3D Computer Vision Towards the Robotic Harvest of Shiitake Mushrooms

By

ROWLAND, THOMAS EDWARD

MSc Robotics Dissertation

Department of Engineering Mathematics
UNIVERSITY OF BRISTOL
&
Department of Engineering Design and Mathematics
UNIVERSITY OF THE WEST OF ENGLAND

A MSc dissertation submitted to the University of Bristol and the University of the West of England in accordance with the requirements of the degree of MASTER OF SCIENCE IN ROBOTICS in the Faculty of Engineering.

October 16, 2023

Declaration of own work

I declare that the work in this MSc dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Thomas Edward Rowland - 05/09/2023

Acknowledgement

I would like to thank:

Professor Mark Hansen, my project supervisor, for listening to my project ideas and his continued support in planning and conducting this project. It has been a pleasure to work with him.

Georgina Thompson, for her patience and support over the years and for always encouraging me to follow my interests.

Abstract

There is a substantial requirement for the modernisation of our agricultural practices. Current methods are damaging to the environment, inefficient and susceptible to economic and geopolitical pressures resulting in labour shortages and food wastage. However in the last two decades, robotics and machine learning, have undergone dramatic technological growth and it is now feasible to automate and optimise many of the agricultural processes previously reliant on human labour; such as the harvesting of crops. Attention thus far, both from academia and industry has focused on the most profitable crops such as apples, tomatoes and strawberries. This project however looks to broach a new challenge, in exploring the utilisation of computer vision and machine learning in the harvesting of Shiitake mushrooms. Shiitake mushrooms are one of the most valuable gourmet mushrooms, but also the most labour intensive to grow. The work conducted in this project has proven that current state-of-the-art techniques are suitable and effective at multiple stages throughout the harvest of shiitake mushrooms. Through the creation of the first publicly available segmented shiitake mushroom dataset, YOLOv8-seg and Detectron2 Mask R-CNN models have been trained to an mAP of 94.9% and 77.7%, segmenting up to 85 shiitake mushrooms in a single image. Additionally exploratory work with a smaller keypoint dataset was conducted to determine the suitability of both of these architectures to plot the cut points for the harvest operation with results proving the feasibility for the task. Further information and code relating to this project can be found at the projects GitHub Repository [1] (`https://github.com/trow-land/MSc-Dissertation-Shiitake-Harvest`)

Number of words in the dissertation: 13315 words.

Contents

Page

1 Introduction **7**

2 Literature Review **11**

2.1 Review Methodology 11

2.2 Key Computer Vision Concepts 11

2.3 An Overview of Shiitake Mushroom Cultivation 16

2.4 Robotics and Automation in Agriculture 17

2.5 Computer Vision Techniques in Agriculture 19

2.6 Summary 24

3 Research Methodology **25**

3.1 Initial Hypothesis 25

3.2 Dataset 25

3.3 Model Choice and Annotation 29

3.4 Model Training 33

3.5 Depth Model Training 36

3.6 Depth Reconstruction 38

3.7 Cut-Point Estimation 39

3.8 Evaluation Metrics 43

4 Results and Discussion **48**

4.1 Colour Instance Segmentation 48

4.2 RG-D Instance Segmentation 52

4.3 Cut Point Estimation 53

4.4 Discussion 57

5 Conclusion **60**

5.1 Future Work 61

References **68**

List of Tables

3.1 Key Hyperparameters for Training a CNN 34

4.1 Comparison between Detectron 2 Mask R-CNN and YOLOv8m-seg 48

4.2 Evaluation Metrics for Bounding Box and Segmentation 52

4.3 Keypoint detection results for each model on the validation set 53

1 Introduction

Current intensive agricultural practices are in need of a significant technological shift. Agriculture is responsible for a huge amount of environmental damage, caused by herbicides, pesticides, the excessive use of fertilisers and loss of biodiversity from large areas of monocrop growing methods. AgriTech is a field by which technological innovation is incorporated with current and new agricultural practices. It has the aim of increasing control, quality, nutrients and yield of a crop whilst aiming to increase the efficiency and reduce the environmental impact of producing sufficient food to support a healthy population. There are many different opportunities for technology to increase productivity throughout the lifecycle of a food product from crop sowing to crop harvest and beyond.

There has rarely been such a need for a change to an industry, with the global population predicted to increase to 9.7 Billion by 2050 from 7.9 Billion in 2021 and this increase in population will require an increase in food production by 56% [2], [3]. It is also estimated that 68% of that 2050 world population will be living in urban areas, compared to 55% in 2018. One option to support large urban populations such as these could be the uptake more localised and generalised artificial growing facilities. An example of this is the US Agritech company AppHarvest [4]. They utilise large state-of-the-art highly automated greenhouses to significantly increase the efficiency of the crop, enabling it to grow year round with a drastic reduction in the environmental impact. Growing in a system such as this can see up to 30 times more yield per acre compared to traditional open-field agricultural methods whilst using 90% less water and no chemical pesticides or soil erosion. Within automated growing systems and in open-field agriculture, ensuring the crop is successfully harvested, at the optimal time and without damage is still a significant challenge. Problems such as unpredictable weather patterns and supply chain complexity has resulted in crops that have been optimised for characteristics such as long shelf life and robustness against handling damage where instead they should be grown for a high nutrient content and taste. Socio-economic factors such as a shortage of seasonal farm workers, poor planning and limited demand sees that many of the crops

grown never actually get harvested. Two reports published by the WWF, [5] and [6], detail that up to 40% of food grown gets wasted at some point prior to consumption and that 15.3% of all food grown never makes it off of the farm.

Within both controlled environment agriculture and open field agriculture, robotics and computer vision can have an impact in making the required changes. Computer vision, a field of Artificial Intelligence that enables computer and robotic systems to interpret visual information can be applied extensively to various stages in the management of crops. Examples of possible applications include, crop detection, crop health monitoring, crop growth monitoring and information specific for harvest. This information can then be applied to robotic systems to enable accurate and dexterous harvest of many different crop types. Both computer vision and robotics have become increasingly present in agriculture in the past two decades due to technological advancements enabling vast increases in computing power which has allowed improved processing techniques and expertise to be readily available in all sectors of industry.

Research and commercial investment in computer vision and robotic harvest thus far has understandably focused specific readily consumed high-value crops. A 2022 paper [7] reviewing robotic harvesting applications showed that 61% of applications focused on just 3 crop types, Strawberries, Tomatoes and Apples. Computer vision and robotic research for crop harvesting can be a highly transferable topic, despite differences in shape, size and colour between different crop types, similar challenges must be overcome to apply this successfully. For this reason this project has been designed to broach a novel problem within the robotic harvesting, specifically how computer vision can be utilised for the robotic harvest of Shiitake mushrooms.

The Shiitake mushrooms, *Lentinula edodes*, ranks among the most popular and heavily cultivated edible fungi in the world. They are considered highly profitable to sell, but the cultivation and harvesting processes of these mushrooms are complicated and labour-intensive. The shiitake mushrooms are grown on a fruiting block made up of hardwood sawdust and additions such as wheat bran for nutrients [8]. In commercial cultivation these blocks typically weight between 2 to 5kg, upon which the shiitake mushrooms grow in all direction from the block. Due to the delicate nature of both the fruiting block and the shiitake mushroom crop, every single shiitake mushrooms has to he cut by hand by a human with scissors or a similar tool. For this reason the harvesting

operation for this crop is time consuming, labour intensive and expensive at scale.

Shiitake mushrooms unlike many other crops are already predominantly cultivated in an indoor highly regulated environment. Parameters such as light, temperature, humidity are dynamically controlled with sensors. The uniform growing conditions simplify aspects of the computer vision task, reducing external variability such as light, background and other plants, make the exploration into an automated harvest system increasingly practical. With robotic harvest of shiitake mushrooms the productivity of the crop, optimal timing for the harvest will increase the shelf life, reduce the change of damage and increase the taste of the mushroom.

This project represents another avenue of research to how robots can be utilised in agriculture, promoting intelligent data driven systems. By exploring the novel problem of shiitake harvest, insights learnt may be passed on to other crops, increasing the utility of technology in agriculture whilst promoting sustainable food production, reducing food waste and the inefficient use of growing resources.

1.0.1 Aims and Objectives

Aims

The aims of this project are as follows:

- To determine if current computer vision methods can be used to accurately detect shiitake mushrooms growing on a sawdust fruiting block
- To investigate whether 3D imaging data will improve the detection of the shiitake mushrooms compared to regular 2D imagery
- To research how current vision techniques can be used to determine suitable locations for a cut across the mushrooms stem

1.0.2 Objectives

To achieve these aims the following objectives have been decided:

- Create a dataset of shiitake mushrooms using a 3D camera
- Train an instance segmentation model to detect the shiitake mushrooms on the fruiting block
- Compare results using 2D and depth datasets to establish any advantage
- Create a separate dataset of the mushroom stems with annotated cut points
- To train a computer vision model to determine suitable cut-point locations

2 Literature Review

2.1 Review Methodology

The following literature review gives an explanation of some of the key concepts of computer vision on which this project is built upon. There is also background information regarding current techniques for shiitake cultivation and examples of state-of-the-art systems for various applications of autonomous crop detection and robotic harvesters.

The literature has been sourced with a mixture of internet searches, Google Scholar searches and reviewing related study citations.

2.2 Key Computer Vision Concepts

2.2.1 Neural Networks

Computer Vision is a subfield of artificial intelligence and deep learning. It couples traditional machine vision techniques with the advanced processing power of modern deep learning algorithms. This is achieved through the use of neural networks, a type of network designed to automatically optimise its internal parameters to identify patterns in input data, thereby making accurate predictions on new, unseen data.

To understand this process, it is essential to start with the basic building block: the neuron. An exploded view of the artificial neuron is shown in Figure 2.1. The neuron receives a series of inputs, each of which is multiplied by a corresponding weight. These weighted inputs are summed together, and the resulting sum is passed through an activation function—the most common being the rectified linear activation function, represented as $\mathrm{ReLU}(x) = \max(0, x)$. This process can be succinctly explained with the equation $Y = MX + C$, where Y represents the prediction, X the input, M the weights and C a bias term.

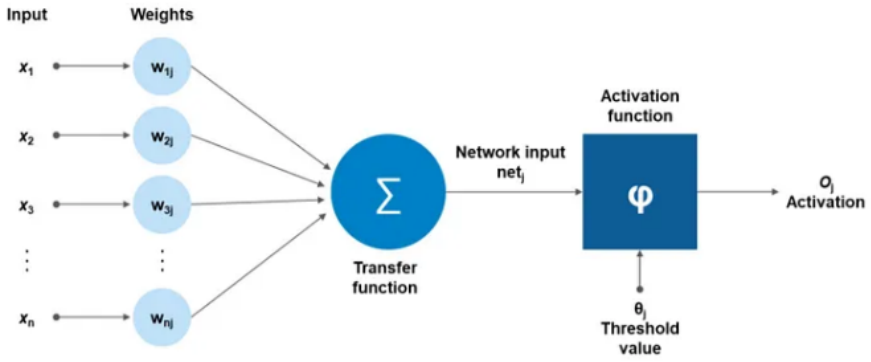


Figure 2.1: The make-up of a basic artificial neuron, replicated from [9]

Artificial neurons can be built up to form a layer, where multiple neurons are stacked in parallel, this can be expanded to form a network where the output from every neuron in a layer feeds into every neuron in the next layer. This concept can be repeated and adapted to form long and complex networks often resulting in millions, or even billions of neurons.

