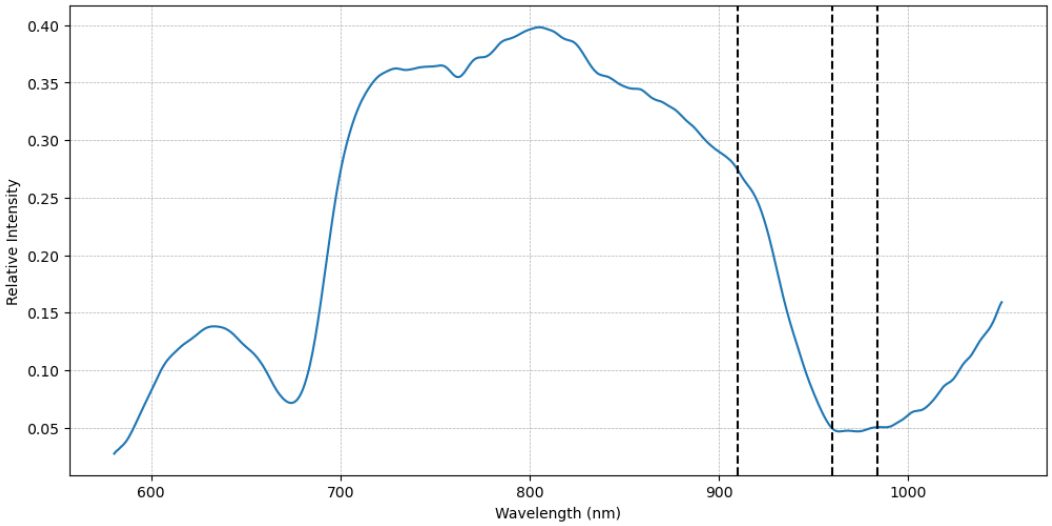

Figure 2.10: Example spectrum of a strawberry obtained with the Sugar Content Sensor. The critical wavelengths for sugar content analysis are marked for reference.

measurement require that the distance to the berry is within a specified distance offset (20 ± 1 cm), and that the interaction region does not "fall off" the berry, meaning both the strip of light and the inspection point must be fully on the surface of the strawberry at the specified distance. However, there are also other factors that can affect the sensor accuracy or quality of the signal, such as occlusions due to leaves, stems, and runners and other other uncertainties like specular occlusion and photon shot noise. The sensor accuracy values can be obtained through the processing of the wavelengths and by establishing a metric to provide a value between 0 and 1. Enhancing the sensor accuracy to achieve values close to 1 can be done by optimizing the sensor's positioning, which may also indirectly mitigate the impact of the uncertainties on the signal.

3
Datasets

In this chapter, we begin by introducing and analyzing the spectral data collected from strawberries, focusing on the influence of sensor positioning on the spectra (Section 3.1). Subsequently, we discuss the acquisition and labeling of the strawberry datasets (Section 3.2, Section 3.3 and 3.4) used in this work, with the goal of training models capable of detecting fruit instances and classifying strawberries, as well as identifying the widest region of strawberries. Finally, in Section 3.5, we present a summary of the datasets.

3.1
Spectral Data Collection and Analysis

Following the introduction of the Spectroscopic Sugar Content Sensor in Section 2.4, we proceed towards a more specific phase of the current research. After carrying out a series of real-world experiments, primarily aimed at internal testing within the spectroscopy group, we leveraged the opportunity to also explore the impact of sensor positioning on spectrometer readings. Utilizing a manually controlled mobile manipulator, we assisted the sensor group in obtaining spectral data from 10 different strawberries under varying time and light conditions within a single sunny/cloudy day. Figure 3.1 presents the day's temperature and light measurements, showing significant variations in light intensity due to transient cloud cover changes.

The data collected on this day of experiments consisted primarily of graphs plotting Relative Intensity against Wavelength from the hyperspectral sensor, as well as the Distance Offset (ΔD) and images of strawberries from a 3D camera. The variable ΔD represents the deviation of the sensor's distance from the strawberries, with a tolerance range of ± 1 cm, determined using a 3D camera. A positive deviation indicates that the sensor is positioned further away from the strawberry than the intended distance. Conversely, a negative deviation means that the sensor is positioned closer to the strawberry than the intended distance. With the data provided by the team, we normalized the spectra to make it more comparable between measurements and equalized the images to enhance the visibility of the strip of light. Below, we present examples of four unique strawberries (denoted as MB02, MB04, MB05 and MB06) to

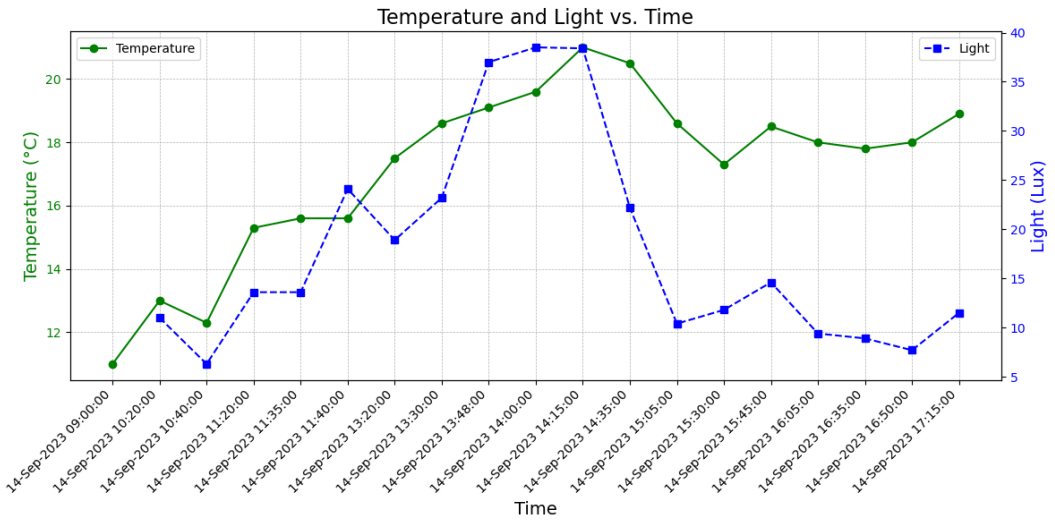

Figure 3.1: Temperature and Light Throughout Day of Experiments

better understand the problem of how the sensor positioning affected the quality of the spectroscopy data collected, specifically focusing on explaining the influence of well-understood variables.

3.1.1 MB02 Analysis

First, we examine the impact of sensor positioning on the MB02 data, focusing on the Normalized Relative Intensity and the Distance Offset from the sensor to the strawberry. Figures 3.2 and 3.3 present these aspects respectively.

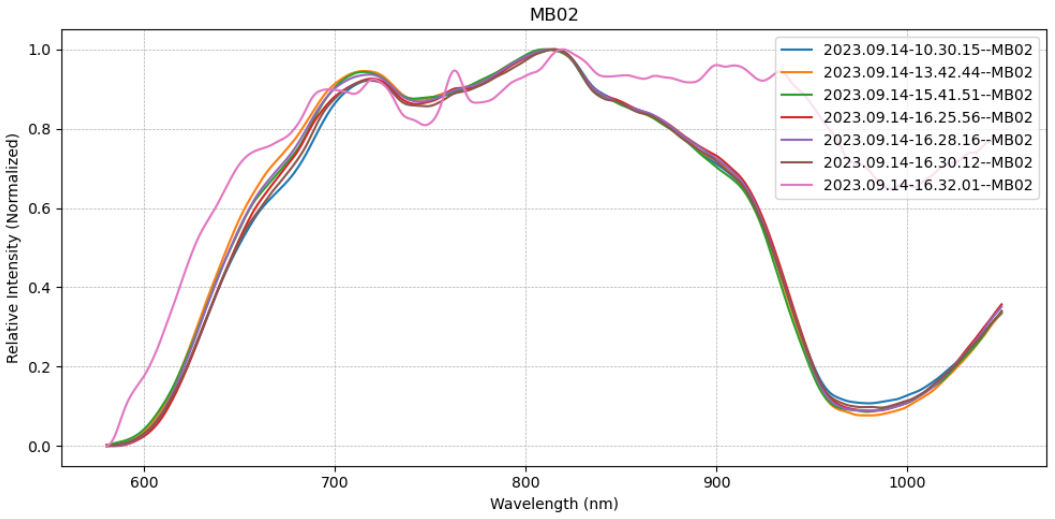

Figure 3.2: Spectral Data of Sample MB02, where the vertical axis represents the Normalized Relative Intensity and the horizontal axis represents the Wavelength in [nm].

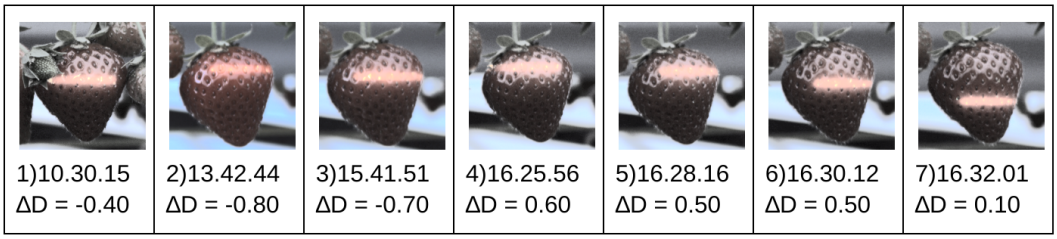

Figure 3.3: Image samples of a single strawberry captured at various times throughout the day, correlated with the spectral data of MB02.

- For samples 1) and 4): The strips of light are well-centered on the fruit, and ΔD are within the range, resulting in smooth wavelengths.
- For samples 2), 3) 5) and 6): Despite the strips of light being positioned more to the right, the wavelengths represented are also smooth.
- For sample 7): ΔD is within the range (the image's sharp focus confirms this), but the strip of light is shifted considerably towards the right and to the bottom of the target. This position of the projection has clearly affected the quality of the signal.

3.1.2 MB04 Analysis

Following our examination of MB02, we next turn our attention to the MB04 data to assess the impact of sensor positioning by observing the Normalized Relative Intensity and the Distance Offset from the sensor to the strawberry. Figures 3.4 and 3.5 illustrate these measurements.

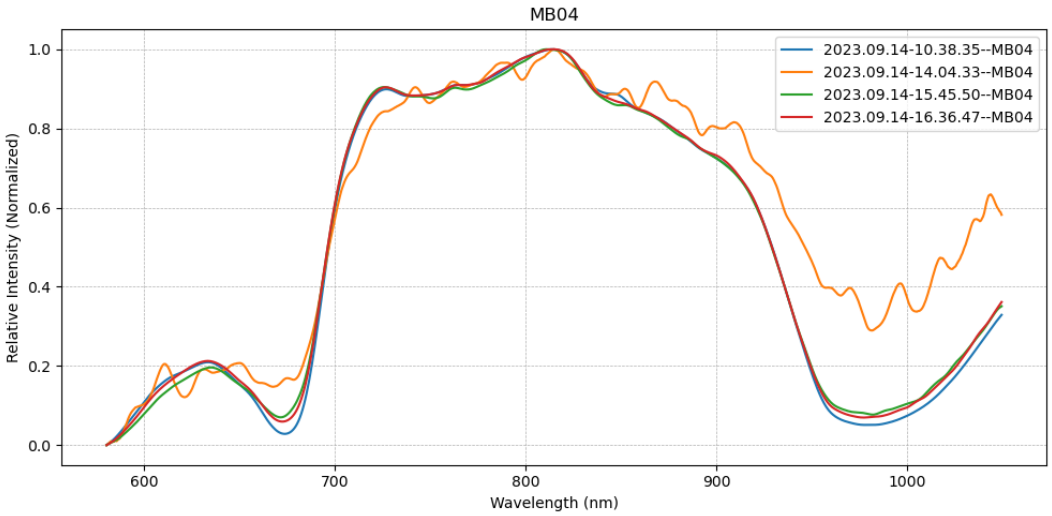

Figure 3.4: Spectral Data of Sample MB04, where the vertical axis represents the Normalized Relative Intensity and the horizontal axis represents the Wavelength in [nm].

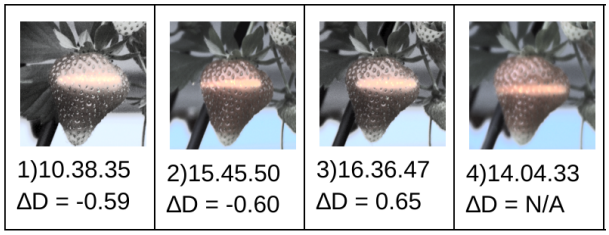

Figure 3.5: Image samples of a single strawberry captured at various times throughout the day, correlated with the spectral data of MB04.

– For samples 1) to 3): The strip of light is well-centered, and ΔD are within the acceptable range. In 2), the strip of light is slightly shifted to the left, but this shift does not impact the wavelength, which remains smooth.

– For sample 4): Given that the strawberry is clearly out of focus, the distance exceeds the tolerance, and this condition affected the reading of the spectrometer.

3.1.3 MB05 Analysis

Continuing our investigation, we analyze the MB05 data for insights into the effects of sensor positioning. We present both the Normalized Relative Intensity and the Distance Offset from the sensor to the strawberry. Refer to Figures 3.6 and 3.7 for these aspects.

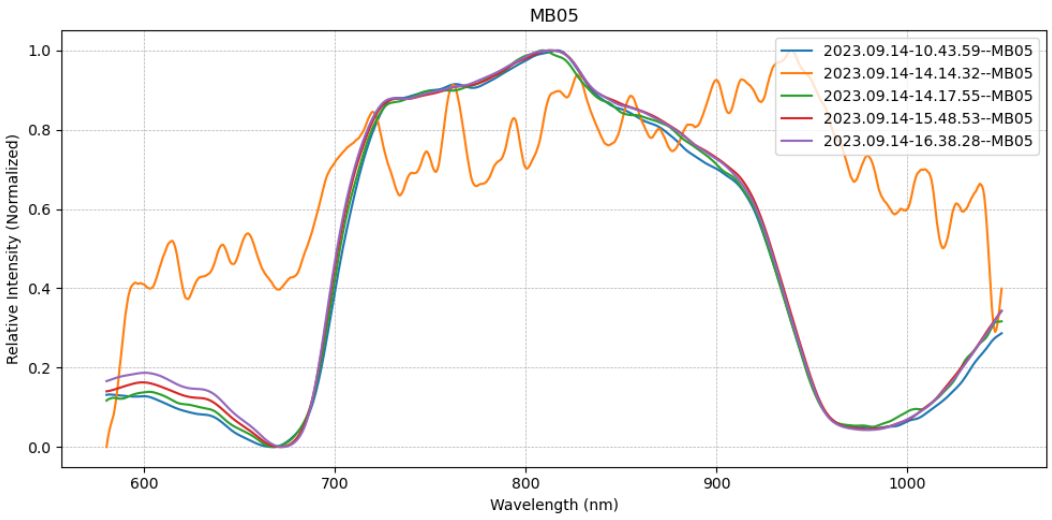

Figure 3.6: Spectral Data of Sample MB05, where the vertical axis represents the Normalized Relative Intensity and the horizontal axis represents the Wavelength in [nm].

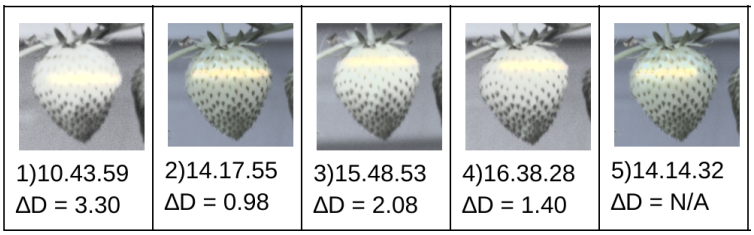

Figure 3.7: Image samples of a single strawberry captured at various times throughout the day, correlated with the spectral data of MB05.

- For sample 1): This sample produced a smooth wavelength despite a few anomalies: the strip of light is shifted to the right and downward, and ΔD far exceeds the acceptable range.
- For samples 2) to 4): All the samples yielded smooth wavelengths. However, only 2) falls within the correct ΔD range and has a perfectly positioned strip of light.
- For sample 5) : The strip of light is a bit outside the strawberry (to the left) affecting the reading of the spectrometer.

3.1.4 MB06 Analysis

Lastly, we examine the MB06 data, focusing on how sensor positioning influences the spectroscopy signal. The Normalized Relative Intensity and the Distance Offset from the sensor to the strawberry are displayed on Figures 3.8 and 3.9.

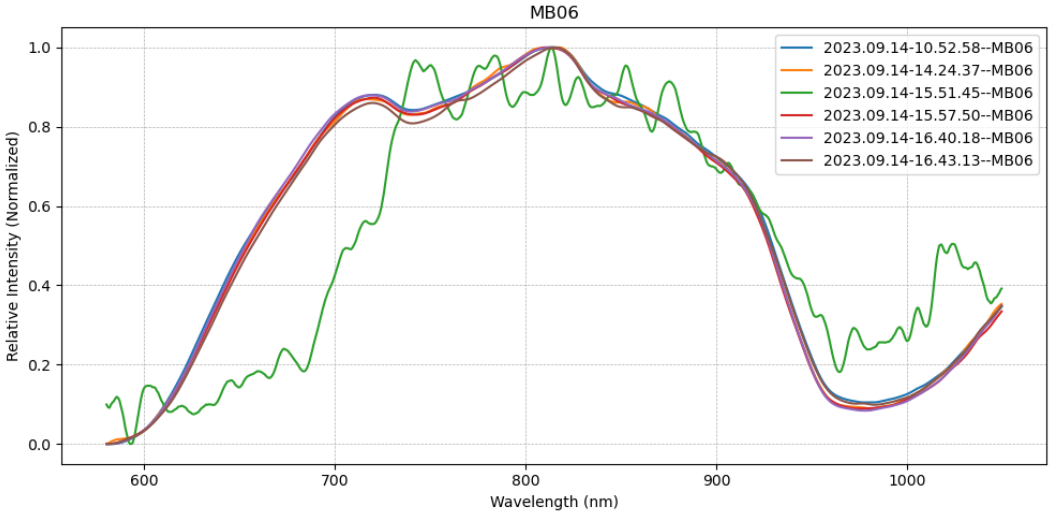

Figure 3.8: Spectral Data of Sample MB06, where the vertical axis represents the Normalized Relative Intensity and the horizontal axis represents the Wavelength in [nm].

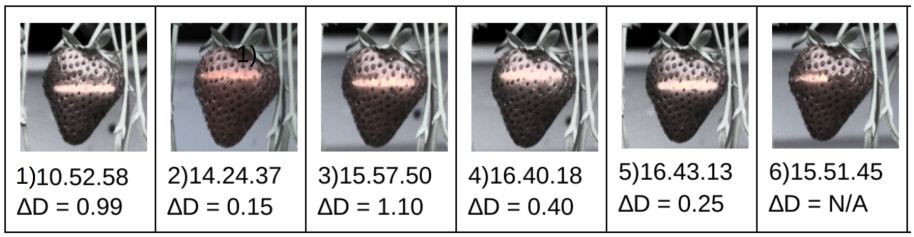

Figure 3.9: Image samples of a single strawberry captured at various times throughout the day, correlated with the spectral data of MB06.

- For samples 1) to 5): ΔD are within the acceptable range, except for 3), which slightly exceeds the range. Despite the varied positions of the strips of light on this strawberry, the wavelength remains smooth in all instances.
- For sample 6): Part of the strip of light is significantly shifted to the left of the strawberry's surface. Likewise, the inspection point may also be misaligned, which had an impact in the reading of the spectrometer.

The analysis conducted here was relevant in order to establish the 'Strawberry Inspection Criteria' to be presented in the Section 4.1.

3.2 Fruit Instance Segmentation Datasets

The datasets for fruit instance segmentation are composed of multiple scenes within polytunnels, captured using stereo cameras, where each scene contains multiple strawberries. Because the fruit detection models that we are using focus on instance segmentation, each strawberry instance was labeled with a unique identifier to distinguish individual objects within the same category. This process, known as instance labeling, requires marking the pixel-level region inside each fruit with a distinct mask. Thus, each strawberry within an image is treated as a separate instance, allowing the models to identify each instance independently. In addition to the instance labeling, the fruit detection models also necessitate assigning a class to each instance. In this work, that means that every strawberry, while being identified as a separate entity with its unique identifier, also needs to be categorized under one of three classes: Ripe, Unripe or Affected. Figure 3.10 shows examples of strawberries from each of the mentioned classes. Subsequently, Figure 3.11 displays an example of a scene and its corresponding instance label from the Dataset.

For training, validation, and testing purposes, the dataset employed was obtained from polytunnels in Norway in 2019 (NO2019 Dataset). We labeled the dataset to include only the categories of ripe, unripe, and those affected by

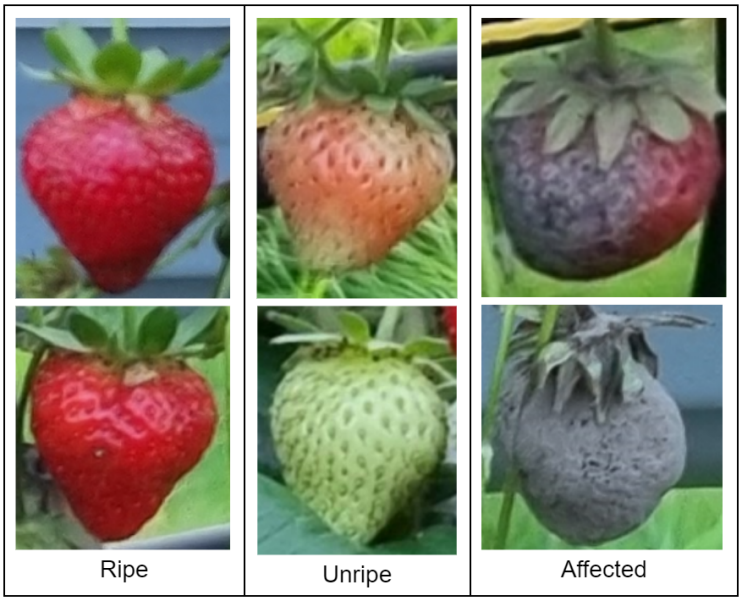

Figure 3.10: Two examples of each class of strawberries: Ripe, Unripe, and Affected

Figure 3.11: Example of Instance Segmentation for Strawberry Detection. Top image: Captured scene in the polytunnel featuring multiple strawberry instances. Bottom image: Corresponding instance segmentation label, where each color represents a unique instance of a detected strawberry. The class information for each instance is stored separately in an associated metadata file, not visually represented in this image.

fungus or other diseases. To more effectively evaluate the results that will be presented in the results section, we have also used a dataset from the United

Kingdom (UK Dataset) as reported in [9]. Additionally, we self-collected a dataset from Norway in 2023 (NO2023 Dataset) and created another dataset from a simulated strawberry polytunnel environment (this last one is explained in detail in Section 3.4). The labeling process for these datasets was conducted manually, using an annotation tool [37].