2.2.2 Learning

In the context of neural networks, learning refers to the optimisation of weights that are used to multiply the inputs. This optimisation occurs through a training process, enabling the network to make accurate output predictions for new inputs based on previously seen data.

While a detailed explanation of model learning is beyond the scope of this project, the process can be summarised as follows:

1. The model receives an input and makes a prediction
2. A loss function then evaluates this prediction, calculating the error between the predicted and actual outputs
3. The model uses this error to update its weights, aiming to minimise the loss
4. These weight adjustments are then backpropagated through the network, updating all weights to better optimize for the given inputs

There are many parameters that control this learning process, two key examples being batch size and learning rate. Batch size refers to the number of training examples processed before a weight update occurs, and the learning rate controls the magnitude of this weight update in a single step.

2.2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep neural networks that are particularly effective for tasks related to image perception. Unlike the networks previously described, where each neuron in one layer is connected to every neuron in the next layer, CNNs have neurons arranged in three dimensions: width, height, and depth. This architecture is specifically designed to automatically learn spatial hierarchies of features in input images, making CNNs highly successful for image-related tasks.

The convolutional network architecture starts with an input layer, the dimensions of which will be the same as the input width, height and depth. The depth of this layer will be the number of colour channels in the image, which is normally three for an RGB image. Then typically there will be a convolutional layer; the core building block of a CNN [10]. Convolutional layers apply multiple filters to every pixel of the input image to produce feature maps. The filters (or kernels) are a learnable weight matrix that slide or 'convolve' across the image. An activation layer like the ReLU previously discussed will transform the data to introduce non-linearity to the feature maps. Pooling layers can then be used to reduce the dimensionality of each feature map whilst retaining important feature information. Depending on the specific architecture this can be repeated multiple times with various special layers inserted depending on the specific problem.

Finally the data will be flattened and fully-connected layers (like those discussed in section 2.2.1) will be responsible for making a prediction.

Figure 2.2 shows a generic architecture for a convolutional neural network where the filter scans over the input image of a car.

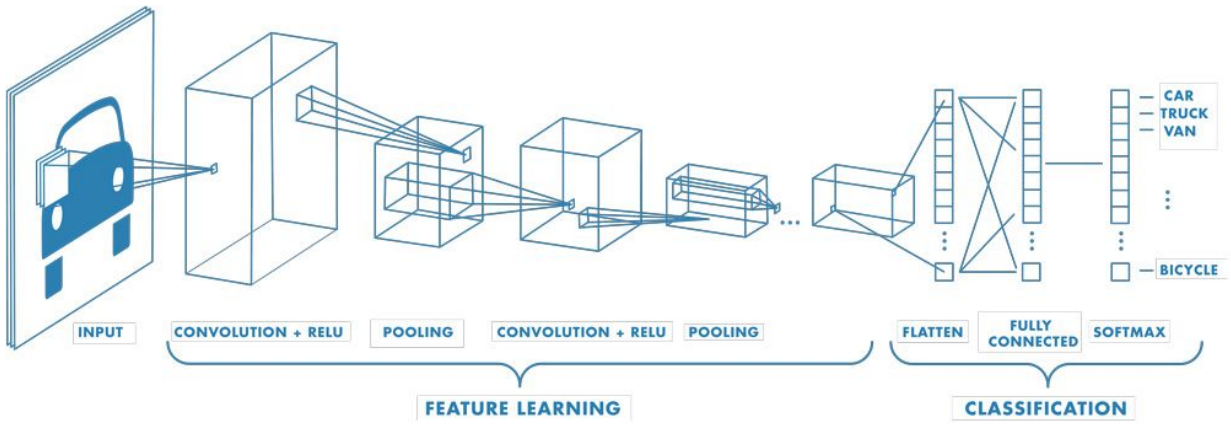


Figure 2.2: An example architecture for a convolutional neural network, replicated from [11]

Classification

Classification is among the most basic use for a convolutional neural network for image processing. The purpose of image classification is to look at the whole image and attach a single class label to that image. Progressions in classification, such as with AlexNet [12] in the the ImageNet [13] challenge in 2012 has defined how CNNs have been used with images ever since.

Object Detection

Object detection is a computer vision task that aims to look at the input images and locate and identifies specific objects. Object detection models are a progression from classification models, they look at a whole image, and can not only classify multiple objects within that image, they try to draw a bounding box around the object.

Convolutional networks have to be adapted to provide this extra functionality, the simplest approach could be to split the image up into various regions and to use a classification network to determine whether any objects are in that region.

Generally object Detection models can be slit into two categories: Two Stage Detectors and Single-Stage Detectors. An example of a two stage detector is the Faster R-CNN [14]. The architecture determines probably candidate object regions using a Region Proposal Network (RPN) and runs a classifier on the selected regions. Regions are determined when the RPN scans the image like a sliding window and suggests possible bounding box sizes. Two stage networks are generally

considered to be more accurate, but at the cost of computing power and processing time.

Single-stage detectors such as YOLO (You Only Look Once) [15] and SSD (Single-Shot-Detector) manage the classification and localisation in a single pass through the network. YOLO divides the image into grids that then predict multiple bounding boxes with class probabilities. If the class probability is above a defined threshold then it is considered a detected object. Conversely to two-stage networks, these are quicker, lighter but less accurate. As development progresses over the years the margins between two-stage and single-stage networks is getting smaller.

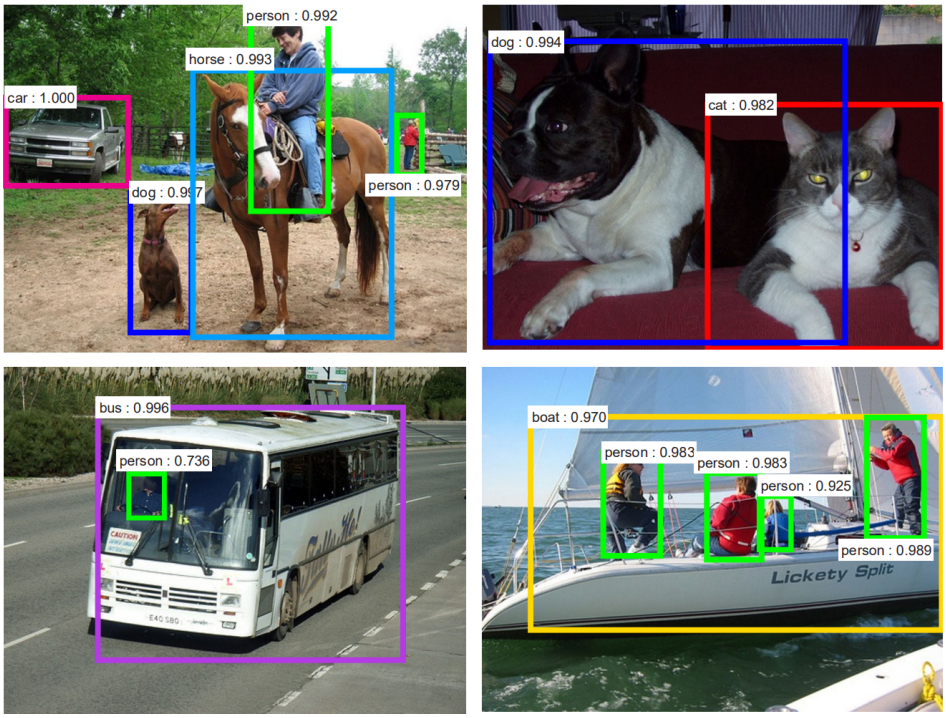


Figure 2.3: Example outputs from Faster R-CNN replicated from [14]

Object detection models generally produce two outputs: The class label which categorises the detected object and the bounding box coordinates. Some models, like in the example shown in Figure 2.3 output a confidence score for the bounding box, indicating the likelihood that the bounding box actually contains the labelled object.

Instance Segmentation

Instance segmentation extend upon the idea of using just a bounding box for object detection. Instead of encapsulating the object in a bounding box and classifying everything within that bounding

box as the object, instance segmentation models look to delineate the exact pixels in the image that make up the object.

The most common method for instance segmentation is with the Mask R-CNN [16]. The Mask R-CNN is the next iteration from the Faster R-CNN [14]. By adding a new parallel branch for predicting the object (Region of Interest / ROI) alongside the Faster R-CNNs branch for bounding box recognition. Instance segmentation models will have the an object binary mask output in addition to the class label and bounding box coordinates. The most recent YOLO iteration, YOLOv8 has also been equipped with a segmentation mode [17], previously it has only been suitable for object detection and classification tasks.

Keypoint Detection

Keypoint detection (also called landmark detection) models focus on predicting specific points in an image or an object. These points could just be regions of interest such as eyes or skeletal joints.

The most common application for keypoint detection is human pose estimations, where a basic skeleton is superimposed over a human to estimate the body position [18]. It has also been used previously used to make up components of object like a bike or car.

2.3 An Overview of Shiitake Mushroom Cultivation

The shiitake mushroom has been a staple in the Far Eastern (Japanese, Chinese and Korean) diet for thousands of years and has been used for both its culinary and medicinal characteristics since as early as 199 A.D [19]. Some of the known health benefiting properties of the mushroom include, immune-boosting, anti-inflammatory, cholesterol-lowering and anti-cancer properties.

It grows naturally on fallen hardwood logs, such as the *Shii* tree (a type of Japanese oak where is gets name from), oak, chestnut and beech. Cultivation of the mushroom started in China around 960 AD [19], [8] in a process that looked to simulate the natural growing conditions closely whereby cultivators stacked many inoculated hardwood logs outside. That process remained largely unchanged until the 1900's where commercial mushroom growers adopted sawdust bags as the growing medium of choice. The sawdust bags are composed of a mixture of hardwood sawdust with additional supplements such as wheat bran or soy bean hulls [8]. The shift to an artificial growing

medium saw an increased yield and efficiency [20]. Many of the original manual tasks associated with growing on logs where no longer required and less reliance on weather conditions meant that a good crop was more reliable. However harvesting of the shiitake mushrooms became more complex as a result of the change. The sawdust fruiting blocks themselves are fragile and easily damages if knocked. Subsequently harvesting must be carried out by a human, whereby each individual mushroom is hand-cut with scissors to avoid damage to the mushroom or the fruiting block.

Recent decades have seen a surge in popularity for the shiitake mushroom and they are now commonly consumed worldwide. The shiitake mushrooms global market size is predicted to from 2.3 Billion to 4.7 Billion USD by 2030 [21].

2.4 Robotics and Automation in Agriculture

AgriTech (Agricultural Technology), often shortened to AgTech, is an umbrella term representing the inclusion of technology into the agricultural sector. AgriTech encompasses many subfields including Precision Agriculture, Agricultural Biotechnology, Agricultural Data Analysis, Controlled Environment Agricutlure and Robots and Automation. It is a rapidly growing industry and is expected to reach a total global market size of USD 45 Billion by 2028, with a compound annual growth rate of 14% [22]. This expansion is fueled by technology advancements in many areas such as imaging and processing as well as an increased acceptance by the agricultural community to include technology in their practices.

Robotics and automation is currently deployed at various stages of the produce journey, automatic seed drills have been used for hundreds of years, and mass harvesting systems for wheat rice and corn have become standard in many parts of the world. With advancements in Artificial Intelligence (AI) and Machine Learning (ML) robots can have the capability for a much more precise approach. A new wave of agricultural robots are being developed that can harvest fruit and vegetables depending on characteristics such as size and ripeness and some systems are even capable of harvesting multiple different types of crop [23]. Driven by the growing demand for a stable food supply network and resilience against labour shortages there has been a surge in interest for research on these systems.

Each fruit and vegetable crop presents its own unique challenges for harvest. Shapes and sizes of plant and fruit differ between species and within a species of crop making a generalised system significantly challenging. There are distinct different workspace requirements between some fruits and vegetables meaning that currently, a crop specific robot is the solution for efficient harvest. Deep learning has dramatically improved the ability of systems to learn visual characteristics for crop types and is utilised in the vast majority of dexterous harvesting robots.

The ultimate goal for harvesting robots is to create a system that can robustly generalise between different crop types and varieties, enabling it to harvest multiple different crop types simultaneously for work in diverse cultivation areas.

2.4.1 Harvesting Robots

Horizontal crops are those that require a more horizontal workspace. Typically the fruits will grow directly out of the ground such as asparagus and beetroot. A crop such as asparagus is well suited to robotic harvest. They are most commonly commercially grown in long rows where (assuming no significant weeds), the only part of the crop visible above ground is the asparagus spears for harvest. The current state-of-the-art in asparagus harvest comes from the UK company Muddy Machines [24] with a robot called Sprout. The Sprout robot straddles the row of crops, autonomously moving along the row, differentiating the asparagus from any surrounding weeds and it dexterously cuts the asparagus spears to the farmers specification. The robots vision system for this autonomy comes from 2D and 3D cameras situated on the underside of the robot. The system is capable of harvesting 50,000m^2 in one day. Asparagus is a crop that only has a 12-week season making which increases the risk from labour shortages.

Other horizontal crops such as beetroot and lettuce have received research interest since in the past decade. In 2019, researchers at the University of Cambridge have developed a robot names 'Vegebot' to tackle the harvest of iceberg lettuce [25]. Iceberg lettuce harvest entails different challenges when compared to asparagus. The harvest operation aims to only remove the head of the lettuce, that head must therefore be distinguished from the surrounding leaves and the head is very susceptible to damage during harvest and transportation. The Vegebot was able to successfully harvest 88% of the tested lettuces with a complete harvest time of 31s.

Vertically grown crops have received for research attention and as a result there are more commercial vertical harvesting robots in operation. An important example is the Virgo robot from RootAI - now part of AppHarvest [23] [26]. Originally focused on tomatoes the robot has since improved its utility to be able to harvest multiple crops such as peppers, cucumbers and strawberries. This robot is a big step towards a universal harvesting robot. It can be used in an inside or outside growing environment and operates itself autonomously amongst other human workers. The soft grip end-effector adapts itself to each crop type and orientation to allow a damage free harvest. The robot employs a set of cameras and infrared laser to generate 3D colour scans of the work area from which it can distinguish and analyse the crops for harvest. Reinforcement learning algorithms are employed by self-evaluating the success of each harvest operation.

Like mushrooms, berries have the challenge of a delicate crop. Two examples where companies have developed state-of-the-art systems are Agrobots strawberry harvesting robot [27] and Fieldwork Robotics, raspberry picker [28], [29]. The Agrobot is a large robot that spans over multiple rows of strawberry troughs. The underside houses up to 12 robot arms which are capable of harvesting strawberries simultaneously without touching the fruit of the strawberry - only the stem. This approach reduces the risk of damage to the strawberry and increases the shelf life of the product. Fieldwork robotics [28], a spin-out of the University of Plymouth approaches the delicate fruit problem with a different approach. Utilising soft robotics for their end-effector on 4 robotic arms guided by 3D cameras allowing different soft fruit types to be harvested with minimal risk of damage and slip.

2.5 Computer Vision Techniques in Agriculture

Discussion thus far has focused on the benefits, potential and capabilities of robotic harvesters for fruit and vegetables. A crucial aspect of their success is a system that can view and interpret visual information, which allows for accurate detection and localisation of the crop in question. Additionally the wind, or the actual harvesting operation itself, could cause the fruit to move complicating the harvest further. Many systems previously mentioned such as [24], [26], [30] also analyse the detections to determine the ripeness and readiness for picking allowing for a more optimal harvest for each fruit.

2.5.1 Sensors

Different sensing modalities can be employed to interpret the crop and the surrounding environment. The simplest option is a regular 2D RGB (Red, Green, Blue) camera. Images from an RGB camera is sufficient for many tasks, it is the most common data type for visual information and is capable to display features such as shape, colour and texture. 2D images can be easily annotated and implemented into most existing computer vision algorithms with minimal alterations. However using 2D images as the sole information source can have some disadvantages. As explained in [7], [31] 2D imagery is susceptible to changes in illumination and can have trouble differentiating the fruit from its surrounding, such as the lettuce head from surrounding leaves with the Vegebot [25]. For these reasons another channel, depth, is employed to add additional information to the system. Depth can be combined to the image to form a RGB-D image. There are multiple methods by which depth information can be collected by the system such as stereovision, or a time-of-flight (ToF) camera.

Stereovision deduces depth in a similar way to the human eye, by placing two cameras a known distance apart the depth can be inferred by triangulating the disparity between the images [31], [32]. This method has several limitations, they are unable to calculate the depth for anything below the focal point of the two cameras and they do not work well with untextured surfaces (such as walls) which are very common in man made environments. The process of fusing the input image streams can also require significant computational resources.

ToF is a more active method for determining distance. ToF works by emitting a pulse of light, often in the infrared wavelength. The pulse reflects off of surfaces and the time between the pulse being emitted and detected determines the distance (or depth) of the object to a high accuracy [33].

ToF has some advantages over stereovision. It is effective against untextured surfaces and changing light conditions. Additionally computation required by the system to is less than the image processing for stereovision. The accuracy is consistent within the operating limits of the camera which lends itself well for the purpose of robotic harvest. Time-of-flight sensors do have some notable drawbacks however, strong IR sources such as direct sunlight can cause ambient light interference and objects that are too near to the camera have a poor depth accuracy. For these reasons, a combination of 2D and 3D imaging is generally used in conjunction with each other and

together make for a an important and effective component when designing an agricultural robot [31].

2.5.2 Detection Algorithms used in Harvesting Robots

A systems ability to interpret visual data has drastically improved since the advent of deep learning and with advancements to the Graphics Processing Units (GPU) allowing for vast amounts of visual data to be utilised in a system. Prior to these advancements, conventional methods for machine vision within agriculture often relied more on colour to differentiate a fruit from its background. In 2011, [34] used a series of conventional methods to first segment the tomatoes by equating the contrast between the tomato colour and the background, then by converting from RGB colour space to HSI and YIQ colour spaces and looking for red in the tomato to determine ripeness. The model did well, recognising ripe tomatoes with an accuracy of 96% but still struggles with darker lighting conditions making the tomato appear more red. This approach was highly specific, and would not be able to translate to other environments or crops. Machine learning algorithms such as K-Nearest Neighbour (KNN) and Support Vector Machine (SVM) have also had success in classifying crops. [35] utilised both of these methods to detect mature broccoli heads achieving an 95% accuracy.

The majority of dexterous harvesting robots will utilise some form of deep learning algorithm to locate the fruit. The two-stage Faster R-CNN was used for the detection of orchard fruits including apples, mangoes and almonds in [36]. They found that the model managed an impressive level of accuracy despite the relatively small dataset, achieving an F1 score of accuracy of over 0.9 or 90% on apples and mangoes, at the time superior to other studies. The F1 score as shown below in Eq. 2.1 is a measure of how the model performs by taking into account both the precision (how many of the predicted positive instances are actual positives) and the recall (how many of the actual positive instances were predicted as such) of the model. A high score, such as the one achieved by the Faster R-CNN indicates that the model has a balanced performance with respect to both false positives (predicting a fruit where there isn't one) and false negatives (missing a fruit that is present).

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2.1}$$

The Faster R-CNN was also used for 3D passion fruit detection in [37]. They trained the modified Faster R-CNN on the RGB data and depth data separately and the fused the detectors together for RGB-D detection. Utilising both RGB and depth made their model more robust to the changing light conditions. Similarly a modified Mask R-CNN was employed for asparagus harvest in [38]. Depth and colour information was used to create a Depth-Aided Mask R-CNN (DA-Mask R-CNN). The depth stream was used for both the region proposal network and the classification stage. The additional depth information proved useful in segmenting the asparagus spears from the ground, improving the accuracy of the system but increasing the processing time for detection.

Cutpoint Estimation

There are a variety of techniques that you can utilise for making the cut in a harvest operation. The specifics will dependant on the crop in quest, sometimes the geometry of the crop makes it simple and reasonable uniform like with the asparagus [38]. If the crops get picked instead of cut, like with RootAI's virgo, and FFRobotics apple picking robot [23], [39] then it is enough to have the manipulator take hold of the crop then twist and pull it off the plant.

Other times with hanging fruits like a vine of tomatoes or a pepper a more active approach needs to be considered. One solution is to use keypoints as discussed in section 2.2.3, to determine the correct place and angle of a cut. This method was used with success by [40], [41] and [42]. [40] used RGB-D images to first mask a vine of tomatoes, then mask the penduncle (the stalk connecting the tomatoes to the stem). With the regular geometry of a hanging vine of tomatoes, the penduncle mask was used to estimate a keypoints at the top and bottom of the mask with a cut point finally chosen in the middle. [41] repurposed two human-pose estimation models, the Stacked Hourglass Network [43] and Simple Baselines model [44] for comparison with their keypoint detection network trained to determine suitable cut points on the stems of citrus fruit. Lastly researchers at Bristol Robotics Laboratory's Centre for Machine Vision [45] adapted a facial landmark detector for use cutting lettuce. They trained the facial keypoint detection model, to instead identify cut points either side of the stem, by which a cutting tool could be used to precisely seperate the head of the lettuce. All of these studies show that keypoint detection methods can be effectively used to determine cut locations when the parameters of the crop require delicate harvesting.

2.5.3 Computer Vision and Robotics in Mushroom Cultivation

From reviewing the available literature it is evident that the majority of research efforts have been focused on fruit and vegetable harvest, with little attention to fungi. Fungi are a challenging crop, they are delicate, time-sensitive and complicated to harvest. There have however been some efforts to use computer vision and robotics within mushroom cultivation.

Mushroom Detection

Not all mushrooms are grown on sawdust blocks like the shiitake, button mushrooms for instance are grown vertically on a large horizontal beds. Button mushrooms have had some research attention due to this regular growing position. In 1994 John Reed attempted to develop robotics harvester by using a simple rig that applies suction to the mushroom cap then twists out the mushroom [46]. Their system used tradition image analysis techniques to locate the white mushrooms from the black bed allowing the robot to detect 88% of the mushrooms and successfully remove 67.5%. More recently adaptive contouring has been used to segment button mushrooms in [47] by recognising the circular shape of the mushroom when viewed from above.