3.3 Widest Region Detection Datasets

Utilizing the instance segmentation datasets, we processed the data to obtain individual strawberry images. This derived dataset specifically targets the detection of the widest horizontal region of each fruit. The 'widest region' is defined as the area on a strawberry that has the largest horizontal span when viewed from the camera's perspective, essentially the part of the fruit that extends the most from one side to the other. In this context, the X-coordinate is determined by the central point along the horizontal axis of this widest region, and the Y-coordinate corresponds to the vertical position of this midpoint on the strawberry. For each strawberry image, the dataset defines the output as a two-dimensional vector, indicating these pixel coordinates of the widest region.

In order to conduct the labeling process, we developed a custom tool that labels the location of the widest region for each fruit. The user interface of the tool displays a single strawberry image and allows the labeler to choose the widest horizontal span with a simple click. This action then generates a label consisting of a pair of values, corresponding to the X and Y coordinates of that instance, ready for use in training. We selected only a subset of each dataset for labeling, due to the high volume of strawberries present in the scenes. Examples of these labelings are illustrated in Figure 3.12.

3.4 Simulated Test Set

In order to create a simulated test set, we used the strawberry plant generator to be described in Section 4.2.2 to randomly generate a total of 30 scenes encompassing a diverse array of strawberry plants, with a cumulative count of 274 strawberries. For annotating this simulated dataset, we employed the same instance segmentation annotation tool referenced in [37], as well as our labeling tool for determining the widest region of each strawberry, as previously described. It is important to highlight that the simulated dataset is limited to only two classes: 'Ripe' and 'Unripe'.

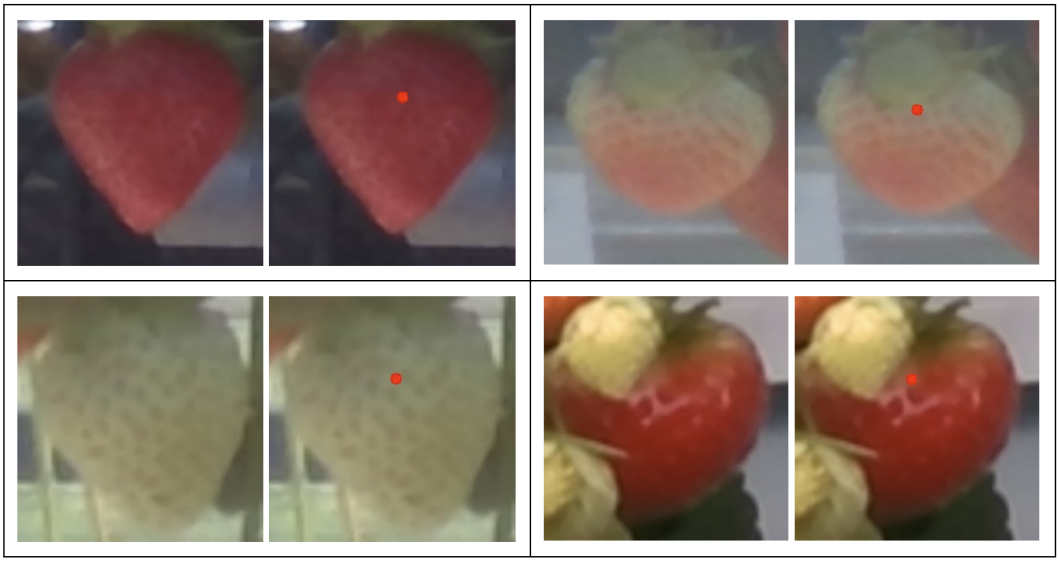

Figure 3.12: Four illustrative examples from the widest region detection dataset. The images are labeled with the X and Y pixel coordinates of the widest region, indicated by the red dots in the images.

3.5 Datasets Summary

The datasets for Fruit Instance Detection and the Identification of the Widest Regions used in this research are summarized in Table 3.1.

Table 3.1: Overview of the datasets used for fruit instance detection and widest region identification. Datasets marked with (*) were split into training, validation, and testing sets with proportions of 80%, 10%, and 10% respectively. For the remaining datasets, all data was used for testing purposes. The instance segmentation datasets are scenes containing many strawberries, while for the widest region detection datasets are single strawberries.

Dataset	**# Training Images**	**# Validation Images**	**# Testing Images**
Fruit Instance Segmentation* (NO2019)	1445	181	180
Widest Region Detection* (NO2019)	5484	685	685
Fruit Instance Segmentation (UK/NO2023)	-	-	45
Widest Region Detection (UK/NO2023)	-	-	315
Fruit Instance Segmentation (Simulated)	-	-	30
Widest Region Detection (Simulated)	-	-	274

4
Models and Methods

This chapter describes the methodology adopted in this research, which begins with the Strawberry Autonomous Inspection Criteria (Section 4.1) and then presents the real and simulation robot setup (Section 4.2).

Next, we describe the fruit instance segmentation models (Section 4.3), discussing the nuances of both a baseline model and an enhanced model. We then detail our approach to identifying the widest region of the fruit (Section 4.4), comparing a contour-based detection approach with a learning approach. Upon establishing these methods, we proceed to outline the integrated pipeline configuration designed for practical application, as detailed in Section 4.5.

Finally, we introduce our network gradient-based method aimed at maximizing the spectrometer accuracy through positioning, as detailed in Section 4.6. The approach begins with data acquisition, utilizing a grid scanning technique that scans across various positions on the strawberry. The method proceeds with selecting a reward function that simulates sensor accuracy, developing an accuracy estimation model, and applying the network gradient-based method to Visual Servoing.

4.1
Strawberry Autonomous Inspection Criteria

The capability of the Spectroscopic sensor to accurately measure the sugar content in strawberries is generally linked to the sensor's positioning relative to the fruit, as investigated in the Section 3.1. Considering the outlined factors, we define the following criteria that our autonomous inspection system must meet:

- Scan Ripe and Unripe strawberries only;
- Ensure the strip of light stays within the strawberry's surface;
- Position the inspection point directly on the strawberry;
- Maintain the sensor within the specified distance offset (20 ± 1 cm);
- Avoid occlusions, such as leaves, stems, and runners.

Following the inspection criteria above, Figure 4.1 visually demonstrates how the strip of light from the spectroscopy system targets the strawberry and

the point where the data is read, denoted inspection point. Additionally, the figure presents scenarios to be autonomously avoided, as they do not meet the inspection criteria, and potential solutions for these cases through autonomous sensor positioning.

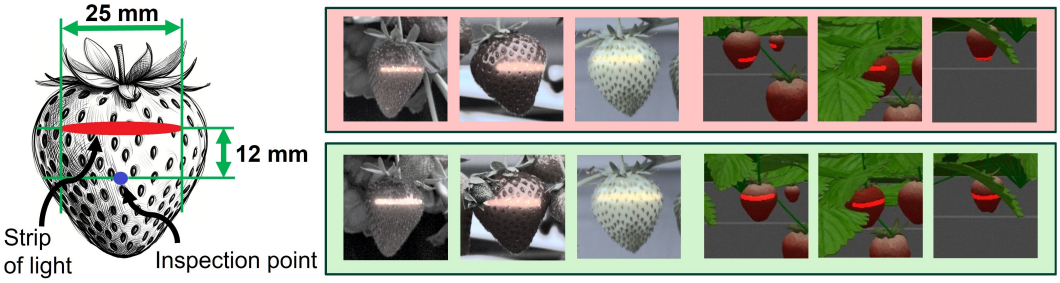

Figure 4.1: Illustration of the spectroscopy system operation (on the left) and examples of undesirable positions (highlighted with a light red background) along with the solutions for these (highlighted with a light green background).

In this work, we address autonomous sensor positioning using a mobile manipulator and a vision system, which is discussed in the following sections.

4.2 Robot Setup and System Design

In this section, we present our robot setup and system design of our real-world mobile manipulator, detailing the integration of the spectroscopy system and a 3D camera. We also present a simulation that mirrors the physical setup, including the strip of light of the spectroscopy system and the 3D camera. Additionally, we present virtual sensors to simulate the spectroscopy accuracy, a virtual strawberry generator and the emulation of strawberry polytunnels which are used in this work.

4.2.1 Real Mobile Manipulator

Our mobile manipulator design integrates the Omnidirectional Agricultural Robot Thorvald Slim [38] with the industrial robotic arm Mitsubishi RV-2AJ. The robotic arm is mounted on the agricultural robot and features 5 Degrees of Freedom (DoF). At its end effector, it is equipped with a 3D camera to visualize the environment and a spectrometer system used for measuring the sugar content in strawberries, configuring a eye-in-hand setup. The design of our mobile manipulator is presented in Figure 4.2, illustrating the integration of components.

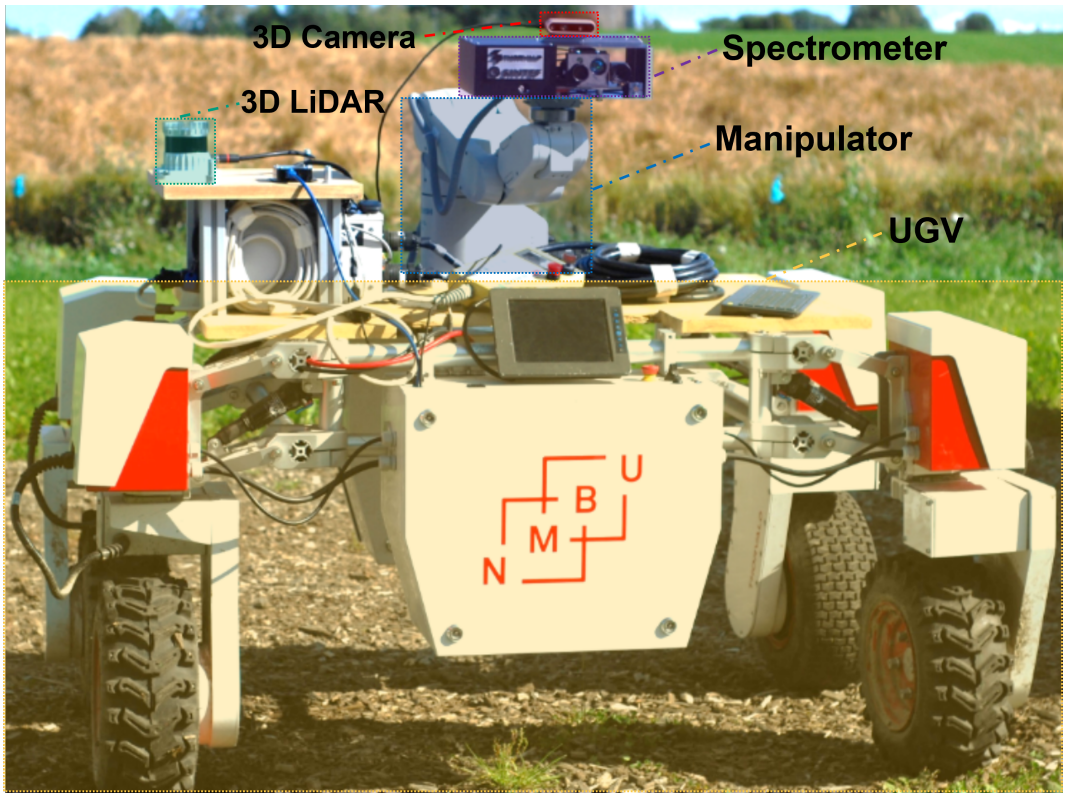

Figure 4.2: Mobile Manipulator comprising the Thorvald Slim UGV, the Mitsubishi RV-2AJ Manipulator, equipped with the Spectrometer System and a 3D Camera. Additionally, a 3D LiDAR is included specifically for future applications in autonomous navigation.

4.2.2 Simulation

We conducted an enhanced simulation that closely mirrors the real-world equivalents of the agricultural robot, robotic arm, 3D camera, and spectrometer system, along with the polytunnel environment and a strawberry plant generator that includes the simulation of strawberries.

Initially, we show a comparison between the real and simulated mobile manipulators, as presented in Figure 4.3.

As illustrated in Figure 4.3, the model of the 3D camera was mounted onto the model of the spectrometer system in a configuration that replicates their real-world positioning, which is shown in further detail in Figure 4.4.

Additionally, it is important to mention that the simulated spectrometer system does not emulate spectral readings, but we addressed the simulation of the strip of light projector. This enhancement involved the use of multiple virtual lamps, initially oriented to project light forwards. To accurately replicate the real rectangular projector's illumination, we developed a script that positions these lamps within the spectrometer model for desired angles. We

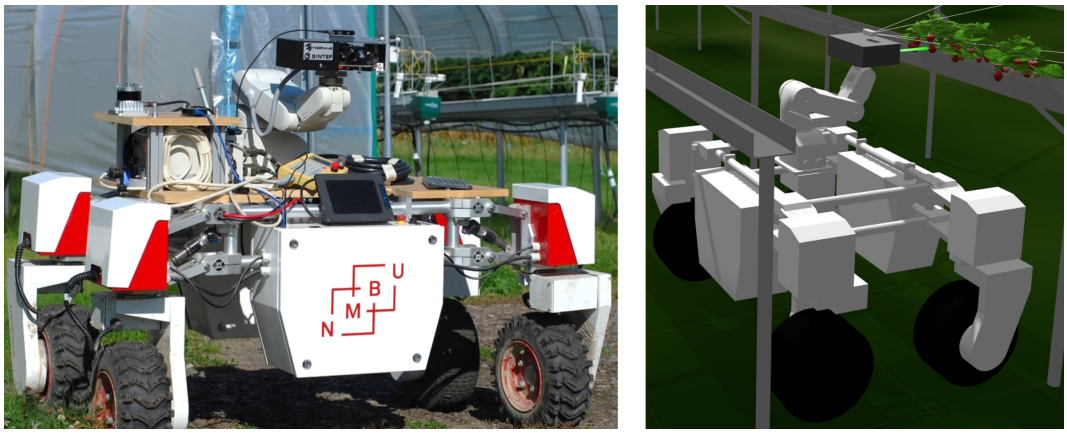

Figure 4.3: Comparison Between the Real (left) and Simulated Mobile Manipulator (right).

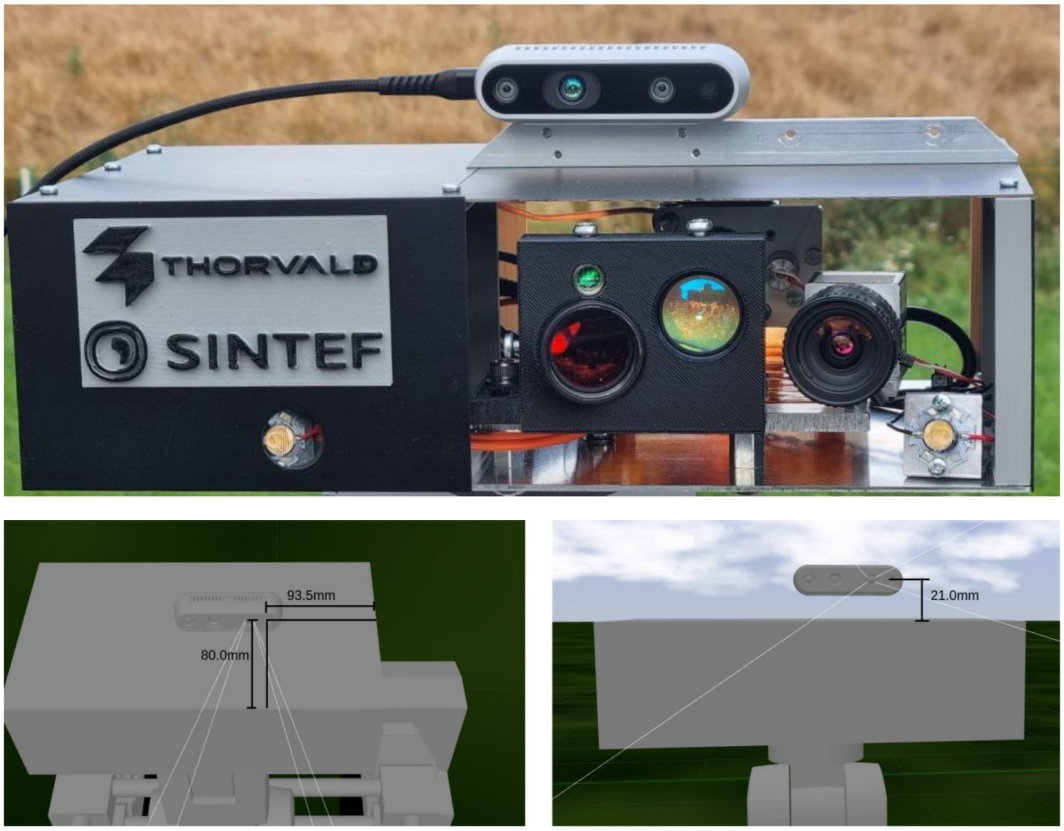

Figure 4.4: Illustration of the Real (up) vs. Simulated (down) Sensors Positioning: 3D Camera and Spectrometer System. The simulated spectrometer system matches the real spectrometer in dimensions, and the coupling of the 3D camera to the sensor also matches the real-world configuration.

calculated the necessary aperture angle and subdivided it into smaller angles. This division allowed the alignment of several lamps, each with a projection diameter of 3mm at a distance of 20cm. Such adjustments in the lamp positions enabled the accurate projection of the rectangular shape, measuring 25mm in width by 3mm in height, at a distance of 20cm from the spectrometer sys-

tem to a target fruit. Figure 4.5 compares the real strip of light projected to a real strawberry against the simulated strip of light projected to a virtual strawberry.

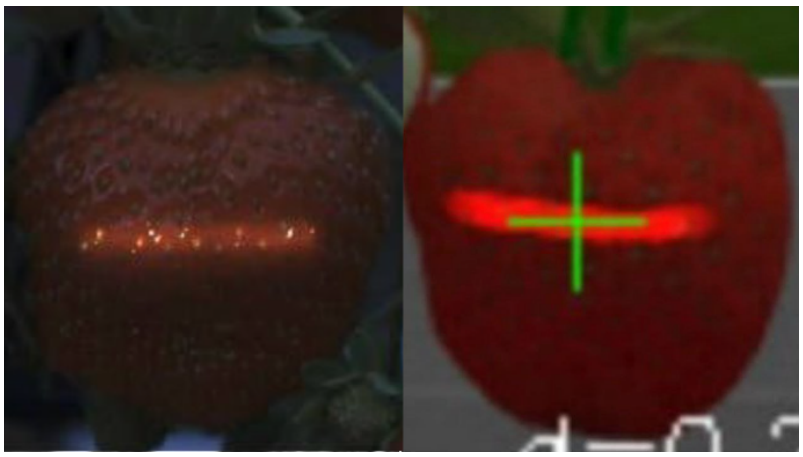

Figure 4.5: Comparison of the real strip of light projected on a real strawberry (left) versus the simulated strip of light projected on a virtual strawberry (right).

In our simulation, two Laser Range Finders (LRFs) have been integrated to provide sensing for the positioning of both the strip of light and the inspection point on the strawberry's surface. We have adjusted the LRF for the strip of light to have the same aperture angle as the array of lamps that we implemented in the simulation, ensuring that we capture all readings from the strip of light. For the inspection point, we have set a single reading for the LRF to sense that point at 12mm below the center of the strip of light. The main reason for this is to simulate the spectrometer accuracy, but it will be explained in more detail in Section 4.6. Figure 4.6 illustrates both the beams of the strip of light projector and the sensing beams of the LRFs.

For the simulation of strawberry plants, we used a randomized strawberry plant generator [39]. This tool enabled the specification of a variety of features ranging from berry models and textures to leaf orientations, as outlined in Table 4.1. Additionally, we modified the generator to produce virtual strawberries at varying heights, thereby introducing additional complexity into the test set to better mimic real-world conditions. Figure 4.7 provides an example of a comparison between a real scene and a simulated scene.

Lastly, we present the simulation of polytunnels which can be observed in Figure 4.8.

Another important implementation for the simulation is the automatic distribution of random strawberry plants within the polytunnel trails. This implementation allows for the predefined specification of both the desired number of strawberries per row and the total number of rows. The system

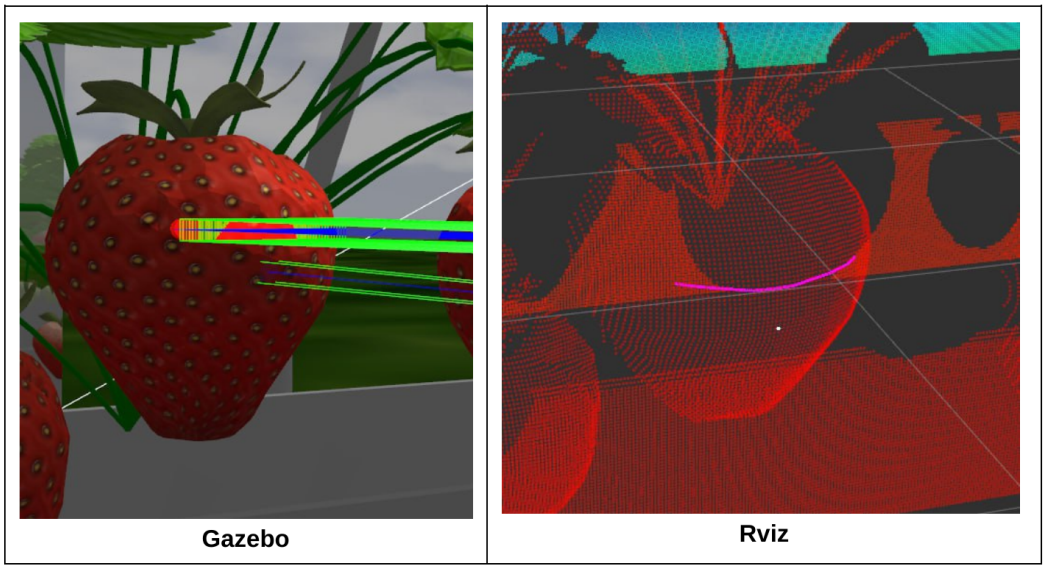

Figure 4.6: llustration of the strip of light projector beam and the sensing beams of the LRFs. The left image shows the simulation in Gazebo with the strip of light beams in green and the LRF beams in blue. The right image shows the LRFs readings in RViz, with the strip of light sensing in pink and the inspection point sensing in white. Notice in these images that the projector beams are aligned with the sensing beams, ensuring that all readings from the extent of the strip of light and the inspection point are captured by the LRFs.

Figure 4.7: Comparison between a real scene (left) and a simulated scene (right).

automatically aligns these plants along the trails, ensuring a consistent spacing with a small random deviation between each plant. Figure 4.7 presents 3 plants within a single trail of the polytunnel, while Figure 4.8 similarly displays 15 plants in a single trail. For the purposes of our study, we will adopt the configuration as presented in Figure 4.7, utilizing 3 plants in a single trail.