Deep learning has allowed more distinct features to be used for detection of less regular mushrooms. An oyster mushroom detecting system was designed using the SSD in [48]. Their system was trained on 4000 images and achieved an accuracy of 95.1%. This gave their system an accuracy of 2.43mm, well below their threshold of 5mm which would lead to an unsuccessful harvest.

Shiitake Mushrooms

There is limited research into automation during shiitake mushroom cultivation. As previously discussed, they grown on all sides of a fruiting block but unlike other mushroom varieties like lions mane or oyster mushrooms they don't grow in large clumps. This means that both detection and harvesting systems will have to contend with a large number individual mushrooms in multiple orientations.

Shiitake mushroom classification was performed post-harvest by [49]. Their system used a vision transformer which is a different deep learning architecture derived from natural language processing. Their transformer outperformed the 3 compared CNN architectures at classifying the

3 quality-types of shiitake mushroom, beating the best performing CNN, InceptionV3 98.5% to 96.4%.

The single-stage detector YOLO was used for shiitake mushroom detection in [50] and [51]. [50] created their detector by modifying YOLOv3 [15] to form MYOLO which which they trained on 1800 shiitake mushrooms and achieving an accuracy of 97.1%. Their study was described as primer for robotic harvest, their dataset however only contains images of the mushrooms from above and their model focuses on the details on the top of the caps. Conversely Mushroom-YOLO was created and used in [51] in a more realistic harvesting orientation. Their system was part of a whole automated growing area where they are trying to replicate conditions where the shiitake mushrooms flower (split), enabling growers to fetch 5-8 times the market price due to the desirable mushroom heads [49]. Their Mushroom-YOLO system, modified from YOLOv5 was trained on 300 shiitake mushrooms and claims an accuracy of 99.24% at differentiating flower mushrooms from regular shiitake mushrooms.

2.6 Summary

This literature review has covered the base from which this research project is extending upon. Advanced technology is finding its utility in more areas of agriculture every year, equipping new and existing farms with robotics and computer vision system. Computer techniques can be effective for the cultivation of most crops but exploring the unique challenges faced by fungi is an important step in the ubiquity of harvesting robots. Agricultural crop harvesting robots is a an active research area and there are more applications researched in line with progressions in deep learning and computer vision.

This projects works towards progressing current research into to the use of robots in mushroom cultivation. Currently there are no dexterous mushroom harvesting robots and research into this field is in its early stages.

3 Research Methodology

3.1 Initial Hypothesis

The following experiments were set up to establish the viability of the current state-of-the-art computer vision techniques in the harvest of shiitake mushrooms and to investigate possible techniques for increased accuracy.

It is hypothesised that modern computer vision techniques will achieve a sufficient level of accuracy for use in the robotic harvesting of shiitake mushrooms. Furthermore, it is thought that the incorporation of depth information will improve system accuracy, given the similar appearance between shiitake mushrooms and the fruiting block.

3.2 Dataset

Deep learning models require a significant amount of annotated data to accurately learn the patterns in the data to make predictions. The dataset should be as diverse as possible to prevent any biases in the data and to ensure that the model will be able to generalise well to other required situation - for example a different background. After searching online and through related works, it was established that there are currently no open-source datasets of shiitake mushrooms growing on a fruiting block in either 2D colour or in 3D. To satisfy the aims of this project and investigate the hypothesis, curation of a new 2D and 3D dataset of shiitake mushrooms would be required.

3.2.1 Choice of Sensor

Initial testing was carried out using two different 3D cameras; the Asus Xtion Pro and the Microsoft Azure DK [52], [53]. The Xtion Pro is an older camera that designed for general consumer and developer usage with a focus on gesture control. It is controlled through the OpenNI Frameworks, an open-source software developement kit (SDK) [54] and has both a colour output and depth

output with a range of 0.8 to 3.5m. The depth is calculated by projecting and infrared (IR) structured light pattern made up of a dot matrix. The Azure DK camera, with a range of 0.5 to 3.8m, is more advanced with its technology and is primarily designed with commercial and research developers in mind. It utilises time-of-flight technology to determine depth, measuring the time for a emitted IR pulse to rebound off objects. Microsoft provides an SDK for usage [55].

Comparative testing between the two cameras was carried out with a fruiting block with shiitake mushrooms in the early stages of growth to determine how capable each camera was in distinguishing individual shiitake mushrooms from the background using the depth output. Figures 3.1 and 3.2 show examples of the test.

Figure 3.1: 3D and colour from the Xtion Pro Live camera of an immature shiitake mushroom fruiting block as viewed with the OpenNI SDK.

This initial testing show that both cameras do have the depth precision so distinguish the shiitake mushrooms from each other and the surrounding. Visually both comparing the depth and colour outputs the Azure looks to be the better option for the remainder of this study. The colour output is capable of much higher resolution and the depth output suffers significantly less noise and data loss compared with the Xtion.

3.2.2 Data Collection

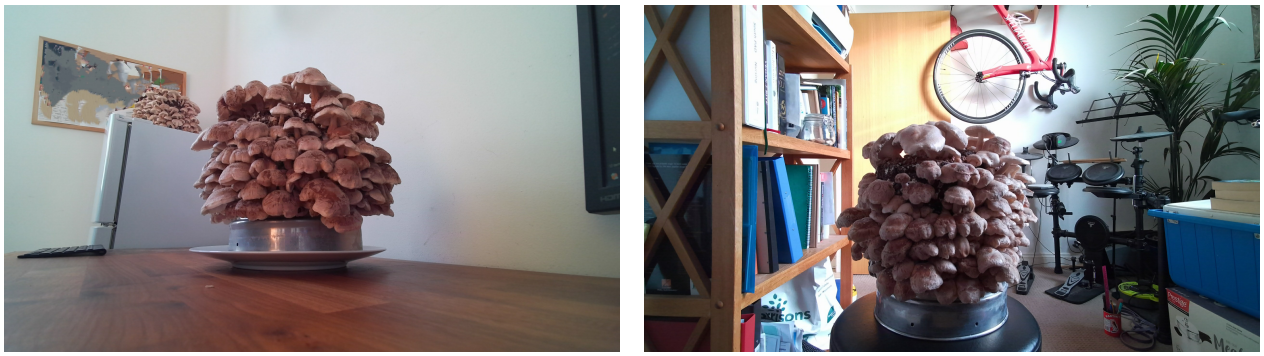
Two shiitake mushroom fruiting blocks were sourced online and further testing was carried out to determine the camera settings that gave the highest visible granularity of the shiitake mushrooms in the depth output.

The azure camera has 4 different depth settings relating to the field-of-view (FOV) and the bin-

Figure 3.2: Examples of the colour and 3D outputs from the Azure DK 3D camera, viewed with the Azure Kinect Viewer which is part of the SDK.

ning/combining of pixels. They are: Narrow-FOV Unbinned, Narrow-FOV 2x2 binned, Wide-FOV Unbinned and Wide-FOV 2x2 binned. Visual comparison of the outputs showed that WFOV Unbinned was the best for this task.

Due to the fast growing nature of the mushrooms it was essential that sufficient data was collected for future use in the project but given the time constraints of the project there was limited scope for fully diverse data. The two shiitake mushroom fruiting blocks were used, in front of two different backgrounds, one simple and one cluttered. Additionally the data was captured from three different view points, level with the camera, from slightly below looking up and from slightly above looking down. Further use of these view points was predominantly limited to the level and underneath views due to the shape of the mushroom necessitating that harvest occur from underneath the mushroom. As the mushrooms grow on all sides of the fruiting block the block was rotated and data collected throughout giving a lot of diversity given the limited number of fruiting blocks. It us estimated that each block had about 200 individual shiitake mushrooms growing which gave a sufficient variation in shape and size. Figure 3.3 shows these fruiting blocks from 2 of the perspectives and each background.



(a) Plain background looking underneath

(b) Busy background looking level.

Figure 3.3: Two example of the colour images from different viewpoints with different backgrounds.

3.2.3 Recording

With the camera view correct, the recording of the data has to be achieved through a command-line instruction:

```
k4arecorder.exe -d WFOV_UNBINNED -r 30 -l 60 -c 1440p --imu OFF {
    output_path}shiitake1_below.mkv
```

This command records with the following parameters as specified at [55]: depth mode = WFOV_Unbinned, Frames per second = 15, recording length = 60s, colour resolution = 1440p, Inertial measurement unit = OFF. The fruiting blocks were rotated slightly by hand and left to pause at each point for a second or two. The 60s video allowed for several full rotations of the block with each mushroom being viewed from slightly different angles.

The result is a recording file in the Matroska (.mkv) container format. The video file contains tracks for storing the colour, depth, IR and IMU data, although our usage only requires colour and depth. The ffmpeg tool [56] or a python script [1] can then be used to unpack the the colour and depth information into separate folders for annotation with the command-line instruction:

```
python azure_kinect_mkv_reader.py --input "D:/ShiitakeHarvest/
    AzureCaptures/WFOV_UNBINNED/shiitake1_below.mkv" --output "D:/
    ShiitakeHarvest/AzureCaptures/WFOV_UNBINNED/colour"

# or

```

```
ffmpeg -i D:/ShiitakeHarvest/AzureCaptures/WFOV_UNBINNED/shiitake1_below.
    mkv -map 0:0 -vsync 0 D:/ShiitakeHarvest/AzureCaptures/WFOV_UNBINNED/
    colour/shiitake1_below_colour%04d.png
```

These scripts will separate the .mkv recordings into 900 colour frames and 900 depth frames that will appear black and are of type uint16.

Given the time constraints and requirements of project each of the video frame folders was reduced by only keeping every 20th image, ensuring that both the colour frames and depth frames were still matched correctly.

3.3 Model Choice and Annotation

For harvest the model should be able to view the image of the shiitake mushroom block and detect where each of the shiitake mushrooms are in that image. This information will give you each mushrooms location as well as an expected yield for each frame. A regular object detection algorithm such as Faster R-CNN [14], SSD [57] or YOLOv8 [17] would suffice for this task. Annotations and the output would be in the form of a bounding box whereby every pixel contained within the bounding box is associated with that class. The close and overlapping spatial relationship of this crop however would make it difficult to isolate individual mushroom in an annotation and the irregular shape and size of the visible part of the mushroom meant that a segmentation model would be more appropriate. An instance segmentation model such as the Mask R-CNN [16] or YOLOv8-seg [17], can do everything that other bounding box object detectors do but instead of just a bounding box, a polygon segmentation mask is added to the output.

3.3.1 Data Annotation

There is a wide selection of data annotation software available for annotating computer vision tasks. It was decided that Roboflow would be a good option for this segmentation task as it provides an intuitive user interface to do the annotations and to manage various characteristics of the dataset.

Annotating segmentation masks is the most time consuming of the visual annotation tasks. It involves outlining the object as closely as possible with points making a polygon surrounding the whole object. The more accurate you can be with the annotations, the better a model will be

at giving accurate predictions for where an object lies in a image. The Roboflow software does contain a 'Smart Polygon' feature, where much of the polygon can be automatically outlined with one click. This did save time but required a lot of touching up due to the close overlapping nature of the mushrooms. The smart polygon often struggled to annotate the mushroom stem and the cap as part of the same mushroom so it was often quicker to annotate manually than to find the sweet spot point for the smart polygon to work correctly. An example of the the Roboflow user interface and the smart polygon feature can be shown in Fig. 3.4

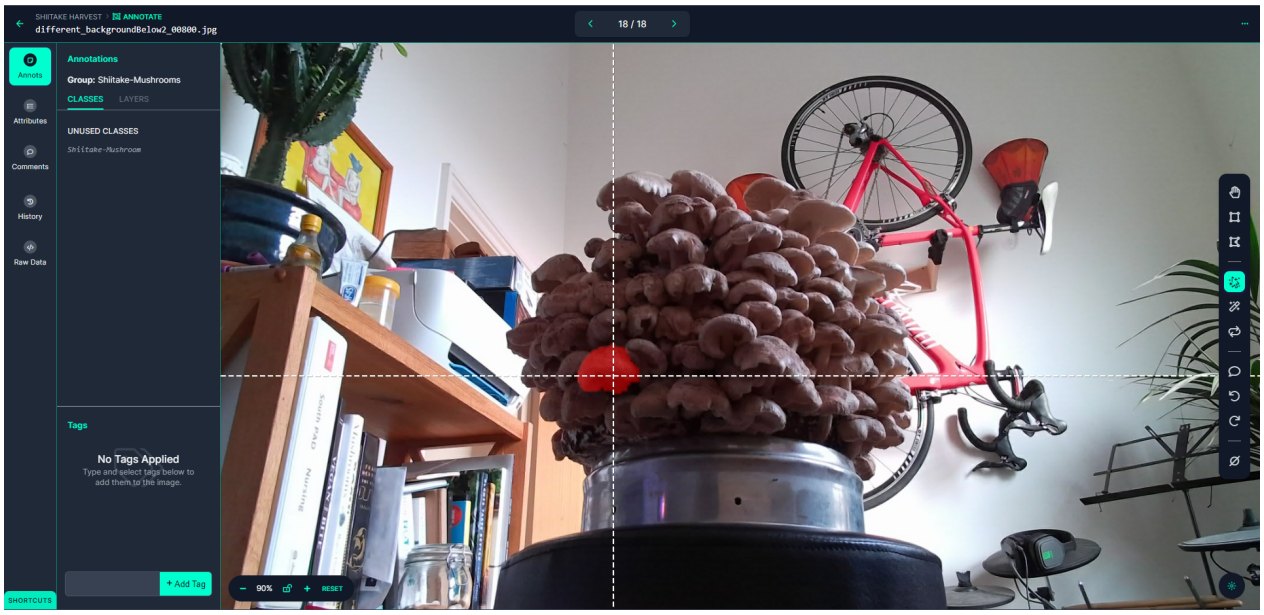


Figure 3.4: An example of the Roboflow user interface and annotating with the 'Smart Polygon' feature.

The larger the dataset, generally the more accurate the model will be so it was important at this early stage of the project to ensure there were as many annotations as possible. Some basic rules were established to govern how annotation would be carried out:

- Annotations would be outlined as clear as possible from the picture with the human eye
- Extremely small or unclear mushrooms would be left out to avoid guessing
- Only the visible part of the mushrooms were annotated
- If mushroom was split into two because of occlusion from another mushroom, the part including the cap would be annotated.

The number of annotations per images was very high for a typical dataset, normally a dataset will focus on a few object so it was unknown how well a model would cope with such a large number of instances. Figure 3.5 shows an example of a fully annotated image. In that image there are 73 individual shiitake mushrooms annotated.

Figure 3.5: A fully annotated shiitake mushroom fruiting block from the dataset.

In total a total of 92 images were annotated with 5,802 individual shiitake mushrooms giving approximately 63 mushrooms per frame. These images then get separated into the required train-validation-test split. The training images are used to update the weights during training, the validation set used for validating the model during training, and the test set kept for a separate evaluation post-training on completely unseen images.

Roboflow allows the addition of data augmentation which can effectively increase the size of the training data by adding altered versions of the training examples. For this project, only horizontal flip augmentation as added to maintain realistic examples the manipulator might come across. This effectively doubling he training set resulting in a training:validation:test split of 118:15:10 after the augmentation.

3.3.2 Instance Segmentation Algorithms

There are multiple possible state-of-the-art instance segmentation algorithms to choose from for this task. For many years the Mask R-CNN has been the benchmark algorithm for segmentation so the Detectron 2 Mask R-CNN implementation [58] will be included for testing. For general object detection the YOLO algorithms have showed impressive accuracy and speed. The most recent iteration of the YOLO (You Only Look Once) family, YOLOv8 [17] has the utility to do classification, detection, segmentation and pose estimation so the segmentation model will also be implemented as to have a comparison.

3.3.3 YOLOv8

YOLOv8 [17], released on 10th of Jan 2023 by Ultralytics is the latest iteration on the YOLOv5 model. It offers a selection of model sizes, with YOLOv8n/s/m/l/x containing 3.2, 11.2, 25.9, 43.7 and 68.2 millions neurons respectively. The increased model size will represent addition layers such as more convolutional layers, more filters or repeated blocks. There has not been an official research paper released alongside the architecture but there is extensive information within the documentation and Ultralytics Github profile [17].

The single-stage YOLOv8 is the first and only YOLO model that houses extra functionality besides basic object detection.

3.3.4 Detectron 2

The Mask R-CNN originally developed by Ross Girshick et al [16] in 2017 has long been the state-of-the-art for segmentation models. It was released by Matterport in the original implementation but has since been re-released within Detectron2. Detectron2, released by Facebook AI Research (FAIR), is an open-source platform that implements state-of-the-art object detection algorithms, including Mask R-CNN [58]. It is written in PyTorch and offers a modular and extensible architecture that makes it easier to customise models and configurations. Detectron2s 'model zoo' contains a variety of models pre-trained on different datasets for easy benchmarking.

The Mask R-CNN model used in this project comes with a ResNet-50 backbone and is pre-trained on the Microsoft COCO dataset [59]. The ResNet-50 backbone is a CNN with 50 layers

and is a commonly used framework in feature detection models. Like YOLOv8, Detectron2 has the capability for multi-modal outputs including, classification, segmentation, object detection and keypoint detection.

3.3.5 Pre-training

Both models have been pre-trained on the Micrsoft COCO dataset [59], a dataset that contains 328k annotated images containing every objects. Pre-trained models are extremely useful, especially with smaller custom datasets like the one in this project. The models have already been highly optimised to pick up the basic features and shaped within an image. With this advanced starting point, further training will not only be quicker and less resource intensive, but more accurate as the training process will involve specialising the parameters at the end of the network which handles the high level features and not every layer.

3.4 Model Training

Training for the models took place on Google Colaboratory (Colab). Google Colab allows for quick and easy environment set up and GPU enhanced training which makes it a valuable took for prototyping models. The Google Colab script itself is in the form of a modified Jupyter Notebook allowing users to write and execute Python code where each cell in the script should be run in order once the runtime has been connected to the Google hardware. Google Colab allows limited use to powerful Graphics Processing Units (GPUs) and Tensor Processing Units (TPU's) that are essential to speed up computations when training vision deep learning models due to the large memory requirements.

3.4.1 Hyperparameters

Finding the optimal hyperparameters is a complex and often time-consuming task that can be addressed through various hyperparameter tuning techniques. The majority of the hyperparameters were kept at their default values as provided by the Ultralytics implementation. This decision was based on this being preliminary experiments to understand the underlying principles of the model.

Table 3.1: Key Hyperparameters for Training a CNN

Hyperparameter	**Description**
Learning rate	Controls the step size during gradient descent
Batch size	Number of training examples in one forward/backward pass
Number of epochs	Number of complete passes through the training dataset
Activation function	Function applied at the end of each neuron (e.g., ReLU, sigmoid)
Optimiser	Algorithm to update weights (e.g., Adam, SGD, RMSProp)
Loss function	Measures the difference between predicted and actual values
Regularisation	Technique to prevent overfitting (e.g., L1, L2, Dropout)
Kernel size	Size of the filter in convolutional layers
Stride	The step size the convolutional filter takes
Momentum	Helps accelerate gradients in the right directions

However, some adjustments were made to cater to the specific dataset and the problem being solved.

Some important hyperparamaters and their descriptions are shown below in Table 3.1.

3.4.2 YOLOv8 Training

The Ultralytics YOLO models have been highly optimised for ease of use, allowing more people to get involved and use YOLO in their machine learning projects. As such the Colab notebook is reasonably concise and easy to follow, with the Ultralytics repository cloned the pre-trained YOLOv8-seg moodel can be imported and initialised for training. Downloading the dataset to the runtime can be quickly achieved with a Roboflow download code. The model is then trained as per the defined hyperparamters. Validation on the validation data, and inference is conducted on a test dataset with specific parameters. Finally, the inference results are displayed numerically to evaluate the model's performance and inference images are saved for viewing.

Initial testing was carried out prior to the main training runs to determined whether there would be any reasonable level of learning given the large number of instances per image and the relatively small number of images.

For this initial training run the only hyperparameters that were changes were the batch size

and the number of epochs. The batch size was set to 2 to balance manage to available computing resources. The number of epochs was set to 50, a value chosen after monitoring of the training log.

The training log and validation results provided substantial evidence that the model was learning well, and the dataset size was deemed sufficient for the project's requirements. Further tuning of hyperparameters could be conducted in future iterations to possibly enhance the model's performance, but for the scope of this project, the chosen settings were found to be satisfactory at this time.

3.4.3 Detectron 2 Mask R-CNN

Implementation of the Detectron 2 Mask R-CNN is more involved and offers a powerful tool for both object detection and instance segmentation but this architecture requires careful configuration and tuning than YOLO. The Detectron2 repository is imported and the pre-trained Mask R-CNN is loaded from the Detectron2 model zoo. As with the YOLO script the dataset can be easily downloaded into the runtime, but then is required to be registered in COCO format with visualisation to verify the data. The training paramaters are then configured, the model trained and evaluation conducted on the validation set and the unseen test-set with the use of the COCOEvalutator class.

As with the YOLOv8 script, predominantly default values were used for the hyperparameters. With models that require changing hyperparameters through a configuration file often "Max Iterations" is used instead of "Epochs". Max iterations refers to the total number of iterations, or updates, the training algorithm will make during the learning process. In the context of Mask R-CNN, an iteration is a single update of the model's weights, usually performed once per batch of data. Mathematically, the total number of iterations is given by:

$$\text{Max Iterations} = \frac{N}{B} \times E \tag{3.1}$$

where N is the total number of samples, B is the batch size, and E is the total number of epochs.

The Mask R-CNN training script uses a batch size of 2, max iteration of 3000 and there are 118 images in the training dataset after augmentations have been added. This would equate to 50 epochs.