Table 4.1: Diversity of the original Strawberry Plant Generator

Feature	**Description**
Berry Models	5 unique models representing shapes
Berry Textures	7 textures representing stages of ripeness
Berry Size	Randomized berry size
Berry Pose	Randomized position and orientation
Calyx Models	10 unique models
Stem Models	5 unique stem models for different leaf heights and berry orientations
Leaf Models	5 unique models
Leaf Pose	Randomized position and orientation

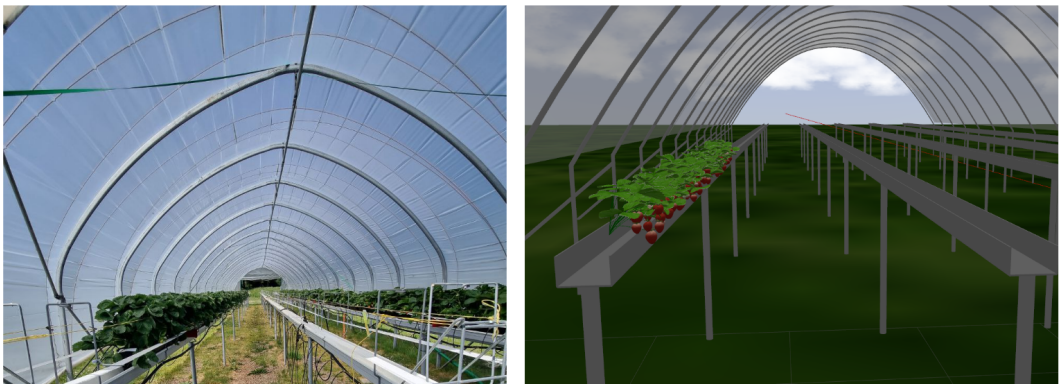

Figure 4.8: Comparison between the real polytunnel (left) and simulated polytunnel (right).

4.3 Fruit Instance Segmentation Models

This subsection examines the Deep Learning segmentation techniques for fruit instance detection adopted in this paper. The discussion begins with the baseline method, MaskRCNN, a widely used model for the identification and classification of fruits in complex agricultural environments. This model, as well as the subsequent enhanced approach, relies on the labels described for their training process. The enhanced approach builds upon the capabilities of the baseline MaskRCNN model, aiming to achieve more accurate segmentation performance specifically tailored to the unique challenges encountered in fruit instance detection.

4.3.1 Baseline: MaskRCNN

The baseline model that we employed for the instance segmentation task is MaskRCNN, responsible for taking an RGB image as input and identifying the location and classification of each strawberry in the image. This model serves as the basis upon which we compare the performance of an

enhanced segmentation approach. In this model, the input is processed through a Backbone Network (e.g. ResNet [24]) combined with a Feature Pyramid Network (FPN), to produce feature maps. These maps are then used by a Region Proposal Network (RPN) to propose potential object locations. The next stages of the model refine these proposals, obtaining bounding boxes and assigning class labels, while simultaneously generating segmentation masks for each instance. The output is a combination of these masks, bounding boxes, and class labels for each object, such as strawberries, in the image. The architecture of MaskRCNN is illustrated in Figure 4.9.

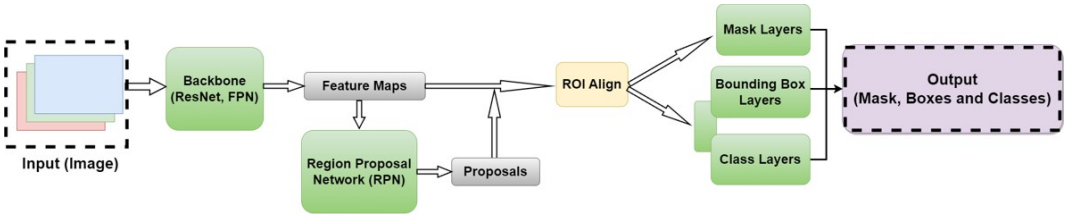

Figure 4.9: MaskRCNN architecture flowchart showing the process from input image to the output.

In the Section 4.3.2, we present the modifications to the backbone of the MaskRCNN framework, aimed at improving the fruit instance detection.

4.3.2 Improved: MaskRCNN-D

The enhanced version of MaskRCNN, denoted as MaskRCNN-D, integrates depth information into the original architecture. Modifications to the backbone network were necessary to accommodate the additional depth channel, which were implemented following recommendations from the original GitHub wiki for MaskRCNN [1]. Figure 4.10 illustrates this integration of depth information into the MaskRCNN model.

The inclusion of depth information addresses a significant challenge in fruit instance segmentation: the high incidence of occlusion among the fruits. When fruits are clustered together, traditional RGB data may not provide enough differentiation for the algorithm to accurately segment each instance. Depth data introduces a new potentially lighting-invariant dimension of information that significantly aids in distinguishing fruits that are in close proximity, particularly in terms of their relative positions along the depth axis.

[1]MaskRCNN Wiki, 2018. Available: https://github.com/matterport/Mask_RCNN/wik [Accessed November 5, 2023]

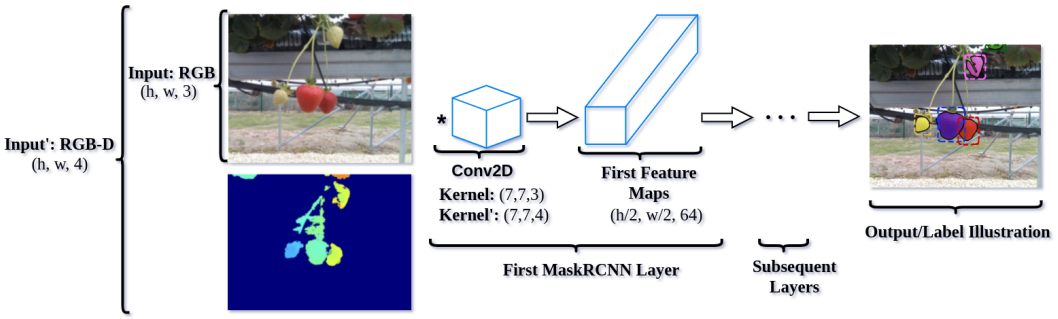

Figure 4.10: Illustration of the MaskRCNN model modified to integrate depth information within its backbone. The inputs are shown as Input (RGB) for the original MaskRCNN and Input′ (RGB-D) for MaskRCNN-D, indicating the addition of depth data. The kernels Kernel and Kernel′ are shown to represent the convolutional operations in each model. The output illustrates the instance segmentation with overlaid bounding boxes and segmentation masks, highlighting the detected instances. Each detected instance is also classified into one of the predefined categories, but these classifications are not represented in this figure.

This adjustment was inspired by the unpublished research of (Orestis, 2018)[2], who demonstrated up to 31% AP (Average Precision) increase in performance on various Datasets when incorporating depth data into the model.

4.4 Fruit Widest Region Detector

This section introduces two approaches for detecting the widest region of a strawberry, the primary objective of which is to accurately pinpoint the X and Y coordinates of this region. The first approach is the ContourMax method, a direct and learning-free approach that operates without the need for pre-labeled data. The second is a Deep Learning alternative, requiring labels for its training as mentioned earlier.

4.4.1 Baseline: ContourMax

Our algorithm, referred to as ContourMax, processes an input known as contour, which is derived from instance segmentation or instance labeling that delineates the outline of an object. It is designed to pinpoint the widest horizontal segment of a strawberry by looping through each unique Y-coordinate of the contour data. For each Y-level, it determines the horizontal span by locating the extreme X-coordinates that lie on this horizontal line. The pro-

[2]Orestis, 2018, "Does Depth Matter? RGB-D Instance Segmentation with MaskRCNN", unpublished manuscript, available at: https://github.com/orestis-z/mask-rcnn-rgbd. Accessed on: November 5, 2023

cess involves comparing each width to find the maximum, updating this value along with the corresponding Y-coordinate when a wider segment is identified. The algorithm concludes by returning the Y-coordinate of the maximal width and the X-coordinates of the boundaries of this segment. The Y component of the widest region corresponds to the identified Y-coordinate, and its X component is defined as the midpoint between the X-coordinates of the segment's boundaries. An illustration of this algorithm in discerning the widest region of a strawberry by its contour is shown in Figure 4.11.

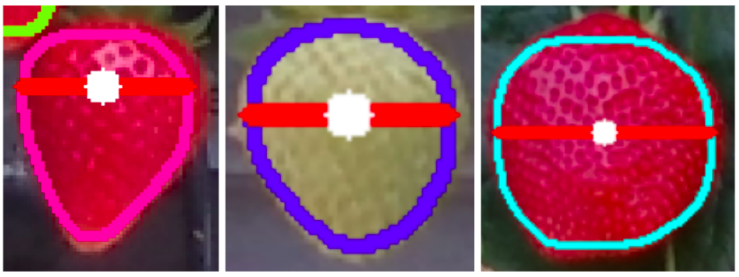

Figure 4.11: Illustration of the baseline ContourMax algorithm identifying the widest region of a strawberry. Each panel shows an instance of a strawberry with its contour highlighted, and the widest region marked with a horizontal red line which is the output of the ContourMax algorithm. The white dot represents the X and Y pixels for the widest region.

4.4.2 Improved: VGG-WSCNN

To improve the precision in identifying the widest region of the strawberries, we employed an enhanced version of the VGG-16 architecture by incorporating Weight Standardized Convolutions (WSCNNs). This choice was made because the classic VGG-16 model did not offer enough precision for accurately pinpointing the widest region of the strawberries. These modifications have been applied to the non-residual model structure of VGG-16, and according to the authors, such advancements offer greater stability during training, a reduced tendency for overfitting, and improved generalization capabilities [40, 41]. In this enhanced model, we modified the output layer to act as a regressor, which is illustrated in Figure 4.12, outputting two values representing the X and Y coordinates of the widest region's pixels.

Coupled with this Deep Learning approach, we implemented a preprocessing step that augments the reliability of input data. To mitigate the inaccuracies from instance detectors, which can arise due to occlusions from overlapping strawberries or leaves, the system generates a bounding box around the initial contour from the instance detector (or label). This box is then

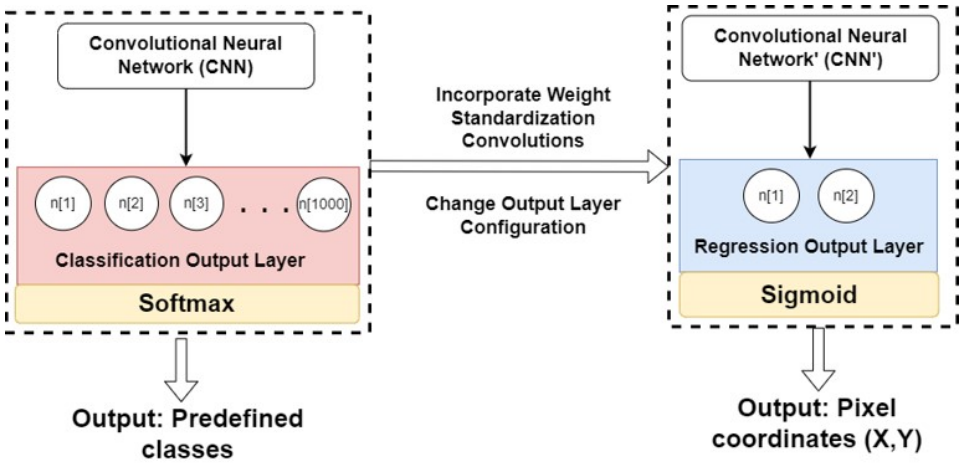

Figure 4.12: Modification of the VGG-16 architecture to incorporate WSCNNs and a regression output layer. On the left, the original CNN architecture with a classification output layer using softmax for predefined classes. On the right, the modified CNN' with an output layer configured for regression, using a sigmoid function to output pixel coordinates (X,Y) for the widest region of the strawberries.

expanded, forming a crop region that ensures inclusion of the full strawberry within the regressor's analysis frame. Such enhancement of the input area adds robustness against incomplete detections, helping the regression model to accurately locate the fruit's widest section.

After describing the two methods implemented for detecting the widest region of strawberries (Contourmax and VGG-WSCNN), Figure 4.13 provides a simplified comparative visualization of the two approaches.

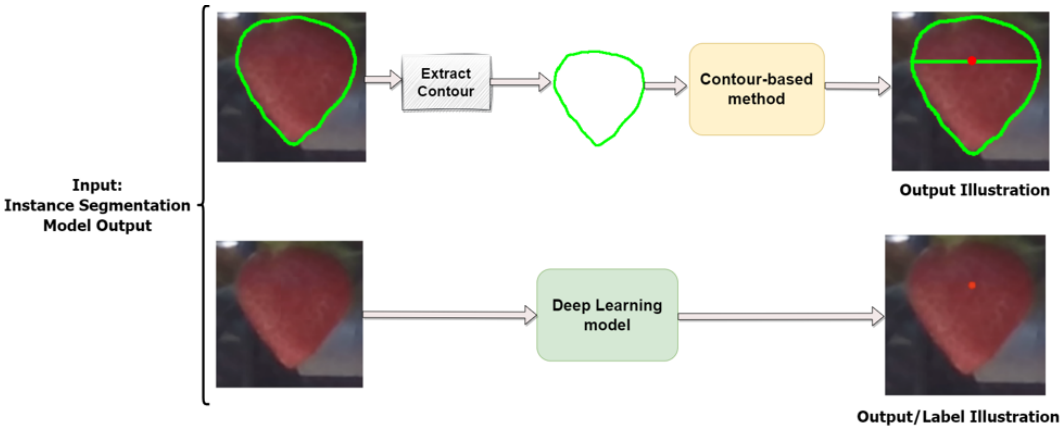

Figure 4.13: Simplified comparison of Widest Region Detection Methods: The top pathway shows the ContourMax method, from the extraction of the contour from the instance segmentation model output to identifying the coordinates of the widest segment. The bottom pathway illustrates the Deep Learning approach, utilizing the modified VGG-WSCNN model to infer the widest region coordinates.

4.5 Fruit Detection Pipeline and Strawberry Approach

The integrated Fruit Detection Pipeline is built upon the combination of the Fruit Instance Detection and Widest Region Detection methods, with the incorporation of point cloud data from a 3D camera. This pipeline operates autonomously to process scenes with multiple plants and strawberries, ultimately identifying the widest region of each strawberry in 3D space. The schematic of the pipeline can be observed in Figure 4.14.

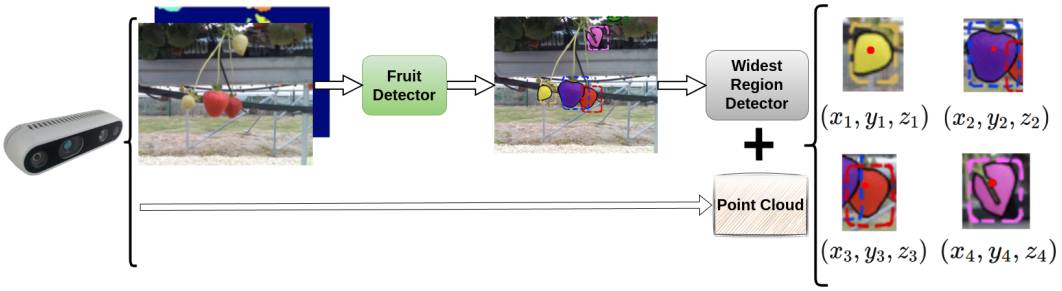

Figure 4.14: Illustration of the Fruit Detection Pipeline workflow, beginning with the input of RGB image data (scene) and depth information, which may vary depending on the method used. The scene contains strawberries, and the output is the 3D location of each strawberry within that scene.

The objective of the pipeline is to generate goals targeting the widest region of strawberries for moving the spectrometer to specific positions through the use of Position-Based Visual Servoing (PBVS). An important aspect of the goals is maintaining a frontal distance of 20cm between the spectrometer and the target fruit, as mentioned in the problem description.

At this point, we introduce the coordinate systems to be used from now on in this work, as shown in Figure 4.15.

The first coordinate system expresses the camera's position at a point $p = [x, y, z]$ relative to the robotic arm's base. The second coordinate system indicates the camera's position at a point $\bar{p} = [\bar{x}, \bar{z}]$ relative to the current strawberry's widest region, discarding the $\bar{y}$ coordinate because it is kept fixed at 20cm from the spectrometer. Furthermore, the displacement vector between two camera positions $\bar{p}_1$ and $\bar{p}_2$ in the plane is defined in Equation 4-1.

$$d = \bar{p}_2 - \bar{p}_1 = [\Delta x, \Delta z]\,, \tag{4-1}$$

4.6 Network Gradient Visual Servoing Approach

The precise position of the widest region, achieved using the Multi-Step PBVS (to be detailed in Section 4.6.4), may not be optimal in terms of sensor

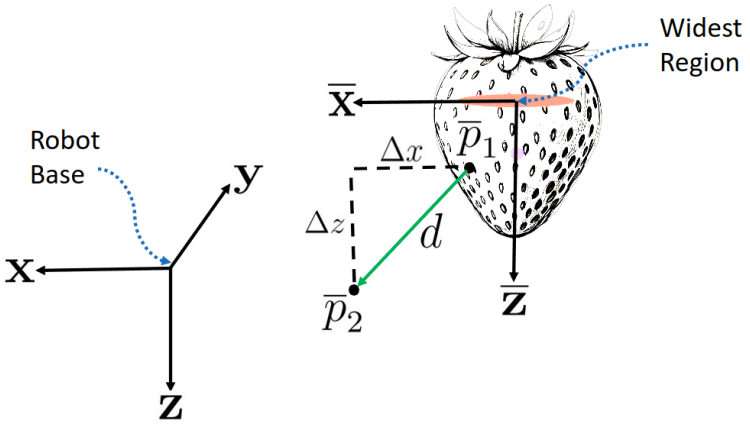

Figure 4.15: Coordinate Systems used in this work. The camera is positioned to face toward the strawberry's front. The displacement vector d indicates the vertical (Δz) and horizontal (Δx) camera movement from a point $\bar{p}_1$ to a point $\bar{p}_2$ in the plane.

accuracy. We therefore subsequently apply an Image-Based Visual Servoing (IBVS) method to improve the position. In order to lay the groundwork for our improved IBVS method, we initially acquire a dataset through grid scanning (Section 4.6.1). Next, we design a reward function to label the dataset and then prepare it (Section 4.6.2) for training our accuracy estimation model (Section 4.6.3). Following this, we present an initial solution (Section 4.6.5) for the optimal positioning problem determined by the sensor accuracy prediction from the estimation model. Finally, we utilize the estimation model and apply gradient theory to develop our Network Gradient Visual Servoing approach (Section 4.6.6).

4.6.1 Data Acquisition Through Fruit Grid Scanning

The data acquisition process is derived from our grid scanning method implemented. This method leverages the Multi-Step PBVS to scan specific points on each strawberry, thereby saving the camera positions relative to their widest regions, as well as the LRFs' readings, and fixed-size images around the center of the spectrometer positioning for these displacements. These images can contain part of a strawberry, a whole strawberry, or multiple strawberries, depending on the displacement. These measurements are utilized as inputs for a reward function, which will be detailed in the next section.

To obtain a grid list of all the points for a single strawberry, the process begins by using the segmented mask, obtained from the fruit detector, for the specific fruit. This mask is dilated to ensure coverage of all the border pixels of the strawberries. Subsequent, binary erosion is applied to the dilated

mask to slightly shrink it, followed by the selection of a subset of pixels at regular intervals, as defined by a specified spacing λ. This creates evenly spaced points, reducing the number of target pixel points. Figure 4.16 illustrates the acquisition of pixel targets through the image processing of the mask obtained by the fruit detector.

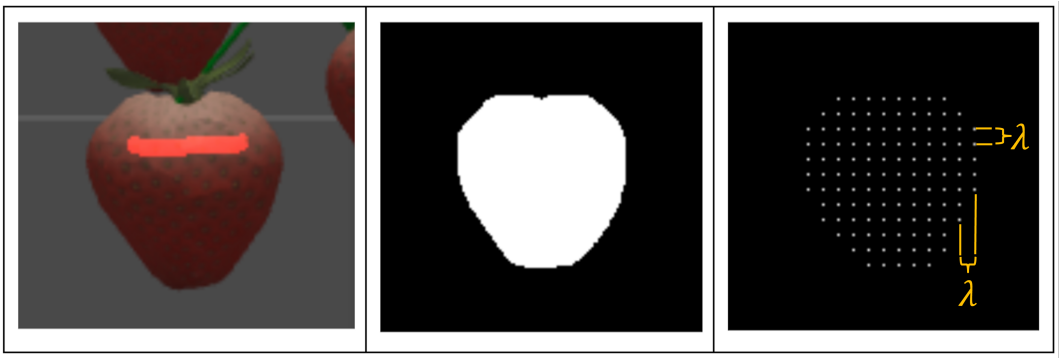

Figure 4.16: Illustration of pixel target acquisition through the image processing of the mask obtained by the fruit detector. The images show, from left to right, an image of the strawberry, the enhanced mask, and finally, the evenly spaced points with spacing λ. The targets are subsequently transformed from pixel to 3D coordinates.

Finally, a Scaling Factor (SF) is calculated to convert the pixel spacing λ directly into 3D coordinates, considering a fixed forward distance (ρ) to the fruit. This distance includes the 20cm specified in the inspection criteria in Section 4.1, plus an additional 8cm accounting for the offset from the camera to the front of the spectrometer. Therefore, given this setup, the pixel spacing λ is multiplied by the SF to transform the targets from pixels to 3D coordinates in centimeters, allowing the sensor to be moved in the plane at the distance ρ during the grid scanning process. An illustration of how the SF is calculated is presented in Figure 4.17. By employing the small angle approximation, a fixed scaling factor is derived from the center of the image.

After detailing the process of obtaining a grid list for each strawberry and specifying the data to be saved, we apply this procedure to a chosen number of strawberries in order to build a dataset, as outlined in Algorithm 1. Within the simulation environment, we implemented a technique to systematically generate new strawberry plants in the virtual space, replacing the previous ones. This allows for the creation of a varied dataset without the need to move the mobile base, as this work is only focused on the robotic arm's positioning.