3.5 Depth Model Training

To test the original hypothesis that depth information would improve the accuracy of an object detection model the depth information had to be incorporated into the dataset. Originally it was assumed that the models architecture could be changed to accept 4 channel RGB-D data instead of RGB, initial investigation however proved that this was a more substantial task than first thought and that it was beyond the scope of this projects time constraints. A solution however was to replace the least used RGB colour channel with the depth data. By utilising an online image inspector and inspecting the brown shiitake mushrooms it was found that the least used colour channel was blue.

3.5.1 Depth Frame Matching and Depth Integration

The images in the dataset had been significantly reduced since the initially data collection phase so it was essential to create a program that can sort through the depth channel images and match them to the relevant RGB images. It did this by looking at the image name for each image in the dataset downloaded from Roboflow and matching them based on the name of the image.

With the images matched it was important to check the alignment of the depth data with with colour data. It was initially assumed that the annotations done on the colour images could be superimposed onto the depth data, allowing for depth annotations without need for additional time spend annotating. However on the Microsoft Azure DK camera the colour and depth information gets collected from two separate cameras spaced with the centres of each camera separated by 32mm. Using data from a different viewpoint could mean that the annotations for one would not accurately correspond to the other.

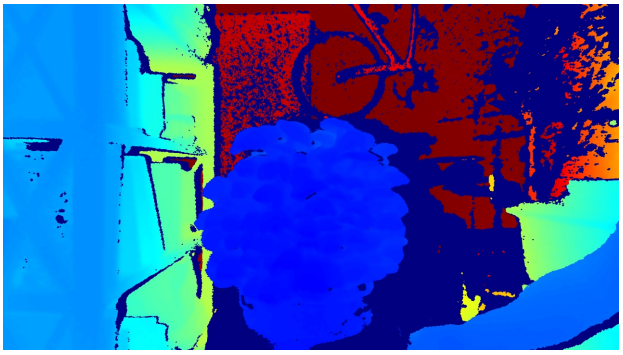
A python script had to first be used to convert the depth image into a human viewable form. When the depth channel gets separated from the .mkv file it gets saved as single-channel 16-bit unsigned integer which appears black to a human without conversion.

OpenCV [60] could be used to read and process the 16-bit depth images, applying a normalisation to transform the pixel values into a format that could be visually understood by human viewers. This conversion was essential to verify the integrity and accuracy of the depth data, ensuring that no information was lost or distorted during the extraction process from the .mkv file. Figure 3.6 shows the colour frame next to the depth image converted from the single-channel depth frame.

The image clearly matches and has a good level of variation in colour which represents the depth.

These two images can be blended to approximately check for alignment as shown in Fig. 3.7. Looking at the image the depth output and the colour output appear to line up sufficiently for the requirements of this project.

(a) Example RGB image used for checking the camera alignment



(b) Depth image converted from a single-channel depth frame

Figure 3.6: Corresponding RGB and Depth image after conversion

3.5.2 RG-D Training

With the alignment verified, the blue channel in the original RGB image can be switched out for the single-channel depth image to create the new dataset. The new version of the dataset was an exact replica of the dataset used in the original training but with the blue and depth channels swapped. These could then be uploaded again to Roboflow along with the original annotations for further testing.

By using Roboflow, training on the RG-D data can be achieved with a near identical training script. With a different download code the the new dataset can be downloaded to the runtime and the training run with the same parameters as before.

Figure 3.7: The result of blending the RGB and Depth images to check for alignment

3.6 Depth Reconstruction

After the initial comparison between colour and depth-assisted colour an investigation into other ways that the depth data could be utilised was conducted. It was thought that with enough data, additional information about the individual mushrooms shape and position could be gained from the combination of the mushroom segmentation results, the depth information and a point cloud reconstruction. Should the segmentation of the mushrooms be sufficiently accurate and the depth information be at sufficient quality then by reconstructing the mushroom you may be able to estimate parameters such as size, orientation, ripeness or weight.

3.6.1 Mushroom Segmentation and Point Cloud Reconstruction

Segmentation of the mushrooms could be accomplished with the already trained Detectron2 Mask R-CNN. The trained model was used to segment the shiitakes and the output colour inference mask could then be used to segment the instances of the mushrooms in the depth images. The depth image mask is then saved as its own file for further manipulation.

Manipulation of the depth data could be achieved with the Python library Open3D [61]. The library enables the user to read the raw depth information and construct a point cloud for visualisation of the object. An example is shown in Figure 3.8.

Figure 3.8: The result of mushroom segmentation and point cloud reconstruction

3.7 Cut-Point Estimation

With the crops detected and localised, the next step for a successful harvest is determining a suitable cut-point by which the robotic manipulator will look to cut along to remove the mushroom from the fruiting block. As discussed in section 2.5.2, there are several methods by which this can be achieved. Given the morphology of the shiitake mushroom and the fruiting block, key point detection was decided to be the most appropriate technique.

3.7.1 Keypoint Annotation

Keypoint annotations are annotated on an image with individual points. The most common use for keypoint detection is to detect the pose of a human body, as such it is common that these key points can be joined together to form a simplified skeleton. For our aim of defining suitable cut-points on mushroom stems a reasonable approach is to annotate two points either side of the base of a stem labelled 'Cutpoint' instead of the 17 points that commonly make up the a human skeleton.

To annotate keypoints, a new dataset was needed to be created. Due to the time limitations of the projects it was decided that this dataset would be smaller than that of the segmentation. The annotation software used for the segmentation annotations, Roboflow, does not have functionality to annotate key points so a new online software was used. CVAT (Computer Vision Annotation Tool) [62] was decided to be suitable for the task.

With a trained instance segmentation model the initial aim was to use the segmented mushrooms for the keypoint detection like with the point-cloud reconstruction. After an initial trial this was proven to be inefficient and hard to annotate. As the majority of mushrooms only have the cap clearly visible, and that with the image quality after inference it was often hard to choose suitable cut-points without the context of the surrounding fruiting block.

Cut-points would be annotated on the whole shiitake mushroom fruiting block. As the dataset would be small, only images from below the fruiting block would be used so that more of the mushrooms stems would be visible.

On CVAT a 2 point skeleton was defined which would be the cut-point annotation. Deciding rules this annotation task was more difficult. The majority of the mushrooms - even those with stems showing - were occluded by other mushrooms. Additionally as this was a proof of concept dataset the rules regarding the annotations were more relaxed, any clear stem would have a cut-point annotated across it at its lowest viewable point. This was decided with the assumption that the harvest operation would likely go from the bottom up, so any occluded mushrooms would become visible as the harvest progressed making the lowest viewable point the base where the cut should occur. Figure 4.3 shows an example of the annotations on CVAT, enlarged for clarity.

The total dataset size was only 18 images, in that images 344 cut-points where annotated. The dataset annotations were downloaded in the COCO keypoint 1.0 format, matched with the images and then split into a suitable training and test split. With the limited number of images a 80/20 train test split was deemed appropriate, resulting in 14 training images consisting of 272 cut-point instances and 4 test images.

Like the instance segmentation, both YOLOv8 and Detectron 2 boast state of the art performance in pose estimation tasks so these models would be implemented on this small dataset. Both datasets have been pre-trained on the MS COCO keypoint dataset [59].

Figure 3.9: An example of the cut-point dataset with 2 point keypoint annotations

3.7.2 YOLOv8-Pose

To use YOLOv8-Pose, the dataset's annotation files needed to be converted into the Ultralytics YOLO format. For this, each image has its corresponding text file containing the necessary annotations. A conversion function, as recommended by Ultralytics' documentation, was employed to change the COCO format .json file to the required YOLO format. Additionally, a simple YAML configuration file was created to specify various parameters and settings for the YOLOv8-Pose model [17].

The YOLOv8-Pose training script, like the segmentation, is simple and can be completed with a few lines of code. The program first installs the libraries, defines the model and starts training. As before, the hyperparameters were largely kept to their default values. YOLO generates visualisations that show both images and annotations from the dataset, to verify they have been loaded.

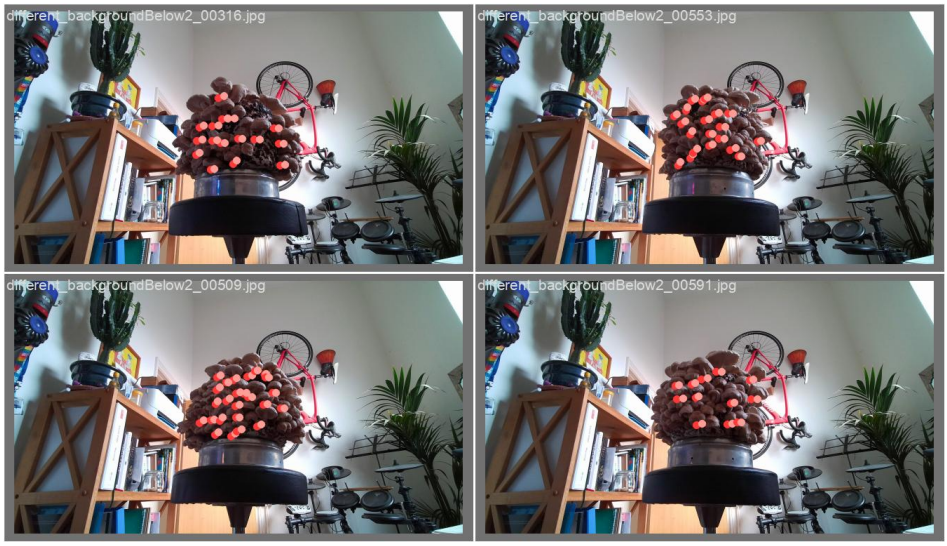


Figure 3.10: Verification images of the dataset and annotations

3.7.3 Detectron2

Training for keypoint detection with Detectron 2 follows a similar schedule to the instance segmentation. The set-up is more complex than that of the YOLO model, requiring object instantiating and dataset registration.

Firstly, the environment is set up to facilitate the project workflow. and Detectron 2 is downloaded and configured. The cut-point dataset, which is accessed through Google Drive rather than Roboflow, must then be registered into distinct train and test sets. These are then visualised to verify that the image data and annotation format are both correct and readable. After this verification, the training hyperparameters, such as batch size and number of iterations are configured. The model is then trained using the DefaultTrainer Class. Once the model is trained, it undergoes an evaluation on the 4 test images. Finally, predictions are made on the test set, serving as a qualitative evaluation of the model's overall performance.

The hyperparameters used in for this model were 4000 training intervals which is equivalent to 571 epochs with a training set of 14 images, a batch size of 2 which a learning rate scheduler increasing the learning rate from 0 to 0.00025 over the first 1000 iterations.

Figure 3.11: Verification image of the dataset and annotations in COCO format

3.8 Evaluation Metrics

To assess the effectiveness of the models it is important to have standard metrics too allow direct comparison which will give a measure of how well the model performs during inference. There are many evaluation metrics, and depending on the model type and the output of the model some may be more relevant than others. For the first parts of the project where YOLO and Detectron were compared and the RGB and RG-D networks were evaluated it was important to train and test the networks on the same dataset. The train-val-test split was maintained and the same images where used for each set.

Understanding these metrics requires starting from basic concepts and expanding them into something more useful. The models in this project output both bounding boxes and masks, the evaluation strategies for both are largely the same, just with the segmentation modelled on an individual pixel level

- **True Positive (TP):** The model has made a correct detection of a ground truth object
- **False Positive (FP):** The model has incorrectly made a detection of an object in the image,

either identifying a non-existent object of by misplacing an existing object.