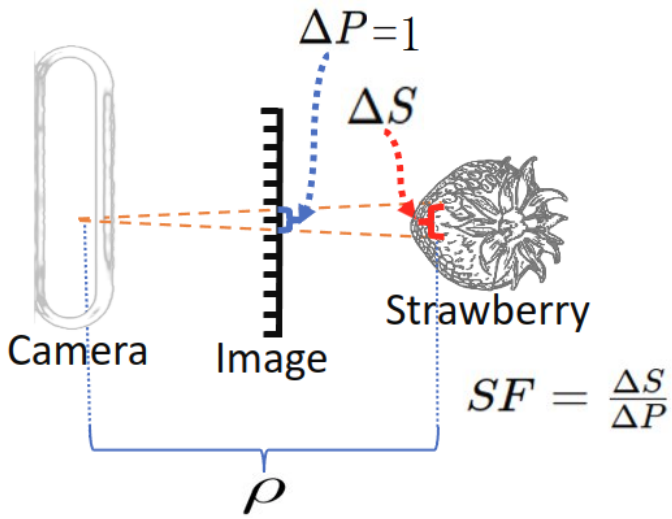

Figure 4.17: Perspective projection illustration for Scaling Factor (SF) calculation. The diagram shows a camera facing a strawberry at a fixed distance ρ. To calculate the scaling factor, consider two adjacent pixels ($\Delta P = 1$) and measure the physical distance between their projections on the strawberry (ΔS). The scaling factor is then determined by the ratio of ΔS to ΔP.

4.6.2 Reward Function and Preparation of the Data

The reward function was designed to simulate the accuracy of the spectrometer's readings, which are influenced by its positioning relative to a strawberry. Ideally, we aim to utilize the actual spectrometer accuracy, but the feedback for sensor accuracy measurement is an aspect yet to be integrated by the corresponding section of the project. Our reward function is calculated utilizing the Laser Range Finders (LRFs) readings for both the strip of light projection (LRF_A) and the inspection point (LRF_B). The calculation of our reward function is detailed in Equation 4-2.

$$\text{R} = -\left(\sigma_{\text{LRF}_\text{A}} + |\text{opt}_\text{dist} - \mu_{\text{LRF}_\text{A}}| + |\text{opt}_\text{dist} - R_{\text{LRF}_\text{B}}| + \Delta_\text{insp}\right) \tag{4-2}$$

where:

- $\sigma_{\text{LRF}_\text{A}}$ represents the standard deviation of the readings from LRF_A. It penalizes the misalignment between the strip of light and the fruit, which also accounts for occlusions. If $\sigma_{\text{LRF}_\text{A}}$ is large, this may indicate that the strip of light is partially outside the fruit's surface, or an occlusion, such as a leaf, is obstructing the light.
- $|\text{opt}_\text{dist} - \mu_{\text{LRF}_\text{A}}|$ calculates the absolute difference between the target optimal distance (opt_dist) and the average of the readings from LRF_A

Algorithm 1: Data acquisition through fruit grid scanning.

```
Data: Number of strawberries to scan
Result: Data acquisition for the strawberries
for strawberry in range(1, Number of strawberries to scan + 1) do
    Identify all widest regions of strawberries using the fruit
     detection pipeline;
    foreach strawberry widest region do
        Position the camera at that region using Multi-Step PBVS;
        Calculate the grid list for that strawberry;
        foreach scanning point in the grid list do
            Employ PBVS to position the camera accurately at the
             point;
            Save the following data:;
                Image around the center of the spectrometer FoV;
                Camera position relative to the widest region;
                LRFs' readings;
        end
        Move to the next strawberry widest region;
    end
    Remove all strawberry plants from the environment;
    Repopulate the environment with new strawberry plants;
end
```

($\mu_{\mathrm{LRF_A}}$). This term penalizes deviations of the strip of light from the predefined ideal distance.

- $|\mathrm{opt_{dist}} - R_{\mathrm{LRF_B}}|$ measures the absolute discrepancy between the target optimal distance and the scalar reading from $\mathrm{LRF_B}$ ($R_{\mathrm{LRF_B}}$). It penalizes deviations of the inspection point from the predefined ideal distance.
- Δ_{insp} is the absolute difference between the scalar reading from $\mathrm{LRF_B}$ ($R_{\mathrm{LRF_B}}$) and the central reading from $\mathrm{LRF_A}$ ($\overline{R}_{\mathrm{LRF_A}}$). It penalizes occlusions between the strip of light and the inspection point.

Building upon the data acquisition obtained in the previous section and applying the Equation 4-2, we present the rewards associated with the displacements for all the scanned strawberries. Initially, Figure 4.18 shows the rewards on a blue graph for horizontal displacements (with vertical displacements near zero), and on a red graph for vertical displacements (with horizontal displacements near zero).

Analyzing the blue graph in the Figure 4.18 we can observe that the horizontal axis is a bit symmetric, reflecting the near-symmetrical nature of the strawberries in the simulation. As the absolute value of Δx increases, the rewards decrease due to the strip of light beginning to deviate from the fruit's

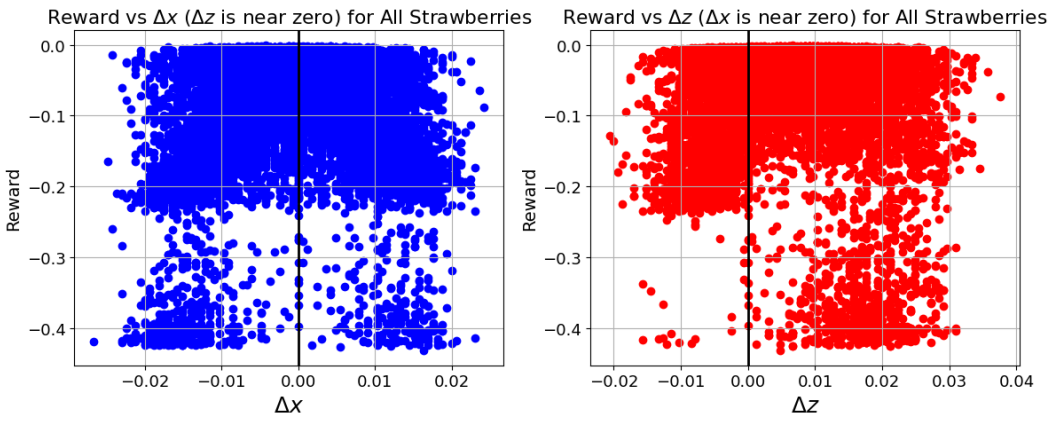

Figure 4.18: Reward Graphs × Displacements (in meters): Blue Graph for horizontal displacements and Red Graph for vertical displacements. The origin $(0, 0)$ represents the widest regions corresponding to zero displacement.

surface, which is even more pronounced with further displacement from the center.

Turning to the red graph in the same figure, we can notice when decreasing Δz values, the rewards increase, consistent with the strawberry being rounder at the top, which means the strip of light is more likely to be on the fruit's surface. Conversely as Δz increases, the rewards diminish because the strip of light might partially come off the fruit depending on the fruit's size. For elevated Δz absolute values, we can notice larger concentration of lower rewards even further depending on the fruit's size, not only because the strip of light might be slightly off the fruit but also because the inspection point is likely moving off the strawberry as well.

In the sequence, we present the heatmap for the rewards, showing how it is distributed for the displacements, of all the scanned strawberries, varying both Δz and Δy, in Figure 4.19.

In Figure 4.19, we observe that the highest concentration of rewards is not necessarily in the origin. This suggests that achieving the widest region does not guarantee the highest rewards. These findings underscore the need for a model that is capable of learning to maximize rewards, thereby optimizing the simulated sensor accuracy of the spectrometer.

Finally, we selected three strawberries data to observe the behavior of the reward curve in relation to the isolated Δz and Δx, as shown in Figure 4.20.

We can observe in Figure 4.20 that the reward-displacement curves for isolated Δx and Δz displacements are well-behaved. This investigation suggests that the choice of employing Deep Learning algorithms is potentially well-suited for this case, not being necessary the use of exploratory algorithms,

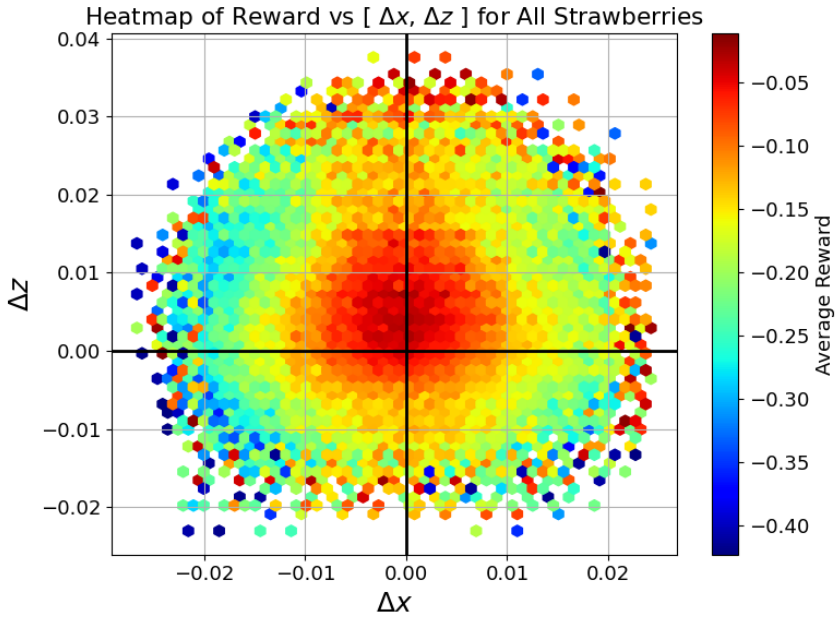

Figure 4.19: Heatmap of Reward × Displacements (in meters) for all the scanned strawberries. The origin $(0, 0)$ represents the widest regions corresponding to zero displacement.

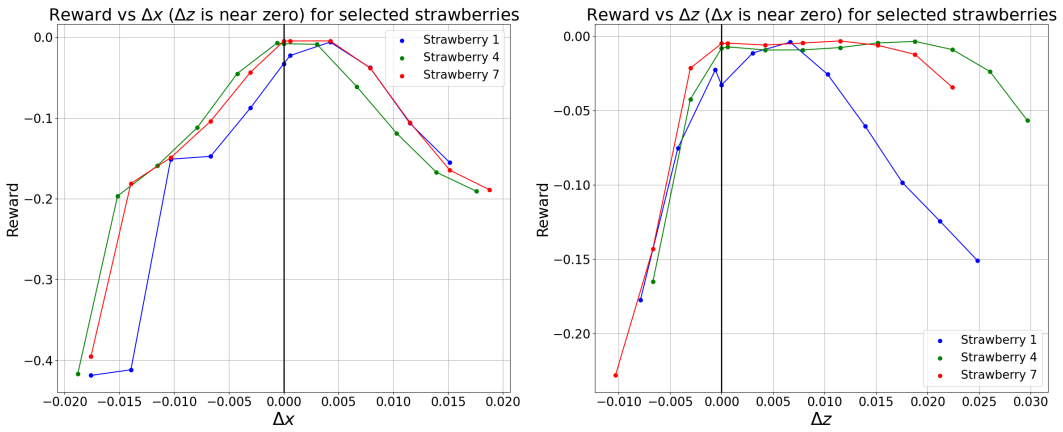

Figure 4.20: Reward Graphs × Displacements (in meters) for three selected strawberries.

such as Reinforcement Learning.

With this perspective in mind, we proceed with the preparation of labeled data for the accuracy estimation model, which will be presented in the next section. This preparation involves the creation of a dataset that will be utilized for training, validating, and testing the model. The inputs for data creation include two components: an image and a displacement, while the output comprises a single component: reward. All data components undergo normalization, especially the rewards, which are transformed into a measure of accuracy, taking on values ranging from 0 to 1. It is important to mention that each input-output pair is specifically associated with an individual strawberry.

The configuration of our input-output pairs is detailed as follows:

- Input 1: Image I_1 corresponding to a given camera position $\bar{p}_1$.
- Input 2: The displacement $\bar{p}_2 - \bar{p}_1$ between two camera positions.
- Output: The sensor's accuracy Acc_2 corresponding to the image I_2.

To generate the accuracy estimation dataset, we initially filtered the data acquired from the prior grid scanning procedure to isolate a total of 940 unique strawberries (associated with an overall total of 64,633 camera positions). This step ensures that the same strawberries are not used across the training, validation, and testing sets, maintaining distinct and separate samples for each phase. In the sequence, we establish the total number of samples for the dataset, where each sample consists of inputs/output pairs. Next, we determine the proportions of 80%, 10%, and 10% for the training, validation, and testing sets, respectively. We present in Algorithm 2 the method for generating a desired set.

Algorithm 2: Desired Set Generation

Result: A list containing the samples of a desired set.

Initialize an empty list for a desired set;
while *list size* $<$ *(proportion* $\times$ *total number of samples)*$+1$ **do**
    Select a random strawberry sample and a random displacement for the same strawberry with corresponding images I_1 and I_2, ensuring that $|\bar{p}_2 - \bar{p}_1| < 0.5$ cm ($\approx 20\%$ of the strawberry diameter, ensuring minor variations);
    Input 1: I_1;
    Input 2: $\bar{p}_2 - \bar{p}_1$;
    Output: Calculate Acc_2 derived from Equation 4-2;
    Add the pair (Inputs and Output) to the list;
end

A summary of the dataset obtained is shown in Table 4.2.

Table 4.2: Summary of the Accuracy Estimation Dataset. The total number of samples is 256,000.

Description	**# Training Data**	**# Validation Data**	**# Testing Data**
Strawberries	752	94	94
Samples	204800	25600	25600

4.6.3 Accuracy Estimation Model

The objective of the accuracy estimation model is to predict the potential sensor accuracy of a positioning adjustment by analyzing a current image and a desired displacement. To achieve this, the architecture of the accuracy estimation model was designed taking into account the input-output pairs as specified in the previous section. The architecture is divided into three main components: one for processing the image input, another for handling the displacements input, and a final component for combining the outcomes from the previous components to estimate the sensor accuracy.

The component of the model responsible for the image input utilizes a CNN as its backbone, whose primary objective is to extract feature maps from the image input. For the displacement inputs, the model employs a Dense Network (i.e., a Neural Network) that processes the displacement by extracting relevant features and transforming the input into a format that can be integrated with the feature maps. After processing these inputs, the feature maps derived from the image are flattened into a one-dimensional vector, which is then concatenated with the features derived from the displacement input. This concatenated feature is fed into another Dense Network, which integrates the information and produces a single output value that represents the estimated sensor accuracy. The schematic of the model's architecture is shown in Figure 4.21.

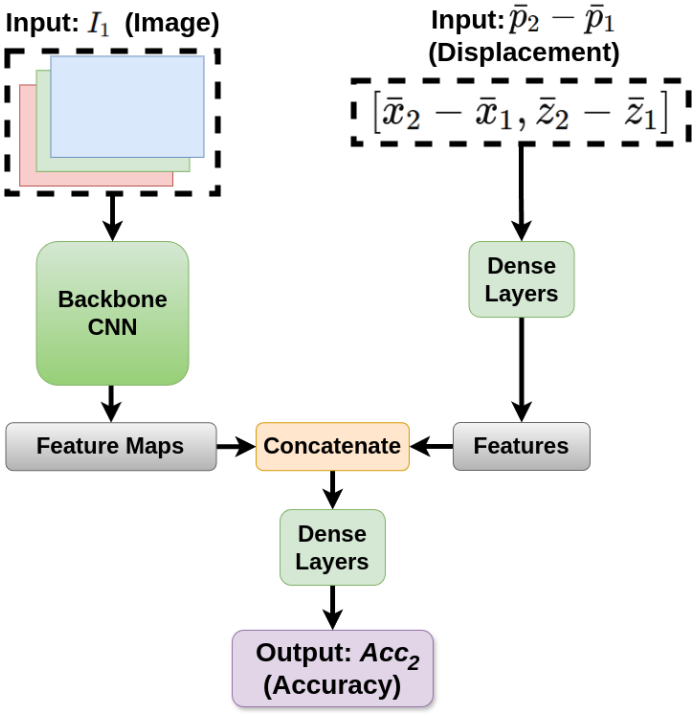

Figure 4.21: Schematic of the Accuracy Estimation Model Architecture

This custom model is trained end-to-end using the training data described in the previous section. It serves as the base of our improved solutions, which are presented in detail in the Sections 4.6.5 and 4.6.6.

4.6.4 Baseline: Multi-Step PBVS

As a baseline method, we use the PBVS approach described in Section 4.5. In order to address errors related to perspective changes and to align the sensor precisely at the center of the widest regions of the strawberries, we have implemented a tracking feature. This feature, essential for tracking a target strawberry among multiple strawberries in an image, uses Euclidean distance to compare the 3D coordinates of strawberries at two distinct moments, providing the PBVS system with updated positions of the strawberries after approaching them. The sensor accuracy achieved in this position is the baseline for performance comparisons of the improved methods. This process is applied in a loop, such that after processing each strawberry, the system moves on to the next strawberry until all the widest regions in a scene are addressed.

4.6.5 Improved 1: Grid-Based Solution (GBS)

Initially, we introduce an approach that leverages the previously introduced Accuracy Estimation Model, designed to position the sensor at the point of maximum predicted sensor accuracy. Unlike the Multi-Step PBVS that relies on image features, this proposed method focuses on function maximization through grid search, avoiding the dependence on specific image features. This approach processes batches, each consisting of pairs of a given image and multiple displacements derived from a pre-established grid. The approach considers that the image in a batch contains a strawberry or part of it, and predicts the sensor accuracy for all displacements within a batch, as shown in Figure 4.22.

The solution to finding the best position $[x^*, z^*]$ to move the end effector in order to achieve maximum reading accuracy of the sensor is determined by choosing the displacement with the highest predicted sensor accuracy $[\Delta x^*, \Delta z^*]$. Therefore, the robot will move to the position illustrated in Equation 4-3.

$$\begin{cases} x^* = x_{\text{current}} + \Delta x^*, \\ z^* = z_{\text{current}} + \Delta z^*, \end{cases} \tag{4-3}$$

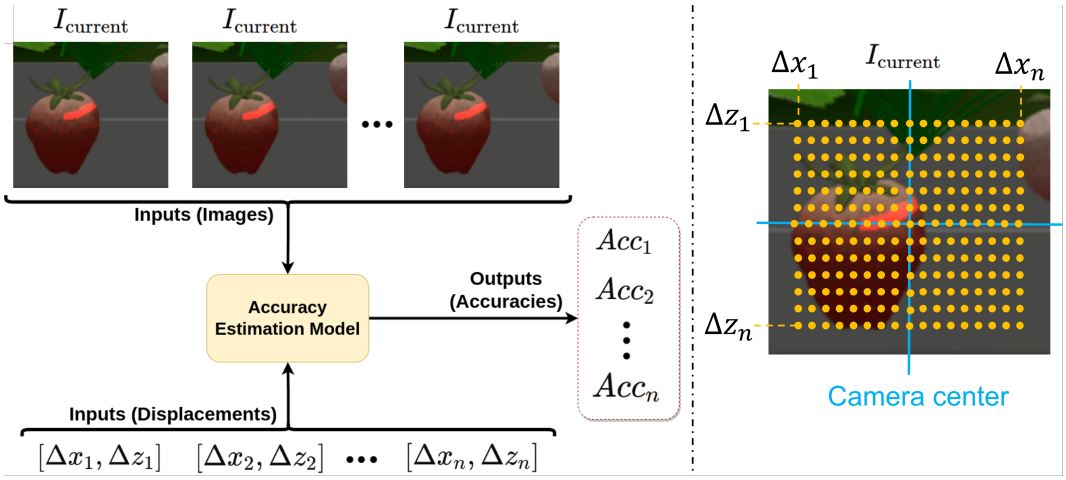

Figure 4.22: Illustration of the process for obtaining the sensor accuracies using a pre-established grid. On the left side of the figure, we present the accuracy estimation model receiving a batch of size n of displacements alongside corresponding duplicates of a single image to predict the sensor accuracy for each displacement. On the right side of the figure, we illustrate the pre-established grid of size $n \times n$ for the same image.

Visually, Figure 4.23 illustrates the process of updating the sensor's current position using the optimal coordinates $[x^*, z^*]$ returned by the model.

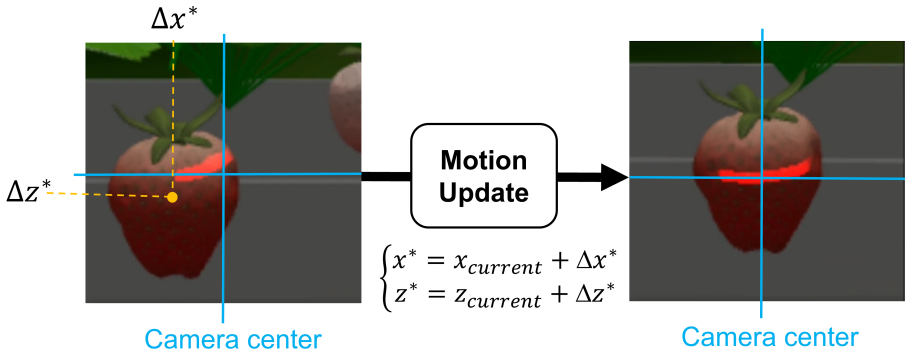

Figure 4.23: Illustration of applying the optimal displacement determined by the accuracy estimation model. The left image shows a strawberry with the sensor at the initial position for accuracy optimization. The optimal displacement $[\Delta x^*, \Delta z^*]$, as determined by the model, is applied to the sensor's current position. This adjustment results in a new position depicted in the right image, representing the predicted optimal location.

In practical situations, the PBVS approach described in Section 4.5 is used to position the sensor close to the fruit initially. Once the sensor is positioned near the fruit, the grid-based solution aids in guiding the robot's movement towards the position $[x^*, z^*]$. Upon reaching this position, the process is applied to the next strawberry, continuing in this manner until all strawberries in the scene have been addressed.

4.6.6 Improved 2: Network Gradient Visual Servoing (NGVS)

Our network gradient-based method leverages the trained Accuracy Estimation Model to compute the gradient of the accuracy output Acc with respect to the input displacement $[\Delta x, \Delta z]$, denoted as $\frac{\partial \text{Acc}}{\partial[\Delta x, \Delta z]}$. Unlike the GBS method, which maximizes a function using grid search, this approach employs direct iterative gradient computations. This computation follows the theory presented in the background section, but becomes considerably more complex to calculate for our dual-input architecture with feature fusion. In practice, however, there are direct solutions available using modern libraries to calculate gradients for specified outputs with respect to specified inputs.

One peculiar aspect of the Accuracy Estimation Model is that selecting the displacement input as $[0, 0]$ along with an image input I_1 theoretically predicts the sensor accuracy of the current image I_1 for the current camera position. A key component of our proposed method involves the calculation of the gradient at this zero displacement, which serves to identify the direction and magnitude of a displacement (i.e. an offset) that should be applied to the current camera position to increase the output value (i.e. sensor accuracy). Specifically, in Visual Servoing, the gradient's outcome is used to guide the control loop, enabling iterative adjustments to the end effector's positioning to maximize the spectrometer accuracy, as illustrated in Figure 4.24.

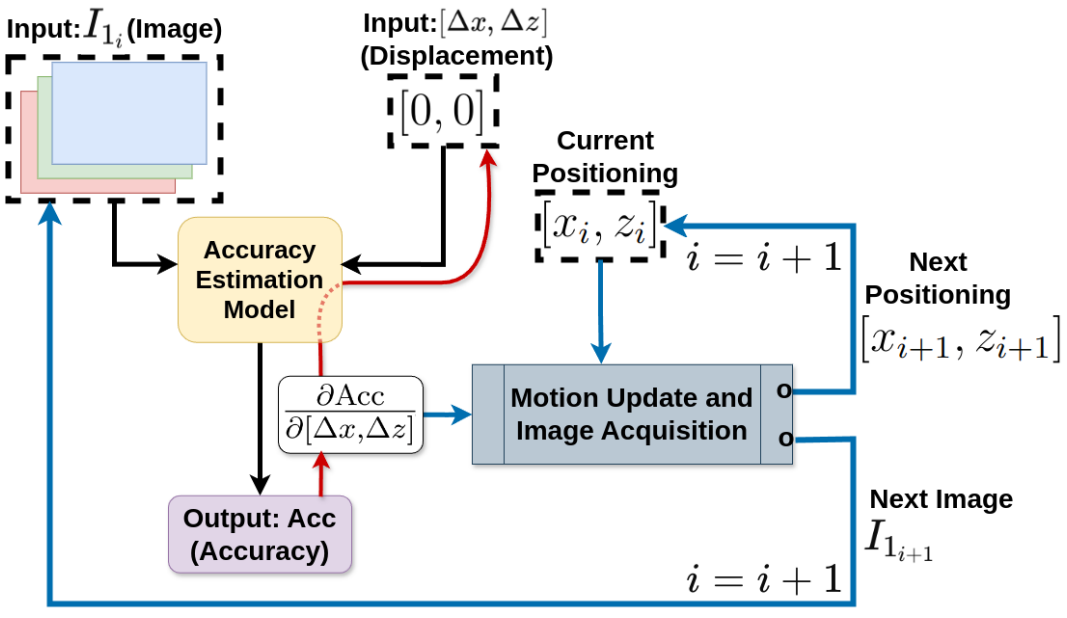

Figure 4.24: Illustration of our Network Gradient Visual Servoing model. The red arrow symbolizes the flow of gradient information through the Accuracy Estimation Model from the Accuracy output towards the Displacement input.

As illustrated in Figure 4.24, the 'Accuracy Estimation Model' receives

a constant displacement input of $[0, 0]$ and an image input I_1. It computes the gradient of the accuracy output Acc with respect to the displacement $[\Delta x, \Delta z]$, denoted as $\frac{\partial \text{Acc}}{\partial[\Delta x, \Delta z]}$. This gradient informs the 'Motion Update and Image Acquisition' block, which adjusts the end effector's positioning accordingly, and acquires a new image for the next cycle, ensuring continuous improvement of the sensor accuracy within the system. The motion update in a single step of the loop, which adjusts the current position based on the calculated gradient is formally defined by Equation 4-4.

$$\begin{cases} x_{\text{i+1}} = x_{\text{i}} + \dfrac{\partial \text{Acc}}{\partial \Delta x}, \\ z_{\text{i+1}} = z_{\text{i}} + \dfrac{\partial \text{Acc}}{\partial \Delta z}, \end{cases} \tag{4-4}$$

The control loop repeats until the magnitude of the gradient vector is smaller than a certain predefined threshold. Figure 4.25 visually exemplifies this process, showing an example where the loop terminates after achieving a gradient vector with a magnitude sufficiently small.

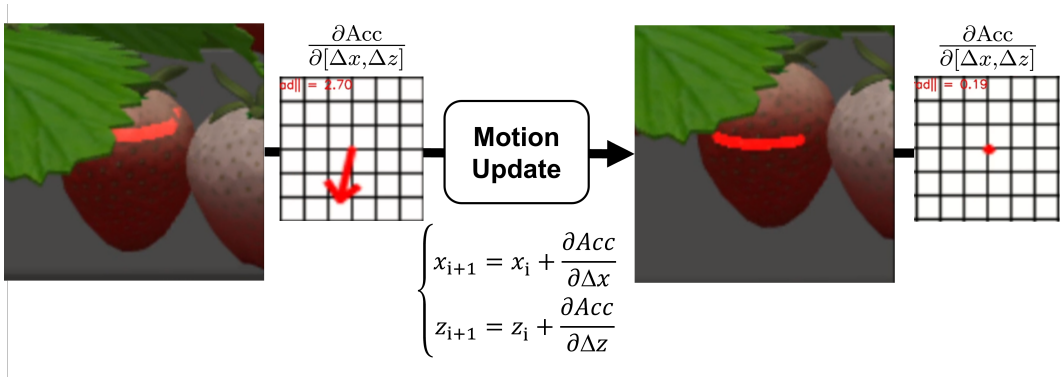

Figure 4.25: The figure illustrates two steps in a control loop employing the NGVS method. The initial gradient, calculated from the first image where the strawberry is partially occluded by a leaf, indicates the direction and magnitude for sensor movement. Following the gradient information, the sensor's position is adjusted, resulting in the second image. Subsequently, a second gradient is calculated, yielding a vector with a magnitude sufficiently small. This indicates an optimal positioning without occlusion, thereby triggering the stop mechanism in the control loop.

We assume that at the beggining of a first iteration the input image contains a complete or partial part of a strawberry. A pseudo-algorithm describing the accuracy optimization process using the gradient method is presented in Algorithm 3.

Algorithm 3: Pseudo-algorithm for the NGVS.

Result: Maximized spectrometer accuracy.

Initialize $i = 0$ and camera position $[x_i, z_i]$;

while $\left|\frac{\partial Acc}{\partial[\Delta x, \Delta z]}\right| > Threshold$ **do**

  Capture image I_{1_i};

  Compute gradient $\frac{\partial \text{Acc}}{\partial[\Delta x, \Delta z]}$ at $[0, 0]$;

  Apply the computed gradient to the current position to determine the new camera position $[x_{i+1}, z_{i+1}]$;

  Move end effector to new position and capture new image I_{i+1};

  $i \leftarrow i + 1$;

end

In practical situations, our NGVS method employs the PBVS approach described in Section 4.5 in order to approach the fruit. Once in proximity to the fruit, it transitions to the NGVS, positioning the sensor in order to maximize the sensor accuracy. This process is applied in a loop so that, following the maximization of the sensor accuracy for a strawberry, the system proceeds to the next strawberry. This continues until the sensor's accuracies for all strawberries in a scene are maximized.

It is important to emphasize that our model requires only an image to operate, enabling its application in real-time (30 FPS or 0.03s). In contrast, using sensor readings would result in a delay of 2 seconds per iteration.

5
Experimental Setup

The datasets discussed, previously presented in Table 3.1, were obtained using a real Intel®RealSense™ D435 camera. As for the datasets for simulation, detailed in Tables 3.1 and 4.2, they were generated through a simulation of the stereo camera, designed to closely mimic its FoV and Depth sensing capabilities. Specifically, the simulation of the 3D camera includes parameters that add noise to enhance the realism of the camera simulation, as shown in Table 5.1.

Table 5.1: Noise Parameters for the Simulated RealSense D435 in Gazebo.

Sensor Component	**Noise Mean**	**Standard Deviation**
Color Camera	0.000	0.007
IR Camera 1	0.000	0.050
IR Camera 2	0.000	0.050
Depth Camera	0.000	0.100

The virtual stereo camera, along with other components such as the Thorvald Slim, Mitsubishi RV-2AJ, Strawberry Generator, Laser Range Finders, Strip of Light Projector and Polytunnel, were simulated in Gazebo using ROS Noetic. The Strip of Light Projector was emulated by employing multiple ROS lamps using a Flashlight Plugin [1].

An NVIDIA®RTX™ A2000 Laptop GPU was used to handle the computational demands of inference, manage the training phases of the trainable models, and run all the simulation in real time. The instance segmentation and widest region detections' experiments were carried out by training the Fruit Detection algorithm and the Widest Region Detector as standalone models to optimize their individual performances. After training these models and obtaining the accuracy dataset through the use of both the best end-to-end pipeline (i.e., MaskRCNN-D and VGG-WSCNN) and the grid scanning technique, we proceeded to train the Accuracy Estimation Model separately.

The subsequent sections delve into the core experimental analysis, concentrating initially on two primary objectives: Fruit Instance Segmentation and

[1]Flashlight Plugin. Available: https://classic.gazebosim.org/tutorials?tut=flashlight [Accessed February 28, 2024]

Fruit Widest Region Detection. In sequence, we discuss the Accuracy Estimation Experiments followed by the Visual Servoing Experiments. The Visual Servoing Experiments specifically utilize the Baseline (Multi-Step PBVS), the Grid-Based Solution, and the Network Gradient Visual Servoing approaches, which are detailed earlier respectively in Sections 4.6.4, 4.6.5, and 4.6.6.

5.1 Fruit Instance Segmentation Experiments

For the task of fruit instance segmentation, we employed TensorFlow 2 (TF2). The model used was an adapted version of the MaskRCNN, originally developed by Waleed Abdulla [42] and subsequently updated for compatibility with TF2[2]. Key configurations applied to both segmentation approaches (i.e., MaskRCNN and MaskRCNN-D) include early stopping (ES) to prevent overfitting, an image resolution set to 1024×1024 pixels, and the use of ResNet101 as the backbone network for feature extraction. An important feature of R-CNN is Transfer Learning, which utilizes pre-trained weights from established datasets. Motivated by the enhanced efficiency and generalization capabilities it provides, we used weights from the SceneNet dataset [43]. In order to improve the convergence process, we used a strategy called exponential decay schedule for the learning rate. This technique methodically reduces the learning rate with each epoch, enabling swift initial learning and progressively finer adjustments to the model's weights as training advances. The Table 5.2 outlines some of the key training configurations used for both models.

Table 5.2: Training configurations for fruit instance segmentation experiments.

Configuration	**MaskRCNN**	**MaskRCNN-D**
Transfer Learning	SceneNet (RGB)	SceneNet (RGB-D)
Epochs (ES)	84	97
Learning Rate	$[10^{-3}, 10^{-4}]$	$[10^{-3}, 5 \times 10^{-4}]$

Another important tunable feature implemented in the model for strawberry detection was to discard all depth information beyond 30 cm from the camera. This adjustment is effective in the polytunnels where strawberries are known to be closer than this threshold, thus focusing the model's accuracy on the typical zone for strawberries.

[2]Mask-RCNN-TF2, 2022. Available: https://github.com/ahmedfgad/Mask-RCNN-TF2 [Accessed November 5, 2023]

5.2 Fruit Widest Region Detection Experiments

The ContourMax model employed for this task is a direct algorithm that operates without the need for hyperparameter tuning, making it straightforward to use. For the learning model, we adapted the VGG architecture within the PyTorch framework, replacing its conventional CNNs with WSCNNs [41]. The resulting model, VGG-WSCNN, consists of 16 trainable convolutional layers. The input images are resized to a resolution of 224×224 pixels with a square aspect ratio, corresponding to the largest dimension of the fruit's expanded bounding box. An early stopping (ES) criterion was also employed to prevent overfitting during training. The table 5.3 summarizes the training configurations for the VGG-WSCNN model. Transfer learning was not applied due to the architectural changes introduced by WSCNNs, making the original pre-trained weights not usable.

Table 5.3: Training configurations for the VGG-WSCNN model used in the widest region detection.

Configuration	**VGG-WSCNN**
Transfer Learning	None
Epochs (ES)	17
Learning Rate	10^{-4}

5.3 Accuracy Estimation Model Experiments

In the accuracy estimation experiments, the best end-to-end pipeline (i.e., MaskRCNN-D and VGG-WSCNN) was employed to identify the strawberries' widest regions. Additionally, the dataset previously presented in Table 4.2 was normalized for model training, following the configuration presented in Table 5.4.

Table 5.4: Normalization parameters for model training.

Data Type	**Normalization Range**
Image pixels	[0, 1]
Reward (Acc)	[0, 1]
Camera position ($\bar{p}$)	[0, 1]
Displacement ($\bar{p}_2 - \bar{p}_1$)	[-1, 1]

The details of our Accuracy Estimation Model's architecture are shown in Table 5.5.

Early stopping (ES) was used to prevent overfitting during training. Table 5.6 shows the training configuration for the Accuracy Estimation Model.

Table 5.5: Accuracy Estimation Model Architecture Details.

Layer	**Description**
Layers for Input 1	
Input Size (Image)	224 x 224 x 3
Backbone	VGG16 for backbone
Backbone Output	25088 features after Flatten operation
Layers for Input 2 (Displacement)	
Input Size (Displacement)	2 x 1
First Dense Layer	64 neurons
Second Dense Layer	128 neurons
Combined Layer	
Feature Combination	25216 total features
First Dense Layer	4096 neurons
Second Dense Layer	4096 neurons
Output Layer	1 neuron, sigmoid activation
Output Size (Accuracy)	1 x 1

Table 5.6: Training configurations for the Accuracy Estimation Model.

Configuration	**Acc. Est. Model**
Transfer Learning	None
Epochs (ES)	11
Learning Rate	10^{-4}

5.4 Visual Servoing Experiments

In the Visual Servoing experiments, we utilized the KDL library in Python for inverse kinematics calculations, which features an important capability to avoid singularities. For the Grid-Based Solution (GBS), we employed a set of 320 displacement, ranging from -3.2 cm to 3.2 cm in the x-axis and from -3.9 cm to 3.9 cm in the y-axis. This range slightly exceeds the dataset range presented earlier in Section 4.6.2, ensuring coverage and additional margin for variability that may occur during the experiments. For the Network Gradient Visual Servoing (NGVS), in order to compute the gradients of the accuracy output with respect to the displacement input of our Accuracy Estimation Model, we employed PyTorch's Autograd feature. It is important to mention that we do not perform denormalization on the gradient, instead we clip the result between -3 and 3 and use it to automatically determine both the direction of positioning (orientation from the gradient) and the step size for positioning (magnitude from the gradient).

We conducted two types of experiments, the first involving both real and simulated environments to determine the performance of the Multi-Step PBVS (Baseline) for reaching the desired widest region and achieving the distance

offset. In this phase, the baseline is evaluated on 200 real strawberries and 200 simulated strawberries, aiming to assess the performance of the best end-to-end pipeline when integrated into the manipulator. The second type of experiment focuses exclusively on simulation, comparing the Baseline with the GBS and the NGVS to determine which method provides superior and more consistent sensor accuracy. For this comparison, the methods are tested on 1000 unique virtual strawberries.

6
Results and Discussions

This chapter presents the outcomes of the Fruit Instance Segmentation, Fruit Widest Region Detection, Accuracy Estimation Model, and Visual Servoing, examining the performance of the algorithms through quantitative metrics and visual comparisons, based on test data. The results for these components are displayed in sequence in the following sections.

6.1
Fruit Instance Segmentation Results

Initially, we present the normalized confusion matrices for the two instance segmentation models, MaskRCNN and MaskRCNN-D. These matrices are derived from a confidence score of 0.8 for the detections, as referenced in [44]. The evaluation covers three test sets: NO2019, UK/NO2023, and Simulated, as shown in Figure 6.1. The confusion matrices indicate that MaskRCNN-D outperforms MaskRCNN for all the three test sets, exhibiting higher accuracy in classifying the categories of Ripe, Unripe, and Affected strawberries. This is evidenced by higher true positive rates along the diagonals and reduced misclassification rates in the off-diagonal elements of the matrices. It is important to note that unlike the other datasets, the 'Simulated' dataset does not have a separate 'Affected' category. Instead, predictions for 'Unripe' and 'Affected' strawberries were combined into a single 'Unripe' category.

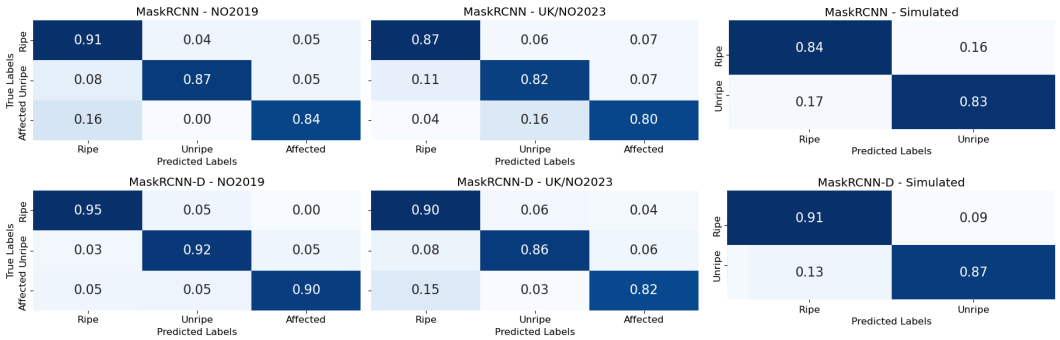

Figure 6.1: Normalized confusion matrices for MaskRCNN and MaskRCNN-D across the NO2019, UK/NO2023, and Simulated test sets.

To complement these findings and provide a more comprehensive assessment of model performance, we adopt a methodology similar to Ge et al. [4],

using the Average Precision (AP) for the two instance segmentation models. AP calculations are influenced by the Intersection over Union (IoU) threshold, which determines True Positives (TPs) and False Positives (FPs) detections. In line with the COCO benchmark standards [45] and mask scoring on MaskR-CNN [44], we consider detections with a confidence score of 0.8 and employ an IoU range threshold from 0.5 to 0.75 to calculate AP for each class, as visually explained in Section 2.1.3. The mean Average Precision (mAP) is then computed as the mean of these AP values across all classes, providing a single performance summary that accounts for various levels of detection difficulty. The equations used for these calculations are shown in Equation 6-1.

$$\begin{cases} \text{Precision} = \dfrac{\text{TPs}}{\text{TPs} + \text{FPs}}, \\ \text{Recall} = \dfrac{\text{TPs}}{\text{GTs}}, \\ \text{AP} = \displaystyle\int_0^1 p(r)\, dr. \end{cases} \tag{6-1}$$

where TPs denote True Positives, FPs denote False Positives, GTs refer to Ground Truths, and AP is the Average Precision which is equivalent to the area under the precision-recall curve, ranging from 0 to 1, with 1 being perfect detection performance. Here, $p(r)$ represents the precision as a function of recall r. Comparative tables that involve these metrics can be viewed in Table 6.1 for AP scores, and Table 6.2 for mAP scores.

Table 6.1: Comparison of AP scores for MaskRCNN and MaskRCNN-D on various test sets. The columns under 'Ripe', 'Unripe', and 'Affected' represent the detection classes for strawberries: 'Ripe' for strawberries in a ripe state, 'Unripe' for strawberries that are not yet ripe, and 'Affected' for strawberries affected by fungal or other diseases.

Test Set	**MaskRCNN**			**MaskRCNN-D**		
	Ripe	**Unripe**	**Affected**	**Ripe**	**Unripe**	**Affected**
NO2019	0.86	0.85	0.82	0.94	0.91	0.90
UK/NO2023	0.80	0.81	0.79	0.89	0.90	0.87
Simulated	0.82	0.79	-	0.94	0.89	-

Table 6.2: Comparison of mAP scores for MaskRCNN and MaskRCNN-D on various test sets.

Test Set	**MaskRCNN**	**MaskRCNN-D**
NO2019	0.84	0.92
UK/NO2023	0.80	0.89
Simulated	0.81	0.92

As detailed in Table 6.1, MaskRCNN-D, integrating depth information, consistently outperforms the baseline MaskRCNN across all test sets and

strawberry classifications (ripe, unripe, and affected). The performance improvement is especially pronounced in the simulated environment, likely due to the smoother depth information from the simulated stereo camera, offering a more idealized representation compared to real-world scenarios. Further analysis of the mAP scores in Table 6.2 supports these findings, showing significant enhancements with MaskRCNN-D: increases of 8.70% for NO2019, 10.83% for UK/NO2023, and 13.66% in the simulated environment. This highlights the depth integration's effectiveness in MaskRCNN-D, demonstrating its substantial impact across various testing conditions.

For a detailed visual examination of the MaskRCNN and MaskRCNN-D models, Figure 6.2 presents comparative examples of the models' performance on instance segmentation tasks. The integration of depth information in MaskRCNN-D not only improves the overall visual results but also significantly enhances the model's ability to distinguish between closely clustered strawberries, as evidenced by the examples in the figure.

6.2 Fruit Widest Region Results

In order to evaluate the performance of the fruit widest region detectors, we use the Root Mean Squared Error (RMSE) metric. The RMSE is derived from the Mean Squared Error (MSE), which calculates the average squared difference between the estimated and actual values as defined in Equation 6-2.

$$\begin{cases} \text{MSE} = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n}\left((X_{\text{r},i} - X_{\text{p},i})^2 + (Y_{\text{r},i} - Y_{\text{p},i})^2\right), \\ \text{RMSE}_{\text{pixels}} = N \times \sqrt{\text{MSE}}, \end{cases} \tag{6-2}$$

where $X_{\text{r},i}$ and $Y_{\text{r},i}$ are the labeled pixel coordinates, $X_{\text{p},i}$ and $Y_{\text{p},i}$ are the predicted pixel coordinates for the i-th data point. The variable n indicates the number of data points in a test set, and N represents the dimension of the square image, as both labels and outputs were normalized. The performance results for the standalone Widest Region Detector (derived from the ground truth segmentation) are presented in Table 6.3. Additionally, the combined results of the Fruit Instance Detector and the Widest Region Detector, showcasing the pipeline's effectiveness, can be found in Table 6.4.

Table 6.3: Comparison of RMSE scores for the standalone ContourMax and VGG-WSCNN on various test sets.