- **False Negative (FN):** The model in has failed to recognise and detect an object that is located in the image

To better understand what is classed as a correct detection, there must be a way to evaluate the positioning of the bounding box with reference to the ground truth bounding box of the object. Intersection over union is commonly used for this decision [63].

3.8.1 Intersection Over Union

Intersection over union (IoU) scores how well the models predicted output matches the position of the actual ground truth of the object. As the name suggests it calculates this value by dividing the union of the two bounding boxes or masks by the area of the intersect.

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \tag{3.2}$$

Figure 3.12: Visual representations of the union and intersection of two bounding boxes

The conventional threshold for a correct detection is 0.5 or 50%, in other words, 50% or more of the prediction overlaps with the ground truth.

3.8.2 Precision and Recall

Most Object detection assessment metrics are derived from the previously discussed TP, FP, FN and IoU which are used to calculate the Precision, P and the Recall, R of a model.

Precision

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} = \frac{\text{True Positives}}{\text{All Detections}} \tag{3.3}$$

Precision focuses on the models success in identifying the relevant objects in the image by taking the number of correct predictions and the number of total predictions to form a percentage of correct positives.

Recall

Recall determines how well the model can detect the relevant object within the image.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} = \frac{\text{True Positives}}{\text{All Ground Truths}} \tag{3.4}$$

Equation 3.4 shows recall to be the ratio between correct positive predictions and all of the ground truths bounding boxes or masks.

Individually these metrics may not be a good judge of model performance. A model may have a low number of false positives, leading to a high precision, but might also miss many true positives, resulting in low Recall [63]. Conversely, a model with high recall might achieve this by being over predicting, thus increasing both true positives and false positives and consequently reducing precision. Therefore, it's often useful to consider both precision and recall to get a more encompassing view of a model's performance. [63]

F1-Score

A common combine both Precision and Recall into a single metric is to use the F1-Score, which is the harmonic mean of Precision and Recall.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.5}$$

The F1-Score will be high only when both precision and recall are high, offering a balance between the two. It's particularly useful when you have an uneven class distribution and gives more context than evaluating Precision or Recall individually.

3.8.3 Average Precision

To further understand the performance of a model, precision and recall can be used to calculate the Average Precision, AP [63]. The precision and recall can be plotted against each other to form the precision-recall curve. AP is essentially the area under the precision-recall curve with a high area under the curve (AUC) indicating both high precision and recall and can be used as a metric for the total performance of the model.

$$\text{Average Precision} = \int_0^1 p(r)\,dr \tag{3.6}$$

Equation 3.6 shows calculation for AP by calculating the area under the precision-recall curve. $p(r)$is the precision at a given level of recall r.

Mean Average Precision

An extension of Average precision is Mean Average Precision (mAP). This is likely the single most used performance metric when training deep learning networks.

Mean Average Precision (mAP) is an extension of Average Precision (AP) that takes into account multiple object classes in the evaluation. When dealing with object detection or segmentation tasks involving multiple classes, it becomes crucial to assess the model's performance for each class separately. mAP accomplishes this by averaging the AP calculated for each individual class.

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i \tag{3.7}$$

mAP is particularly useful when the dataset has class imbalance or when different classes have different levels of importance for the task at hand. A high mAP value indicates that the model performs well across all classes, providing a comprehensive view of the model's capabilities.

By default, training logs will often specify mAP as the performance metric throughout each epoch in training. For this project as there is only 1 class AP (shittake mushroom) so mAP and AP are equivalent terms.

COCO Metrics

For more complex tasks, the Common Objects in Context (COCO) [59] metrics are often used. These metrics extend mAP by evaluating performance at multiple IoU thresholds and provide a more detailed view of the model's robustness and accuracy. The COCO metrics include:

- **AP@0.50:** Average Precision at 50% IoU
- **AP@0.75:** Average Precision at 75% IoU
- **AP^S, AP^M, AP^L:** Average Precision for small, medium, and large objects
- **AP@0.50:0.95:** An average of the average precision at multiple IoU thresholds ranging from 0.50 to 0.95 in 0.05 increments.

The COCO evaluation metrics provide extra information into how well the model is performing on object at different levels of intersection at different sizes.

4 Results and Discussion

4.1 Colour Instance Segmentation

The first aim of this project is to test the suitability for deep learning instance segmentation techniques to segment the shiitake mushrooms growing on the fruiting block. Both models had been previously trained on the Microsoft COCO [59] dataset and where trained for the same number of epochs with the same batch size. Both YOLO and Detectron were able to successfully train on the new shiitake dataset and were able to optimise their network weights towards a high level of accuracy. Some of the evaluations metrics discussed (in section 3.8) have been tabulated in Table 4.1 after training.

Table 4.1: Comparison between Detectron 2 Mask R-CNN and YOLOv8m-seg

Metric	Model	
	Detectron 2 Mask R-CNN	YOLOv8m-seg
AP@[IoU=0.50:0.95]	55.5	67.9
AP@[IoU=0.50]	77.7	94.9
Total Inference Time (ms)	420.5	60.2
Pure Compute Time (ms)	77.5	6.0

The evaluation results show that by the accuracy metrics discussed, the YOLOv8 model appears to perform significantly better than that of the Mask R-CNN. Inference examples of each training set are shown in Figures 4.1 and 4.2.

Examining the qualitative aspect of the evaluation shows the Detectron to be sufficiently accurate at segmenting the shiitake mushrooms. Under close scrutiny, the Mask R-CNN does appear make a few mistakes and miss a few of the harder mushrooms, but the majority of mushrooms look to be accurately segmented.

(a) YOLOv8m-seg

(b) Detectron2

Figure 4.1: Comparison of YOLOv8-seg and Detectron2 inference

(a) YOLOv8m-seg

(b) Detectron2

Figure 4.2: Another comparison of inference on the same image with YOLOv8 and Detectron2

Figure 4.3: A example of Detectron2 inference showing both the instance segmentation and bounding box output

For the remainder of the segmentation aspect of this project, Detectron 2's Mask R-CNN has been chosen. Although the quantitative evaluation metrics presented in Table 4.1 indicate a higher accuracy for YOLOv8, the qualitative analysis revealed that both models are comparably effective for the task at hand.

Detectron2's Mask R-CNN was ultimately selected for its additional flexibility and documentation which is anticipated to be advantageous for further processing and experimentation later in the project

It should be noted that the YOLOv8 model demonstrated superior speed and accuracy in our quantitative analyses. As such, YOLOv8 still remained a strong candidate for future phases of this project.

4.2 RG-D Instance Segmentation

To test the hypothesis in section 3.1, the 2D and the the RG-D dataset created using the same images, where compared to establish if depth yielded any advantage. Table 4.2 shows accuracy scores for both dataset versions. The Mask R-CNN performance on both datasets was comparable but the accuracy on the RGB dataset was slightly higher across all metrics. Figure 4.4 shows an example of the detection and segmentation on the RG-D image. The lack of blue channel gives the whole image a yellow tint but the model still performs well but no clear advantage has been gained from introducing the depth to the data.

Figure 4.4: Detectron 2 Mask R-CNN inference on an RG-D image

Table 4.2: Evaluation Metrics for Bounding Box and Segmentation

	AP	**AP@.50**	**AP@.75**	**AR**
Bounding Box (RGB)	52.442	77.379	58.416	57.400
Bounding Box (RG-D)	51.41	76.259	59.554	56.300
Segmentation (RGB)	55.446	77.653	64.246	59.700
Segmentation (RG-D)	53.653	75.596	63.339	58.100

4.3 Cut Point Estimation

Like with the segmentation, the YOLOv8-seg model was trained first due to its relative ease of implementation. The model predicts an output bounding box and keypoint locations but initially the model do not look to be learning the keypoints. After 150 epochs the mAP for both the bounding box and keypoints was both zero and examining prediction images showed that the model was not making any predictons at all. Eventually really low mAP value started to show which gradually improved over the remaining training time.

Table 4.3: Keypoint detection results for each model on the validation set

	Metric	AP50	AP50:95
YOLOv8-pose	Bounding box	8.29	0.02
	Keypoint	0.44	0.26
Detectron2	Bounding box	14.22	0.02
	Keypoint	0.40	0.00

The result was not expected to be very good. The dataset only consisted of 18 images and the actual keypoint locations are not as obvious as they are with other keypoint applications such as features on a face.

Table 4.3 shows the results from running the trained models on the validation set of 4 images and Figure 4.5 and 4.6 show the results of the inference for visual evaluation. The models have not detected many of the potential cut-points but the predictions they have made appear to be in the correct location at the base of a mushroom stem.

Utilising Detectron 2 for the keypoint detection immediately looked to be more successful. Within 30/4000 training iterations the model looked to be learning and accuracy results were improving steadily throughout training. Looking at Figure 4.6, the Detectron model appears to have outperformed YOLO. Detectron has made more predictions per image and they mostly appear to be located at the base of the mushroom. Looking at the results available during evaluation, both models looked to have performed poorly but overall the Detectron has a higher accuracy score.

Both of these models have proven that current computer vision are be capable at plotting a cut and that the technique of annotating every cut within an image is a reasonable approach. With more training data and further optimisation it can be expected that the model should perform a lot better.