Test Set	**ContourMax**	**VGG-WSCNN**
NO2019	13.06	10.51
UK/NO2023	12.67	10.26
Simulated	12.47	10.21

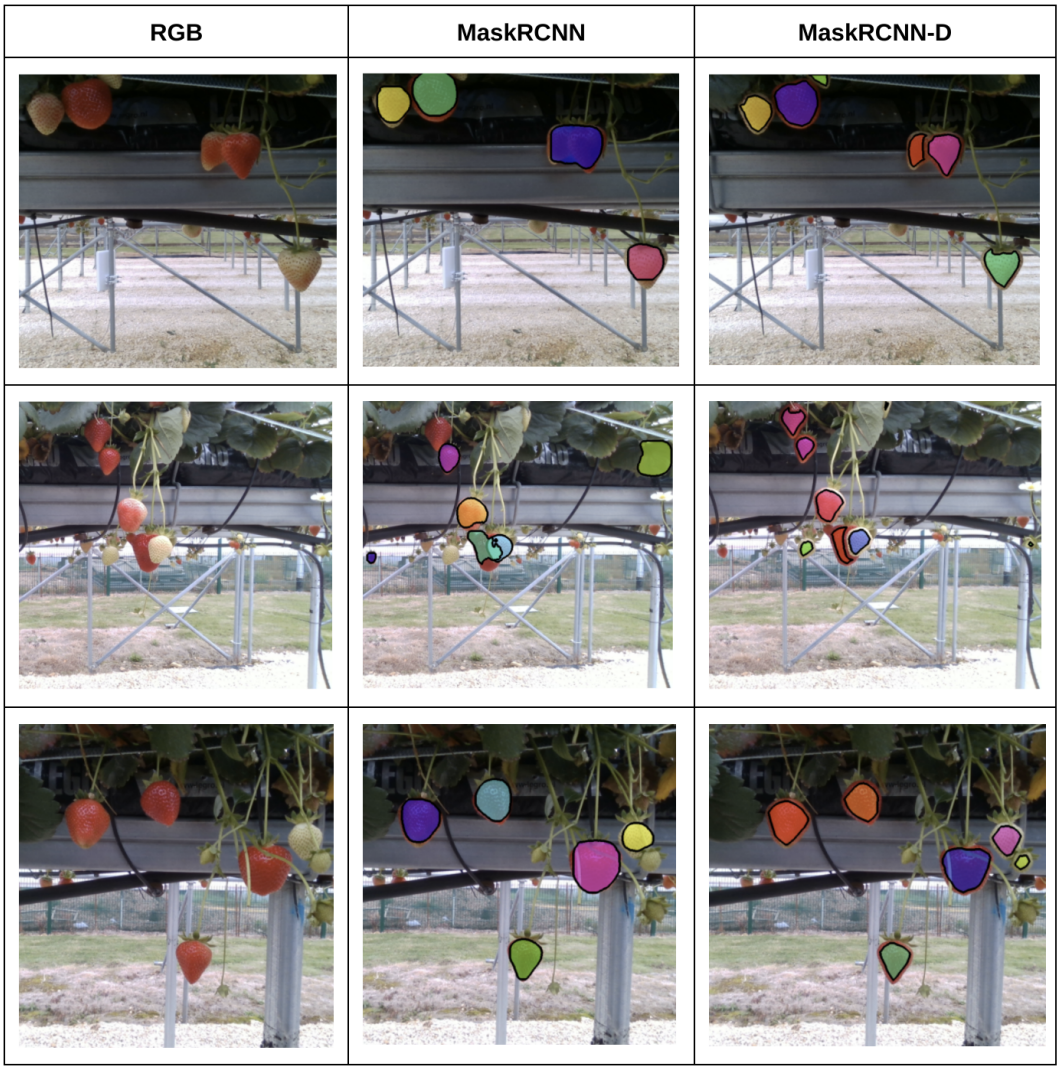

Figure 6.2: Visual examples of inputs (RGB) and outputs from MaskRCNN and MaskRCNN-D. Each row represents a different example. This figure illustrates the segmentation capability of the models, showing how detected instances are outlined.

Table 6.3 shows that VGG-WSCNN outperforms ContourMax in pinpointing the widest fruit region across various test sets. Table 6.4 further reveals that combining MaskRCNN-D with VGG-WSCNN (Pipeline 4) leads to the lowest RMSE scores, indicating that the end-to-end solution is the most effective configuration for precise widest region detection in both real and simulated environments. To highlight the differences reflected in the metric outcomes, Figure 6.3 showcases visual results from a pipeline approach, using the best-performing detector (MaskRCNN-D) for fruit instance detection. It assesses the capabilities of the two different Widest Region Detectors on both real and simulated datasets.

In the real-world dataset examples (R_i), it is evident that the learned approach for locating the widest region generally shows better results. This is

Table 6.4: Comparison of RMSE scores for various pipelines on different test sets. Pipe 1: MaskRCNN + ContourMax, Pipe 2: MaskRCNN + VGG-WSCNN, Pipe 3: MaskRCNN-D + ContourMax, Pipe 4: MaskRCNN-D + VGG-WSCNN.

Test Set	**Pipe1**	**Pipe2**	**Pipe3**	**Pipe4**
NO2019	24.06	19.74	14.76	12.62
UK/NO2023	26.43	21.82	15.56	13.35
Simulated	26.08	21.74	12.84	11.30

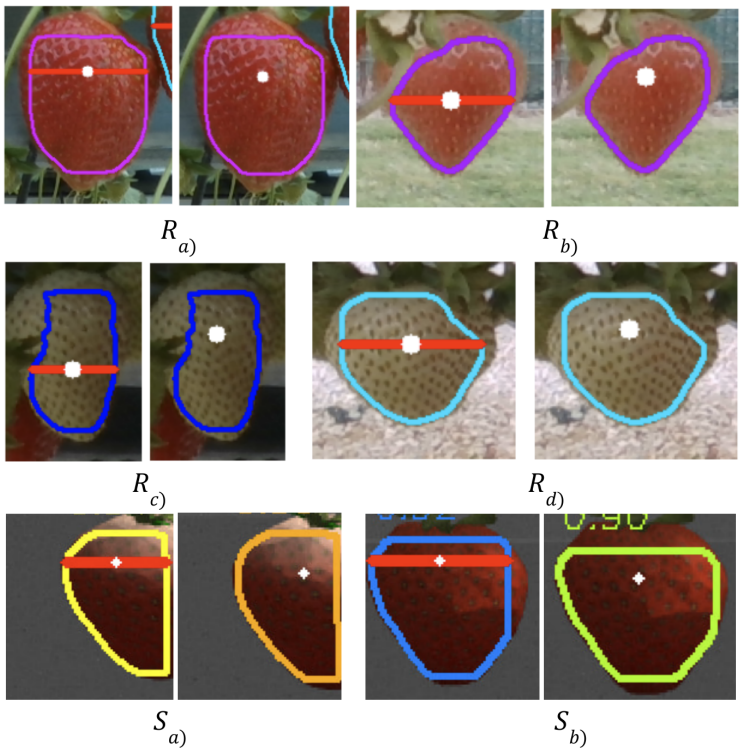

Figure 6.3: Comparative visualization of pipeline results using MaskRCNN-D and alternating the widest region detector, where each R_i corresponds to a pair of images from real-world data and each S_i to a pair from simulated data. These pairs contrast the detection outputs of ContourMax (left) versus VGG-WSCNN (right).

particularly noticeable in examples R_b and R_c, where the instance segmentation outcome was affected by leaf occlusion. Due to the ContourMax's reliance on segmentation accuracy, the widest region was not accurately identified in these cases. However, the learned model, which depends only on the expanded ROI from the segmented masks, provided satisfactory outcomes even in these challenging scenarios. When examining the simulated dataset examples, as in S_a it is noted that part of the strawberry was outside the camera's field of view. However, the learned approach still managed to identify a point for the widest region that was closer to the desired location. In the case of S_b, although the segmentation by the detector was not perfect, the learned method still

outperformed the contour-based approach.

6.3 Accuracy Estimation Model Results

The Accuracy Estimation Model is a regression model, thus we evaluate its performance using the Root Mean Squared Error (RMSE) metric as defined in Equation 6-3.

$$\begin{cases} \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Acc_{\text{r},i} - Acc_{\text{p},i})^2, \\ \text{RMSE}_{\text{accuracy}} = \sqrt{\text{MSE}}. \end{cases} \tag{6-3}$$

where $Acc_{\text{r},i}$ is the sensor accuracy ground truth and $Acc_{\text{p},i}$ is the predicted sensor accuracy for the i-th data point. The variable n indicates the number of data points in a test set. The performance results for the Accuracy Estimation Model are presented in Table 6.5. The sensor accuracies range from 0 to 1, corresponding to 0 and 100%.

Table 6.5: RMSE score for the Acc. Estimation Model on the Testing Dataset.

Test Set	**Accuracy Estimation Model**
Testing Data	0.104

The RMSE score of 0.104, as shown in Table 6.5, represents a 10.4% error margin. In our application, this level of error is acceptable, as our main solution (i.e., NGVS) does not directly use the sensor accuracy prediction, but rather the gradient of the model's accuracy output with respect to the displacement input.

For the next set of experiments, we delve into theoretical validations by visually examining the model's response to a zero displacement input, represented as (0,0). This specific input scenario implies no camera movement, thereby testing the model's ability to predict the sensor accuracy of the current image without any displacement. The outcomes of these tests for a singular strawberry instance in different positions are shown in Figure 6.4.

We can observe that the model was able to interpret a displacement input of (0,0) as an instruction to estimate the sensor accuracy of the present image, confirming the alignment between theoretical expectations and practical performance.

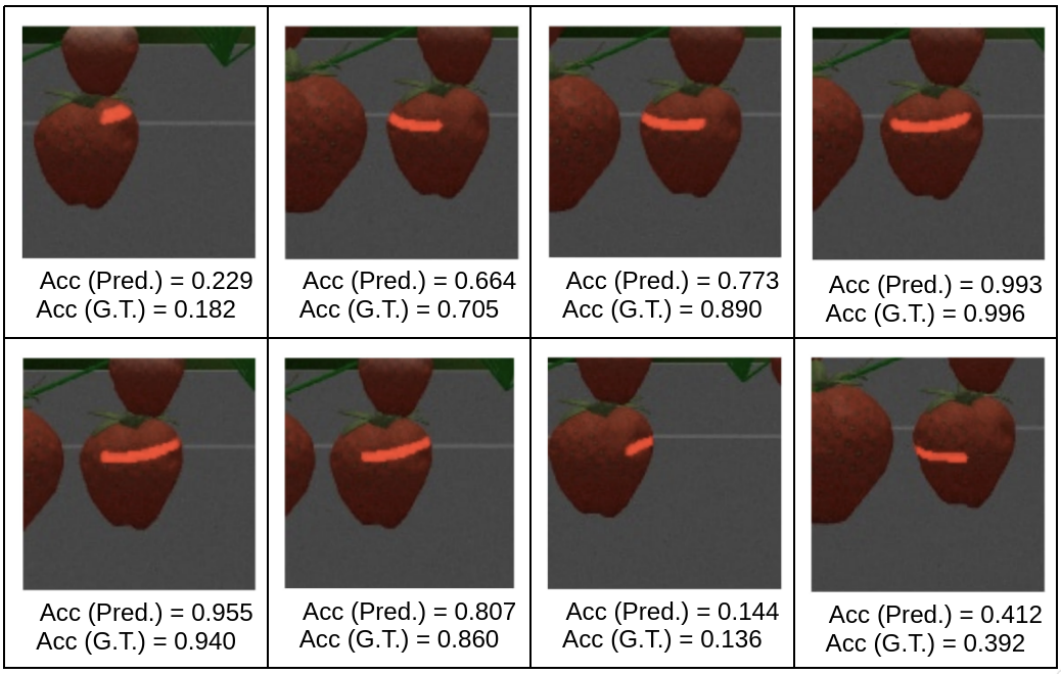

Figure 6.4: Evaluation of the Accuracy Estimation Model under zero displacement condition. In the figure, "Pred." represents the accuracy prediction from the model, and "G.T." denotes the ground truth accuracy.

6.4 Visual Servoing Results

Initially, we employ the Baseline, which utilizes the best end-to-end fruit detection pipeline (i.e., Pipeline 4), for reaching the desired widest region and achieving the specified distance offset from the sensor to the fruits. This evaluation is accomplished by calculating the minimum, maximum, average, and standard deviation of the achieved distance offsets for both the real and the simulated environments. The results are presented in Table 6.6.

Table 6.6: Performance of the Visual Servoing Baseline in achieving the specified distance offset.

Error Type	**Real (cm)**	**Simulated (cm)**
Minimum	-0.970	0.000
Maximum	0.820	0.001
Average	0.201	0.000
Std. Dev.	0.479	0.000

We can observe that the simulation results for reaching the distance offset are ideal, despite the inclusion of parameters that add noise to enhance the realism of the simulated environment previously presented in Table 5.1. According to our experimental observations, the point cloud noise stands out as the most significant factor being much noiser in real environments than

simulation. In our system, we mitigate this noise by implementing a technique where we read a 4×4 pixels grid in the point cloud around the desired point and calculate the average of the depths. Despite the noise problem, these results demonstrate that the fruit detection pipeline when integrated with the robotic arm performs satisfactorily for both simulation and real-world scenarios, meeting the distance offset specification (i.e., $\pm 1cm$).

In the sequence, we proceed to evaluate all the Visual Servoing methods applied to the 1000 unique virtual strawberries generated for this assessment. To illustrate this, Figure 6.5 presents the true normalized rewards (i.e., sensor accuracy) for these virtual strawberries using the Baseline, Grid-Based Solution (GBS), and Network Gradient Visual Servoing (NGVS), alongside displaying the Ground Truth (GT) for comparison. In order to obtain the GT, we used the grid scanning approach mentioned earlier to scan the various positions on each strawberry and returning the maximum sensor accuracy found.

The scatter plots illustrate that the NGVS method consistently achieves higher normalized rewards closer to the GT across all samples, outperforming both the Baseline and the GBS. Notably, the NGVS demonstrates a narrower clustering of data points, with fewer outliers, which visually implies more consistent performance. Next, we evaluate the methods through metrics of the mean, standard deviation, and the margin of error for the 95% confidence interval of the mean. The findings are presented in the Table 6.7, which presents a comparative analysis of the methods.

Table 6.7: Accuracy and consistency evaluation for the Visual Servoing methods.

Metric	**Baseline**	**GBS**	**NGVS**	**GT**
Mean	0.901	0.918	0.945	0.973
Standard Deviation	0.114	0.115	0.077	0.047
Margin of Error (95% CI)	0.007	0.007	0.005	0.003

From the table, we can observe that both the GBS and the NGVS outperformed the baseline, having better average sensor accuracy and consistency. The superior performances of GBS and NGVS are attributed to their use of the Accuracy Estimation Model, which was trained using a normalized reward function (i.e., sensor accuracy), differently from the baseline which follows the training on the widest region, without considering situations that might reduce the sensor accuracy. However, a limitation in GBS that affects the efficacy in optimizing the sensor positioning concerns its reliance on the sensor accuracy predictions from the Accuracy Estimation Model, which we previously demonstrated to have an RMSE of 10.4%. In contrast, the NGVS showcases superior

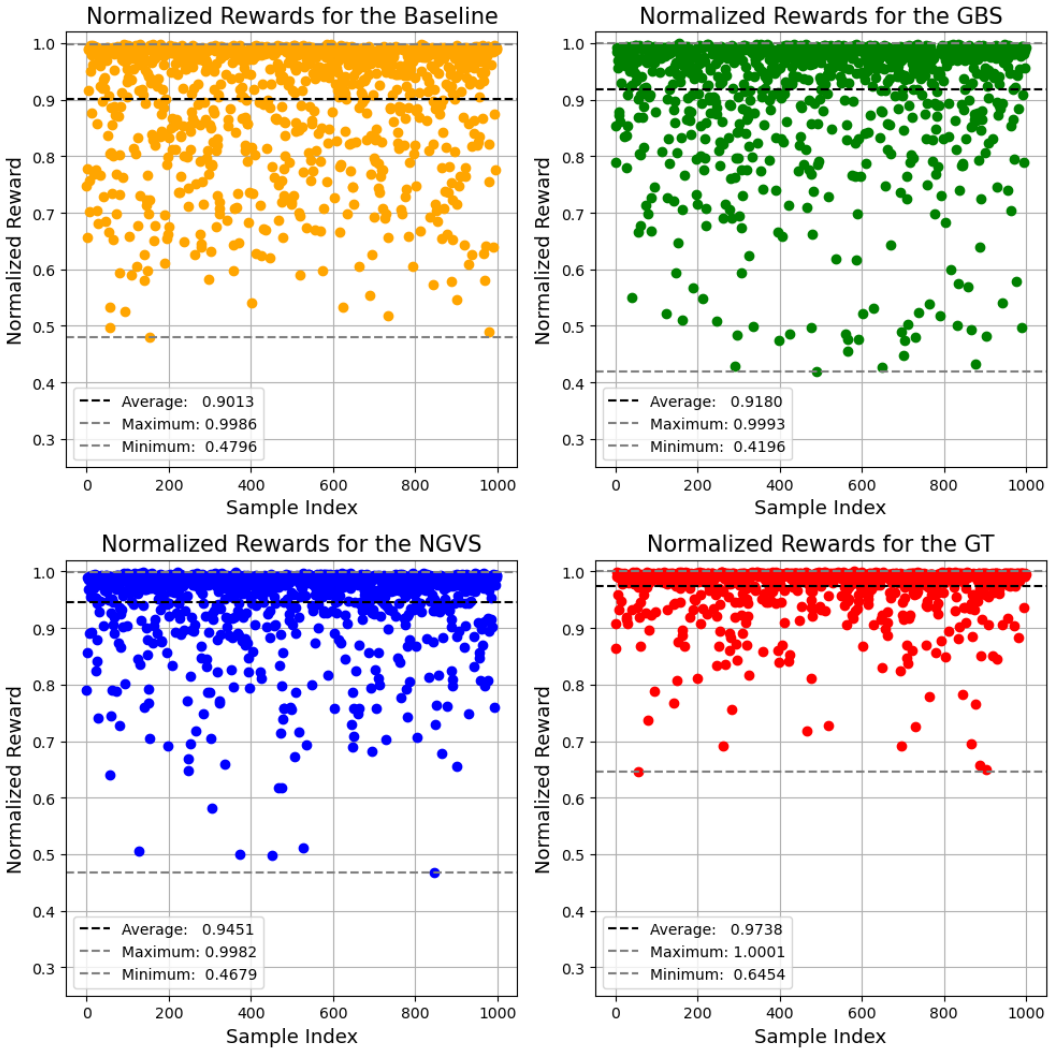

Figure 6.5: Comparison of the Normalized Reward Distributions x Samples for the Visual Servoing methods.

results by utilizing our novel network gradient-based approach in an iterative control, optimizing the sensor's position towards maximum sensor accuracy.

In order to make a comparison between the methods in pairs, we plotted two Kernel Density Estimate (KDE) graphs, chosen for their effectiveness in capturing the distribution of data points accurately. These graphs, comparing two distinct sets of comparative methods, are shown in Figure 6.6.

The upper graph presents the KDE of reward differences between NGVS and the Baseline method, while the lower graph similarly employs KDE for comparing NGVS against GBS, highlighting the distribution of their reward differences. In both graphs, the KDEs suggest a concentration of reward differences around the mean, indicating relatively small differences in performance between the methods. However, the tail on the right hand side of the zero line, indicating situations in which NGVS has better performance, is notably longer than the left hand side. This aligns with previous analyses,

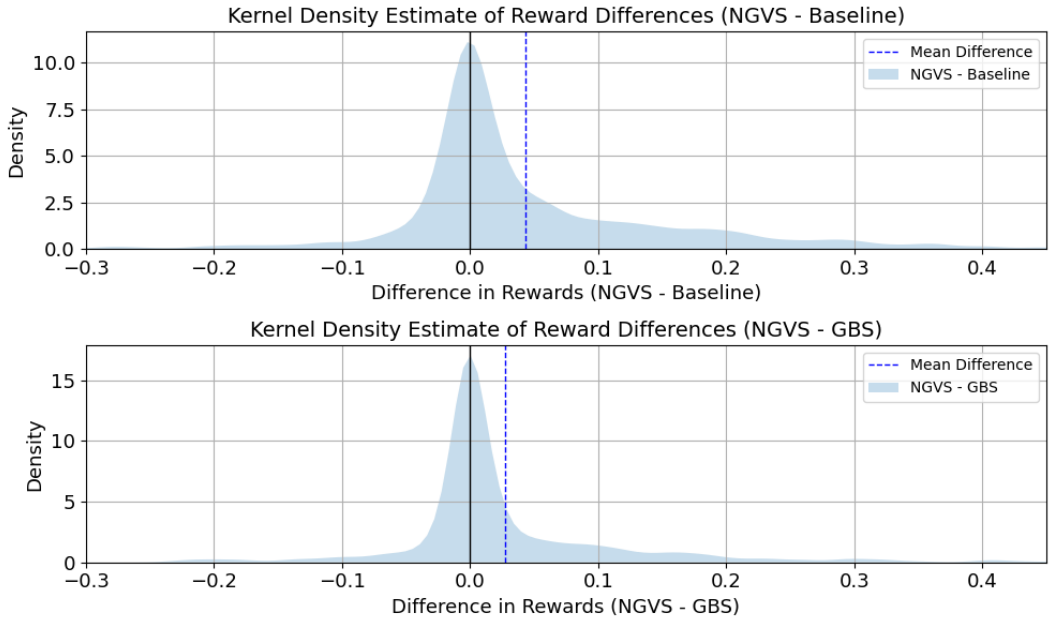

Figure 6.6: Kernel Density Estimate Comparative Analysis. Top image: comparison between NGVS and Baseline. Bottom image: comparison between NGVS and GBS.

supporting the conclusion that our novel approach provides more reliable and consistent outcomes.

6.5 Performance analysis

To visually illustrate the performance of the gradient method, including variations in orientation and magnitude across different scenarios, we present eight illustrative examples in Figure 6.7.

The illustration presents a series of cases where the gradient vectors suggest Visual Servoing signals in orientation and magnitude aimed at optimizing the accuracy of the sensor's readings on the fruit's surface in the simulated environment.

In the first case, the strip of light's position is offset leftward and downward from the strawberry, leading to the gradient to direct the sensor towards a region that would reposition the strip of light more centrally on the fruit's surface. A similar observation is made in the second case, yet here the adjustment is symmetrical to the first, with the gradient indicating a shift to the opposite side.

The third case reveals a strip of light displacement further left, with the gradient vector signaling a corrective action to the right. The fourth scenario presents a more pronounced gradient magnitude pointing upwards, a response to the strip of light and inspection point being critically offset downward,

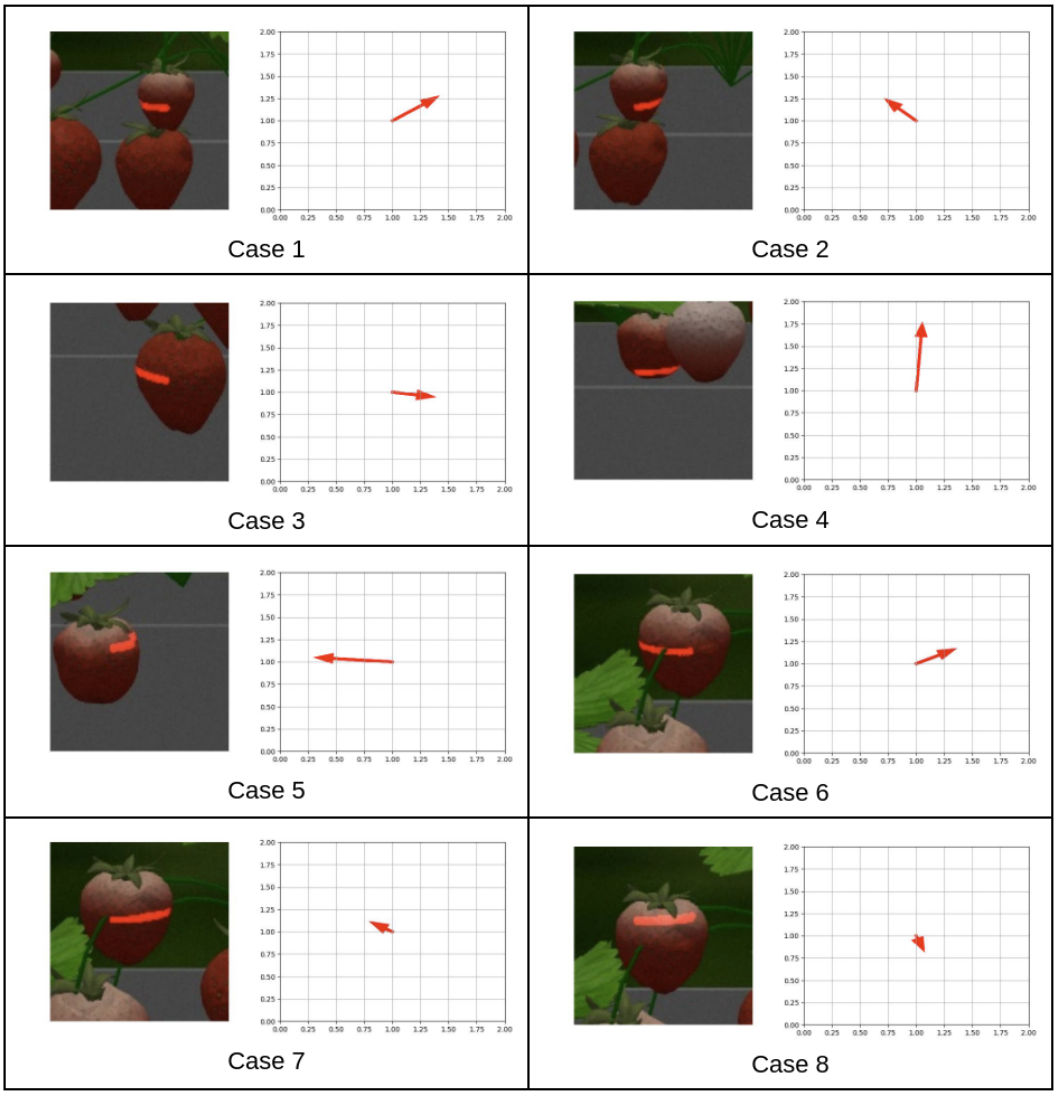

Figure 6.7: Eight examples of NGVS Performance in Various Scenarios. The red arrow on each case represents the orientation and magnitude in which the controller should follow in order to improve the sensor accuracy.

nearly missing the fruit. The fifth scenario mirrors the fourth, yet here the strip of light's misalignment is to the right, with the gradient magnitude significantly pointing to the left as a corrective measure to align the strip of light and inspection point precisely on the fruit's surface.

In the final three cases, occlusions are evident, with the gradient vectors indicating a direction for the manipulator to move the sensor away from the occlusion, suggesting doing so with small magnitude.

The visual examples support the findings, confirming the effectiveness of the gradient system in pointing to the position that increases the sensor accuracy.

In order to provide a deeper insight into the behavior of the gradient within our iterative control, employing the NGVS, we present the frames from

four complete trajectories in our real-time simulator. The first frame in each trajectory begins with the initial position estimated by the Pipeline 4 and then follows the gradient method in pursuit of maximizing the accuracy of the sensor's readings accuracy. It is important to note that the final position used is related to the position over the trajectory frames with the highest predicted sensor accuracy. The first trajectory of interest is illustrated in Figure 6.8.

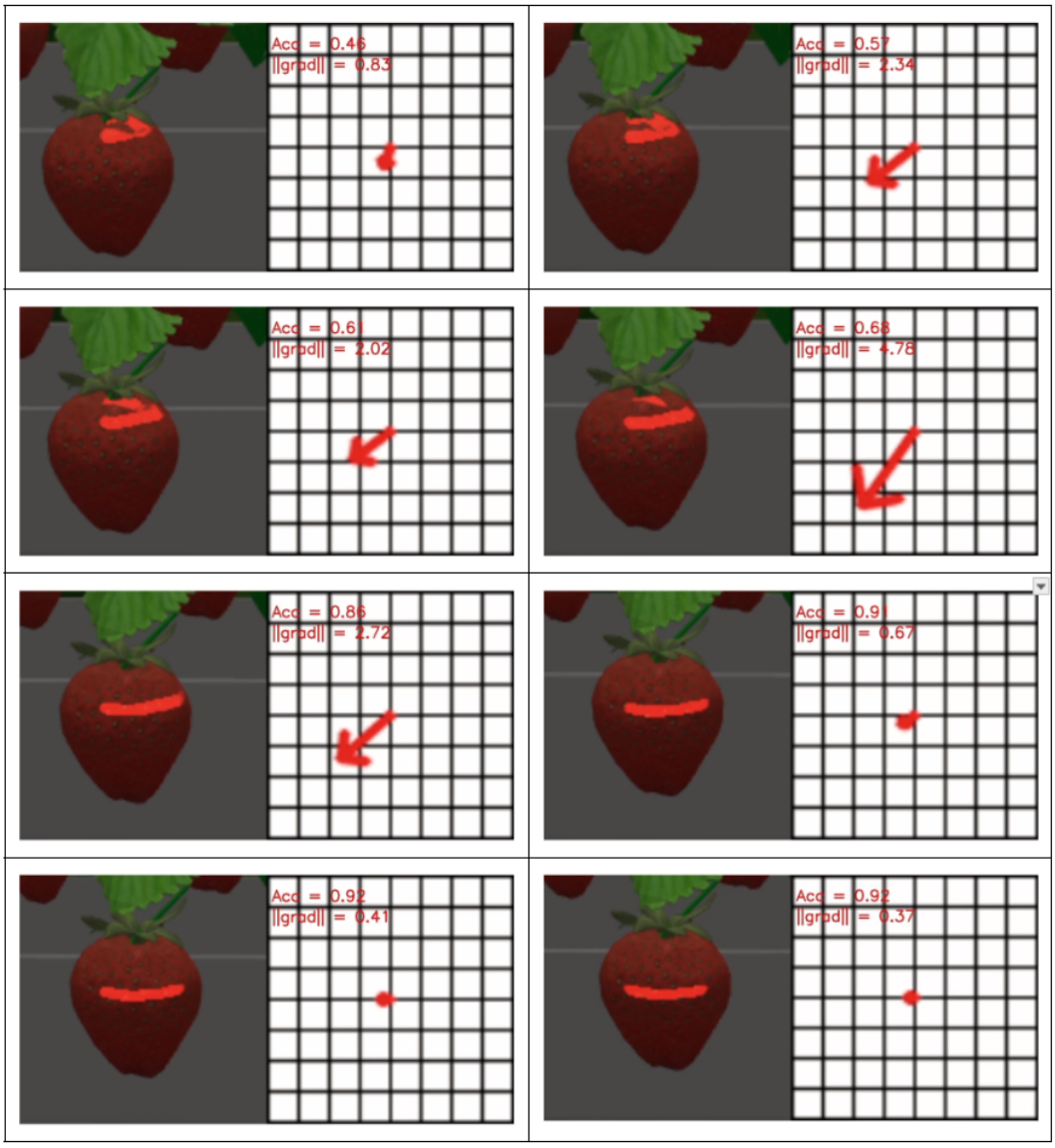

Figure 6.8: First trajectory of interest, consisting of eight frames. Each pair in the figure displays an input frame (image) and its respective gradient vector, provided by the NGVS.

In Figure 6.8, we observe in the first frame that the strip of light is completely misaligned with the strawberry. This occurs because the gradient method employed does not update the position of the widest region to correct for errors related to perspective changes as mentioned earlier. Consequently, the gradient directs towards the interior of the strawberry to enhance the

sensor's positioning. After the first position adjustment, the gradient becomes more precise in its direction, increasing in magnitude until a significant improvement in predicted sensor accuracy is observed in subsequent frames. At this moment, the positioning adjustment slows until it reaches the optimum point as determined by the gradient. Following this analysis, we present the second trajectory of interest in Figure 6.9.

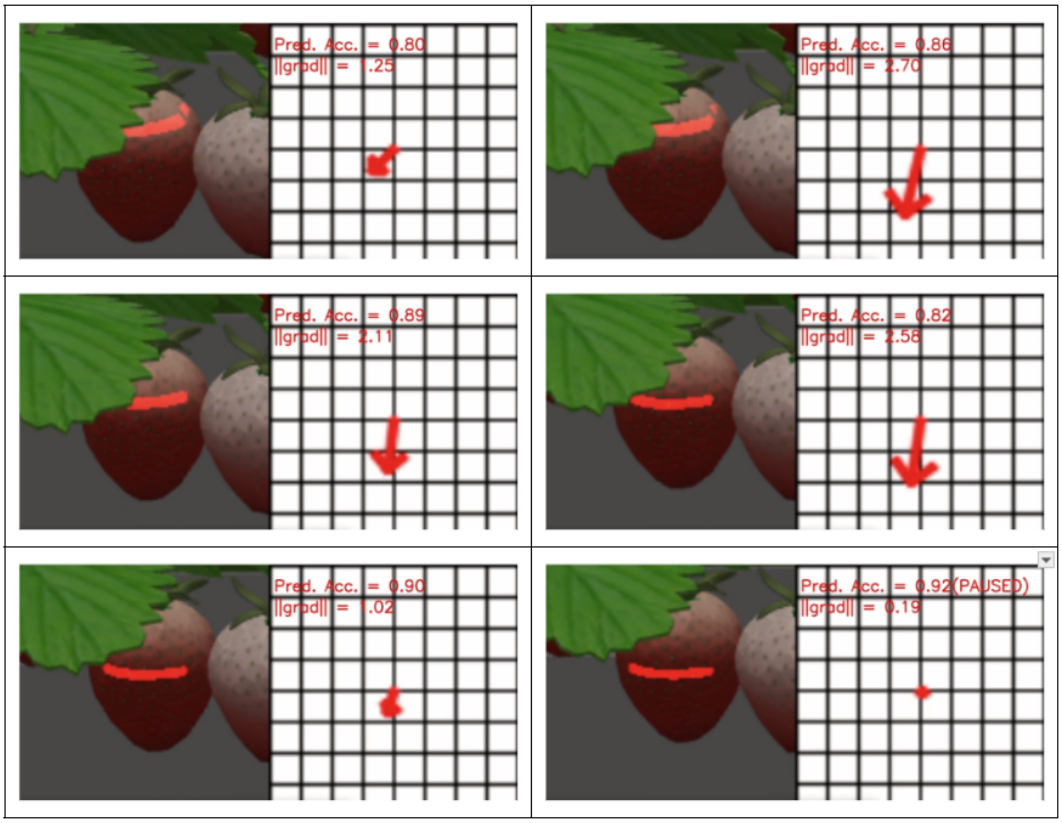

Figure 6.9: Second trajectory of interest, consisting of six frames. Each pair in the figure displays an input frame (image) and its respective gradient vector, provided by the NGVS.

In the trajectory illustrated in Figure 6.9, we observe that the gradient method directed the manipulator's control to incrementally optimize the sensor's reading accuracy, initially avoiding leaf occlusion, and finally achieving a position where both the strip of light and the inspection point were considered optimally. Following this analysis, we examine another trajectory example shown in Figure 6.10.

Figure 6.10 presents another example of sensor accuracy optimization through sensor positioning under occlusion. The method navigates the sensor out of the leaves' occlusion, ensuring that both the strip of light and the inspection point are positioned directly on the strawberry's surface. Subsequently, we proceed to analyze the results in a different scenario, as shown in Figure 6.11.

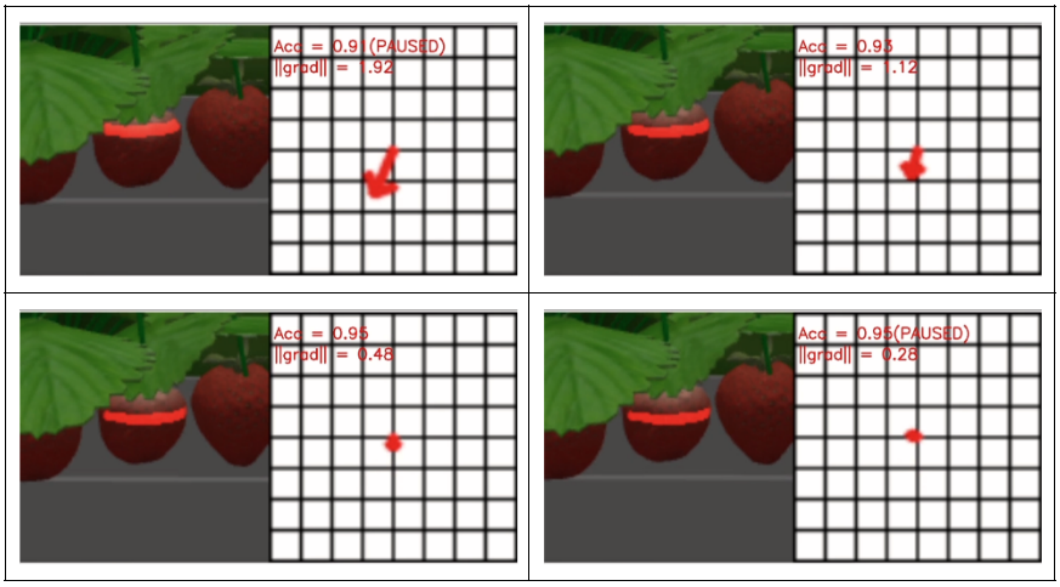

Figure 6.10: Third trajectory of interest, consisting of four frames. Each pair in the figure displays an input frame (image) and its respective gradient vector, provided by the NGVS.

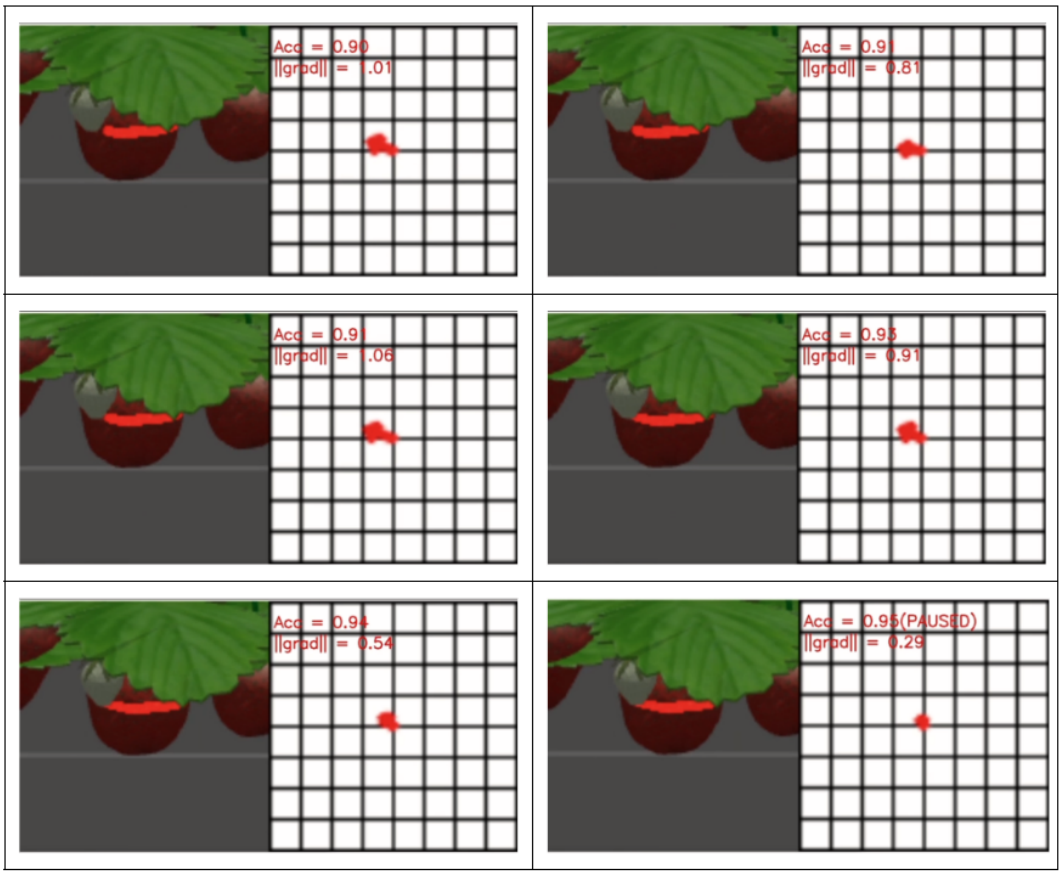

Figure 6.11: Fourth trajectory of interest, consisting of six frames. Each pair in the figure displays an input frame (image) and its respective gradient vector, provided by the NGVS.

In this final example shown in Figure 6.11, we observe a more extreme case of occlusion caused by leaves and a smaller, unripe strawberry. Regarding the gradient's behavior, we observe minor adjustments aimed at incrementally improving the sensor accuracy through positioning, keeping the strip of light more centralized on the fruit which avoided occlusion from the leaf on the right. However, ideally, the iterative gradient method should have also avoided the occlusion caused by the smaller strawberry. The table 6.8 provides a summary of the outcomes for each trajectory analyzed in our simulation.

Table 6.8: Summary of Outcomes for Each Trajectory in the Simulation

Trajectory	**Outcome**
#1	The gradient method directed toward the interior of the strawberry, significantly improving the sensor's positioning and accuracy until the optimal point was reached.
#2	Gradient method directed incremental adjustments to avoid leaf occlusion, gradually improving sensor accuracy until an optimal position was achieved.
#3	Optimization of sensor accuracy through positioning to avoid occlusion by leaves, ensuring both the strip of light and the inspection point are directly on the strawberry surface.
#4	Minor adjustments aimed at incrementally improving sensor accuracy through positioning, keeping the light strip more centralized on the fruit and avoiding occlusion from the leaf on the right, but ideally should have also avoided occlusion caused by the smaller strawberry.

6.6 Validation

For the final results in this work, we aim to demonstrate how our network gradient-based method, trained exclusively with simulated data, performs when faced with real-world images. We expect a marked improvement of these results when real images and spectral data become available for training in the future.

It is important to mention that we will not conduct an iterative analysis due to the unavailability of the necessary data. Instead, we will present sets of real-world scenarios, utilizing data from the same day as the study detailed in Section 3.1. In these examples, given the real-life conditions, the distance between the sensor and the strawberry may vary within the specified distance offset. For the first result, we present the outcome in Figure 6.12.

In Figure 6.12, we analyze the gradient's response to manually removing the stem and leaves from two distinct strawberries. We observe that, when

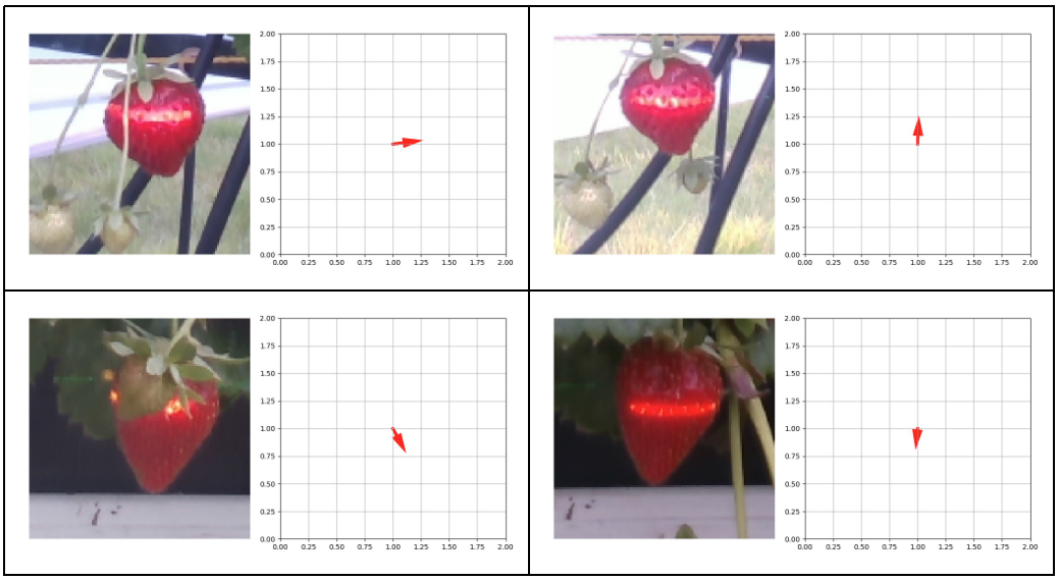

Figure 6.12: First set of examples of the NGVS Performance in real-world situations. The red arrow on each case represents the orientation and magnitude in which the controller should follow in order to improve the sensor accuracy.

the strip of light is obscured by occlusions (images on the left), the gradient directs away from the occlusion. In contrast, once the occlusions are manually cleared (images on the right), we note different behaviors: in the first case, the gradient points upwards to enhance the sensor's reading accuracy while in the second, the gradient is nearly zero, indicating a potential stopping point. Following this analysis, we present other results in Figure 6.13.

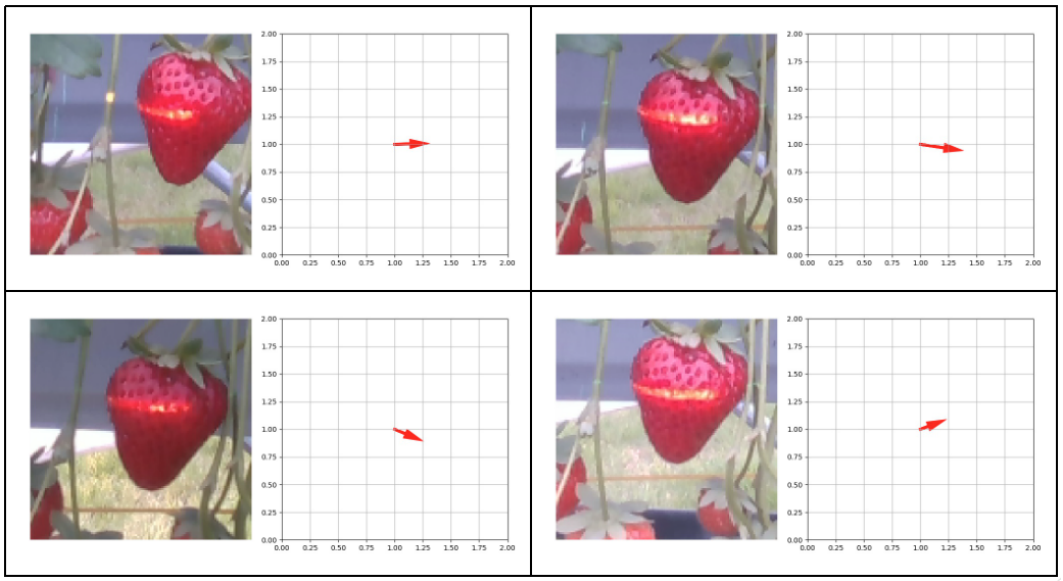

Figure 6.13: Second set of examples of the NGVS Performance in real-world situations. The red arrow on each case represents the orientation and magnitude in which the controller should follow in order to improve the sensor accuracy.

In Figure 6.13, we demonstrate the gradient's behavior at four different points on a specific strawberry. Our analysis reveals that the gradient consistently points towards the region of the strawberry that would allow the strip of light to cover more of the fruit's surface, thereby enhancing the accuracy of the sensor's readings. Following this discussion, we present other results in Figure 6.14.

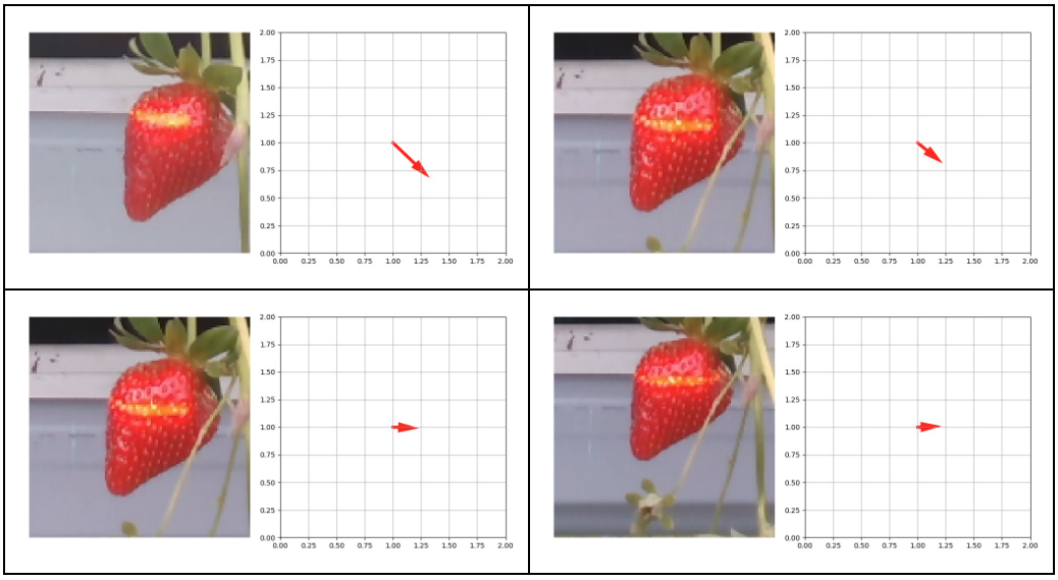

Figure 6.14: Third set of examples of the NGVS Performance in real-world situations. The red arrow on each case represents the orientation and magnitude in which the controller should follow in order to improve the sensor accuracy.

In Figure 6.14, we observe the gradient response to four different positions on a specific strawberry, which presents a challenging diagonal orientation. However, the gradient directions are reasonable and align with expectations for sensor accuracy optimization. Finally, we introduce other results in Figure 6.15.

In Figure 6.15, we analyze the gradient's behavior in four different situations on a specific strawberry. In the first two images, the gradient points with high magnitude outward from the strawberry's calyx, which would trivially enhance the accuracy of the sensor's readings. In contrast, the lower two images present scenarios where, depending on the stopping criteria, the sensor's positioning might oscillate from right to left. In practice, our algorithm is designed to prevent indefinite oscillations by detecting such behavior and then selecting the position of highest sensor accuracy within the oscillation period, which in this situation would be represented by the left image. The table 6.9 provides a summary of the outcomes for each set of examples analyzed under real-world conditions.

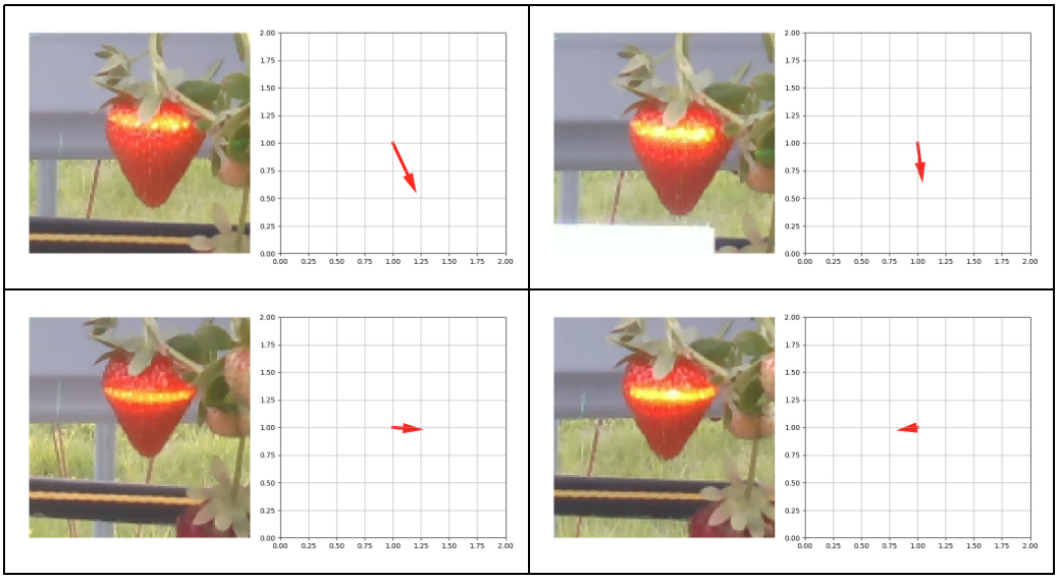

Figure 6.15: Fourth set of examples of the NGVS Performance in real-world situations. The red arrow on each case represents the orientation and magnitude in which the controller should follow in order to improve the sensor accuracy.

Table 6.9: Summary of Real-World Examples as Analyzed in the Study

Example Set	**Outcome**
#1	The gradient directs away from occlusions; after manual clearing, it points upwards to enhance sensor reading accuracy or remains nearly zero, indicating a stopping point.
#2	Demonstrates consistent behavior of the gradient pointing towards regions that enhance the coverage of the strip of light on the fruit, improving sensor accuracy.
#3	The gradient shows reasonable directions in response to a challenging diagonal orientation of the strawberry, aligning with expectations for sensor accuracy optimization.
#4	In different scenarios, the gradient either points significantly outward from the calyx or oscillates, with the algorithm designed to prevent indefinite oscillations by selecting the optimal position within these movements.

7
Conclusions

7.1
Fruit Instance Segmentation and Widest Region Detection

In this work, we demonstrated the value of depth-enhanced Deep Learning models in fruit detection. Our study shows that depth information significantly enhances fruit detection, with our results indicating up to a 13.7% improvement in mean Average Precision (mAP) when integrating depth data into the MaskRCNN-D model, suggesting that this approach should be more explored in the literature.

The learned approach for identifying the widest fruit region outperformed our direct algorithm, reducing the RMSE error by up to 20%, highlighting the potential of Deep Learning models in handling complex agricultural tasks.

The integration of our end-to-end pipeline, combining fruit detection with MaskRCNN-D and widest region identification using an enhanced VGG-16 model, resulted in a notably low localization error, achieving a root mean square error (RMSE) as low as 11.3 pixels on a 224×224 cropped strawberry image. This represents a 57% reduction in RMSE compared to the baseline pipeline. The end-to-end pipeline is particularly suited for real-time field applications, operating with two main components: widest region localization at 30 FPS, and fruit detection using MaskRCNN-D at 5 FPS. Furthermore, we demonstrated its effectiveness by integrating it with a robotic arm performing visual servoing.

7.2
Novel Network Gradient Visual Servoing Method

We successfully developed a novel Network Gradient Visual Servoing approach that leverages Deep Learning gradients to enhance the simulated accuracy of a spectral sensor. Our findings demonstrate that the network gradient theory can be effectively applied as a control signal for robotic positioning in simulated environments, and to the best of our knowledge, this approach has never been implemented or documented in the literature before, highlighting its novelty and potential impact. Additionally, the reward

function we developed was instrumental in helping the model learn how to avoid occlusions from leaves, stems, and other strawberries. Statistically, our method shows a significant improvement of approximately 5% over the baseline in terms of sensor accuracy in the simulated testing environment. Moreover, the reliability of the method is evidenced by a reduction in the standard deviation of about 33%, indicating more consistent performance. Furthermore, the iterative control aspect of the gradient method is capable of running at 30 FPS, making it suitable for real-time control applications.

The initial real-world results of applying the gradient method are promising, as they are outcomes without any fine-tuning on the model trained with data derived from simulation. This demonstrates the method's potential for effective real-world applications.

8
Future Work

8.1
Fruit Instance Segmentation and Widest Region Detection

Future work will involve testing the vision system's capabilities with an actual spectrometer designed to accurately estimate sugar content. This feature is crucial for automated harvesting, particularly for determining the optimal time for harvesting strawberries based on their ripeness level.

Additionally, planned developments will involve adapting MaskRCNN-D to include a specialized head for the widest region detection. This adaptation aims to reduce GPU usage and improve efficiency. The potential integration of other recent Neural Network architectures, such as the YOLO family [31], may also enhance the system's performance.

Integrating these models with embedded systems like NVIDIA®Jetson Nano™ is another important aspect of future research. This integration, along with the use of mobile manipulators, will facilitate practical field applications. Our goal is to employ autonomous agricultural robots, such as Thorvald [38], in conjunction with manipulators like UR-3e, to perform navigation and manipulation tasks based on the fruit detections.

Our simulator uses randomly generated strawberries. Enhancing the strawberry plant generation through the including of additional dimensions of variation, such as fungal infections and irregular shapes, is crucial. This will create an even more sophisticated simulated environment, essential for testing our methods.

8.2
Novel Network Gradient Visual Servoing Method

Upcoming activities will involve carrying out tests in real-world settings using our present Network Gradient Visual Servoing method, employing a developed strip of light projector. We will further propose new architectures for the Accuracy Estimation Model to improve the sensor accuracy prediction and consequently improve the gradient method. Following this, our objective is to derive sensor accuracy measurements directly from a real spectrometer and

proceed to retrain our model. Additionally, we plan to investigate other Visual Servoing methods and explore the potential requirement for implementing Reinforcement Learning with this new set of real spectroscopy data.

We aim to advance the development of the Network Gradient Visual Servoing controller to produce 3D vectors, enabling movement beyond the 2D plane. However, for this to function effectively, reducing point cloud noise in real environments by establishing a filter is an important prerequisite. We plan to adopt approaches similar to those discussed in the work of Ge et al. [4] to address this.

Autonomous synchronized navigation for the mobile manipulator is a key objective. To achieve this, we plan to integrate navigation sensors, such as 3D LiDAR and GPS, and apply the concept of Topological Navigation [46], aiming at achieving full autonomy within polytunnels.

9
Publications during the Doctorate

During the doctoral journey, a total of five papers have been published, with an additional paper currently being prepared for journal submission. Below is a list of these publications.

- **Depth-Enhanced 3D Deep Learning for Strawberry Detection and Widest Region Identification in Polytunnels** [11]. Lins Tenorio, G., Khaksar, W., & Caarls, W. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART2024)*, February 2024.

- **Automatic visual estimation of tomato cluster maturity in plant rows** [5]. Lins Tenorio, G., & Caarls, W. In *Machine Vision and Applications (MVAP)*, May 2021. Qualis A3.

- **Fruit Localization and Environment Perception for Strawberry Harvesting Robots** [4]. Ge, Y., Xiong, Y., Lins Tenorio, G., & From, P. J. In *IEEE Access*, October 2019. Qualis A3.

- **Comparative Study of Computer Vision Models for Insect Pest Identification in Complex Backgrounds** [47]. Lins Tenorio, G., Martins, F. F., Carvalho, T. M., Leite, A. C., Figueiredo, K., Vellasco, M., & Caarls, W. In *Proceedings of the 12th International Conference on Developments in eSystems Engineering (DeSE2019)*, October 2019.

- **Comparação de modelos de Machine Learning aplicados a previsão de casos totais de Dengue** [48]. Carvalho, T., Lins Tenorio, G., Vellasco, M., Figueiredo, K., & Caarls, W. In *Proceedings of the XVI Encontro Nacional de Inteligência Artificial e Computacional*, October 2019.

Bibliography

[1] ILYAS, T.; UMRAIZ, M.; KHAN, A. ; KIM, H.. **Dam: Hierarchical adaptive feature selection using convolution encoder decoder network for strawberry segmentation**. Frontiers in plant science, 12:591333, 2021.

[2] LEE, S.; ARORA, A. S. ; YUN, C. M.. **Detecting strawberry diseases and pest infections in the very early stage with an ensemble deep-learning model**. Frontiers in Plant Science, 13:991134, 2022.

[3] REN, G.; WU, H.; BAO, A.; LIN, T.; TING, K.-C. ; YING, Y.. **Mobile robotics platform for strawberry temporal–spatial yield monitoring within precision indoor farming systems**. Frontiers in Plant Science, 14:1162435, 2023.

[4] GE, Y.; XIONG, Y.; TENORIO, G. L. ; FROM, P. J.. **Fruit localization and environment perception for strawberry harvesting robots**. IEEE Access, 7:147642–147652, 2019.

[5] LINS TENORIO, G.; CAARLS, W.. **Automatic visual estimation of tomato cluster maturity in plant rows**. Machine Vision and Applications, 32(4):78, 2021.

[6] HARINI, S.; DESHPANDE, P.; DUTTA, J. ; RAI, B.. **A deep learning-based fruit quality assessment system**. In: INTERNATIONAL CONFERENCE ON WATER ENERGY FOOD AND SUSTAINABILITY, p. 187–192. Springer, 2021.

[7] WOLD, J. P.; O'FARRELL, M.; ANDERSEN, P. V. ; TSCHUDI, J.. **Optimization of instrument design for in-line monitoring of dry matter content in single potatoes by nir interaction spectroscopy**. Foods, 10(4):828, 2021.

[8] WOLD, J. P.; ANDERSEN, P. V.; AABY, K.; REMBERG, S. F.; HANSEN, A.; O'FARRELL, M. ; TSCHUDI, J.. **Inter seasonal validation of non-contact nir spectroscopy for measurement of total soluble solids in high tunnel strawberries**. Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy, 309:123853, 2024.

[9] LE LOUËDEC, J.; CIELNIAK, G.. **3d shape sensing and deep learning-based segmentation of strawberries**. Computers and Electronics in Agriculture, 190:106374, 2021.

[10] CALLI, B.; CAARLS, W.; JONKER, P. ; WISSE, M.. **Comparison of extremum seeking control algorithms for robotic applications**. In: 2012 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, p. 3195–3202. IEEE, 2012.

[11] TENORIO, G. L.; KHAKSAR, W. ; CAARLS, W.. **Depth-enhanced 3d deep learning for strawberry detection and widest region identification in polytunnels**. In: PROCEEDINGS OF THE 16TH INTERNATIONAL CONFERENCE ON AGENTS AND ARTIFICIAL INTELLIGENCE - VOLUME 2: ICAART, p. 471–481. INSTICC, SciTePress, 2024.

[12] GOODFELLOW, I.; BENGIO, Y. ; COURVILLE, A.. **Deep Learning**. MIT Press, 2016. `http://www.deeplearningbook.org`.

[13] LECUN, Y.; BENGIO, Y. ; HINTON, G.. **Deep learning**. nature, 521(7553):436, 2015.

[14] MICHELUCCI, U.. **Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks**. Apress, 2018.

[15] Chesi, G.; Hashimoto, K., editors. **Visual Servoing via Advanced Numerical Methods**, volumen 401 de **Lecture Notes in Control and Information Sciences**. Springer-Verlag London, 2010.

[16] BELHUMEUR, P. N.; HESPANHA, J. P. ; KRIEGMAN, D. J.. **Eigenfaces vs. fisherfaces: Recognition using class specific linear projection**. IEEE Transactions on pattern analysis and machine intelligence, 19(7):711–720, 1997.

[17] VIOLA, P.; JONES, M.. **Rapid object detection using a boosted cascade of simple features**. In: PROCEEDINGS OF THE 2001 IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION. CVPR 2001, volumen 1, p. I–I. Ieee, 2001.

[18] LOWE, D. G.. **Distinctive image features from scale-invariant keypoints**. International journal of computer vision, 60:91–110, 2004.

[19] DALAL, N.; TRIGGS, B.. **Histograms of oriented gradients for human detection**. In: 2005 IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR'05), volumen 1, p. 886–893. Ieee, 2005.

[20] NICKOLLS, J.; KIRK, D.. **Graphics and computing gpus**. Computer Organization and Design: The Hardware/Software Interface, DA Patterson and JL Hennessy, 4th ed., Morgan Kaufmann, p. A2–A77, 2009.

[21] GU, J.; WANG, Z.; KUEN, J.; MA, L.; SHAHROUDY, A.; SHUAI, B.; LIU, T.; WANG, X.; WANG, G.; CAI, J. ; OTHERS. **Recent advances in convolutional neural networks**. Pattern Recognition, 77:354–377, 2018.

[22] KRIZHEVSKY, A.; SUTSKEVER, I. ; HINTON, G. E.. **Imagenet classification with deep convolutional neural networks**. Advances in neural information processing systems, 25, 2012.

[23] SIMONYAN, K.; ZISSERMAN, A.. **Very deep convolutional networks for large-scale image recognition**. arXiv preprint arXiv:1409.1556, 2014.

[24] HE, K.; ZHANG, X.; REN, S. ; SUN, J.. **Deep residual learning for image recognition**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 770–778, 2016.

[25] TAN, M.; LE, Q.. **Efficientnet: Rethinking model scaling for convolutional neural networks**. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, p. 6105–6114. PMLR, 2019.

[26] LONG, J.; SHELHAMER, E. ; DARRELL, T.. **Fully convolutional networks for semantic segmentation**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 3431–3440, 2015.

[27] RONNEBERGER, O.; FISCHER, P. ; BROX, T.. **U-net: Convolutional networks for biomedical image segmentation**. In: MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION–MICCAI 2015: 18TH INTERNATIONAL CONFERENCE, MUNICH, GERMANY, OCTOBER 5-9, 2015, PROCEEDINGS, PART III 18, p. 234–241. Springer, 2015.

[28] BADRINARAYANAN, V.; KENDALL, A. ; CIPOLLA, R.. **Segnet: A deep convolutional encoder-decoder architecture for image segmentation**. IEEE transactions on pattern analysis and machine intelligence, 39(12):2481–2495, 2017.

[29] HE, K.; GKIOXARI, G.; DOLLÁR, P. ; GIRSHICK, R.. **Mask r-cnn**. In: PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, p. 2961–2969, 2017.

[30] BOLYA, D.; ZHOU, C.; XIAO, F. ; LEE, Y. J.. **Yolact: Real-time instance segmentation**. In: PROCEEDINGS OF THE IEEE/CVF INTERNATIONAL CONFERENCE ON COMPUTER VISION, p. 9157–9166, 2019.

[31] REIS, D.; KUPEC, J.; HONG, J. ; DAOUDI, A.. **Real-time flying object detection with yolov8**. arXiv preprint arXiv:2305.09972, 2023.

[32] HÄNI, N.; ISLER, V.. **Visual servoing in orchard settings**. In: 2016 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), p. 2946–2953. IEEE, 2016.

[33] SHI, Y.; ZHANG, W.; LI, Z.; WANG, Y.; LIU, L. ; CUI, Y.. **A "global–local" visual servo system for picking manipulators**. Sensors, 20(12):3366, 2020.

[34] GAMBA, J. D.; FROM, P. J. ; LEITE, A. C.. **A visual servoing approach for robotic fruit harvesting in the presence of parametric uncertainties**. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA-CBA, volumen 1, 2019.

[35] CANNY, J.. **A computational approach to edge detection**. IEEE Transactions on pattern analysis and machine intelligence, (6):679–698, 1986.

[36] KIRYATI, N.; ELDAR, Y. ; BRUCKSTEIN, A. M.. **A probabilistic hough transform**. Pattern recognition, 24(4):303–316, 1991.

[37] RUSSELL, B. C.; TORRALBA, A.; MURPHY, K. P. ; FREEMAN, W. T.. **Labelme: a database and web-based tool for image annotation**. International journal of computer vision, 77:157–173, 2008.

[38] GRIMSTAD, L.; FROM, P. J.. **The thorvald ii agricultural robotic system**. Robotics, 6(4):24, 2017.

[39] SATHER, J.. **Viewpoint Optimization for Autonomous Strawberry Harvesting with Deep Reinforcement Learning**. PhD thesis, California Polytechnic State University, 2019.

[40] BROCK, A.; DE, S.; SMITH, S. L. ; SIMONYAN, K.. **High-performance large-scale image recognition without normalization**. arXiv preprint arXiv:, 2021.

[41] BALLOLI, V.. **A pytorch implementation of nfnets and adaptive gradient clipping**. `https://github.com/vballoli/nfnets-pytorch`, 2021.

[42] ABDULLA, W.. **Mask r-cnn for object detection and instance segmentation on keras and tensorflow**. `https://github.com/matterport/Mask_RCNN`, 2017.

[43] HANDA, A.; PATRAUCEAN, V.; BADRINARAYANAN, V.; STENT, S. ; CIPOLLA, R.. **Scenenet: understanding real world indoor scenes with synthetic data. arxiv preprint (2015)**. arXiv preprint arXiv:1511.07041, 2015.

[44] HUANG, Z.; HUANG, L.; GONG, Y.; HUANG, C. ; WANG, X.. **Mask scoring r-cnn**. In: PROCEEDINGS OF THE IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 6409–6418, 2019.

[45] LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P. ; ZITNICK, C. L.. **Microsoft coco: Common objects in context**. In: COMPUTER VISION–ECCV 2014: 13TH EUROPEAN CONFERENCE, ZURICH, SWITZERLAND, SEPTEMBER 6-12, 2014, PROCEEDINGS, PART V 13, p. 740–755. Springer, 2014.

[46] MENG, X.; RATLIFF, N.; XIANG, Y. ; FOX, D.. **Scaling local control to large-scale topological navigation**. In: 2020 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), p. 672–678. IEEE, 2020.

[47] TENÓRIO, G. L.; MARTINS, F. F.; CARVALHO, T. M.; LEITE, A. C.; FIGUEIREDO, K.; VELLASCO, M. ; CAARLS, W.. **Comparative study of computer vision models for insect pest identification in complex backgrounds**. In: 2019 12TH INTERNATIONAL CONFERENCE ON DEVELOPMENTS IN ESYSTEMS ENGINEERING (DESE), p. 551–556. IEEE, 2019.

[48] CARVALHO, T.; TENÓRIO, G.; FIGUEIREDO, K.; VELLASCO, M. ; CAARLS, W.. **Comparação de modelos de machine learning aplicados a previsão de casos totais de dengue**. In: ANAIS DO XVI ENCONTRO NACIONAL DE INTELIGÊNCIA ARTIFICIAL E COMPUTACIONAL, p. 658–669. SBC, 2019.