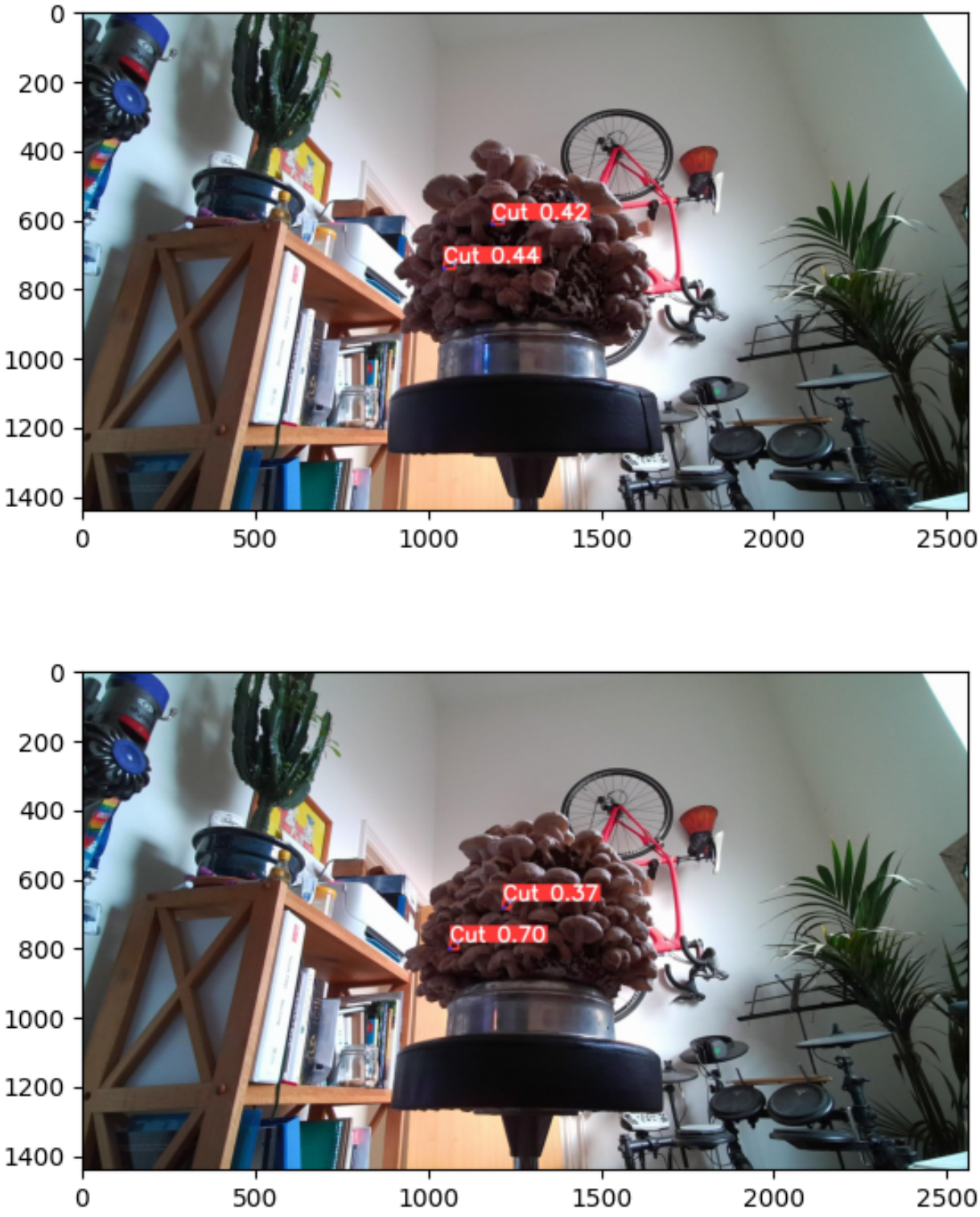


Figure 4.5: Example cutpoint predictions from the trained YOLOv8-pose keypoint detector

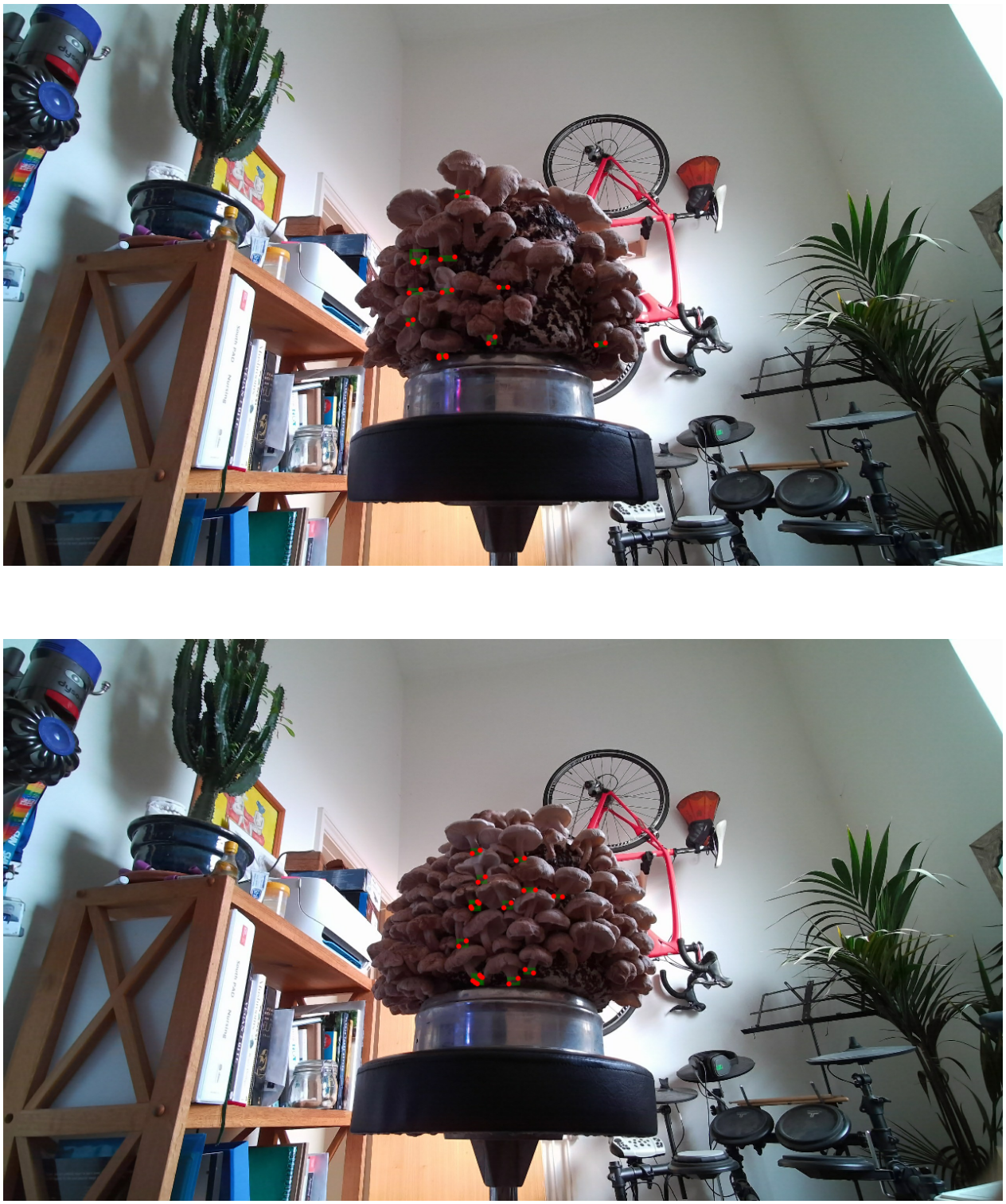


Figure 4.6: Example cutpoint predictions from the trained Detectron 2 keypoint detector

4.4 Discussion

4.4.1 Colour Instance Segmentation

YOLO has justifiable received considerable attention in recent years. Within a short space of time it has undergone multiple significant iterations establishing it often as the go-to solution for object detection tasks. Its high accuracy and ease of implementation has democratised access to computer vision to a wider group of developers. With these recent releases and popularity it was expected to outperform the accuracy of the Detectron2 Mask R-CNN in instance segmentation. While it did perform well and showed higher average precision values, visual evaluation of the inference images show that Detectron2 has comparable accuracy at segmenting shiitake mushrooms given the requirements of this project. Given the length of time Detectron has been released, the more substantial documentation and extra flexibility and customisation it was deemed more appropriate to stay with Detectron2. Given a more time restricted requirement, YOLO may well prove the better option. More testing and comparison between these two models with different GPU and system configurations would be needed to more accurately determine which was most suitable for the task.

4.4.2 RGB vs RG-D

When planning the project it was thought that due to the cluttered and confusing nature of the shiitake mushroom fruiting blocks that the colour-only instance segmentation would do far worse than it actually did. With the success of the first part of the project project it was difficult to imagine that the depth could add much more accuracy performance. Ideally this experiment would have looked at the comparison between RGB and RGB-D, showing a true representation of the advantage that depth may add to the accuracy of the object detector. After initial research this proved to be more complicated to implement than first thought and given the RGB accuracy and the likely increase in inference time with a more complex network it was decided to switch out the blue channel instead.

With the model performing so close to the RGB it is likely that it was learning depth features as in the RGB the blue values are low but not zero.

4.4.3 Depth Reconstruction

Although not included in the results section, using the instance segmentation to segment part of the image and then reconstruct it into a 3D representation proved successful. The results proved that the colour instance segmentation is accurate enough for use as part of a pipeline whereby the results from one aspect feed into another. Some effort was put into determining if the 3D reconstruction could be utilised to gain any further information about the crop such as the pose and maturity of the mushroom. These are likely possible to implement, but with the limited time available and lack of younger shiitake mushrooms it was decided to move forward to the final stage of the project. The fruiting blocks used for data collection grew a lot more densely packed than had been anticipated from the prior research. This meant that the mushrooms were often partially or mostly occluded by other mushrooms, in these situations the segmented cut-out of the mushroom and the 3D point cloud reconstruction wasn't clear and didn't make sense without the context of the rest of the fruiting block.

4.4.4 Cut Point Estimation

From the start it was unclear whether keypoint detection for a cut point would work at all. The literature related to keypoint detection is almost solely focused around the keypoints on a human such as a facial or skeletal features. As such there are fewer resources available to advice how best to approach the problem. Initially it was thought that the best approach would be to take individual mushrooms that had been segmented by the trained Detectron2 model, annotate and train a cut point model one mushroom at a time. As stated in the previous paragraph, due to the high level of occlusion between mushrooms this was hard to implement.

The second approach was where every visible possible cut point in the whole original image is annotated; this proved more suitable. There is limitations to this method however, most of the mushrooms do not have the stem visible and the stems are often unclear to annotate, either because of the colour, a shadow or because they are not situated centrally to the view of the camera.

The results showed that there is promise for this method to work well. Even with only 14 training images, both models could make a reasonable prediction and the Detectron keypoint detector made several good estimates for the cut locations, with both a bounding box over the area and a

keypoint either side of the stem. Judging by this success it is probable that with a larger dataset, the results would be significantly improved.

Comparing the results between Detectron2 and YOLOv8, the Detectron2 keypoint detector is likely more fine tuned than the YOLOv8 counterpart. YOLOv8 itself was only released in January 2023, with the pose/keypoint extension not release until April 2023. The inbuilt YOLO evaluation did not seem a true representation of the lack of success, the precision and recall metrics seemed suspiciously high. Given the proof-of-concept nature of this dataset investigation into this will be saved for future work with a more substantial dataset.

5 Conclusion

The objectives of this project where as follows:

- Create a dataset of shiitake mushrooms using a 3D camera
- Train an instance segmentation model to detect the shiitake mushrooms on the fruiting block
- Compare results using 2D and depth datasets to establish any advantage
- Create a separate dataset of the mushroom stems and make a cut-point estimation program to determine suitable cut-points.

Upon completion of this project these objectives have all been satisfied in exploration of how computer vision can be utilised in the process of autonomously harvesting shiitake mushrooms. A dataset of 92 3D images and $\tilde{5}800$ instances was collected and annotated in 2D to be used for the comparison for RGB and RG-Depth models. Two state-of-the-art instance segmentation models where trained to better understand the potential of computer vision for shiitake harvest. Both models achieved high accuracies with mAP@50% 77.7% for Detectron2 and 94.5% for YOLOv8. Given the relatively low number of training images those accuracy figures prove the utility of these architectures in a robotic harvesting system.

A short comparison between 3-channel colour images, or 2-channel colour + 1 channel depth was conducted to determine if any advantage. The results were comparable with the regular colour images performing slightly better.

A proof of concept extension to the instance segmentation was implemented with the keypoint estimation for cut point planning. Both Detectron2 and YOLOv8 managed to choose some suitable cut locations despite the training dataset only including 14 image. Detectron2, despite displaying low numerical accuracies looked to have successfully predicted multiple suitable cut locations of the visible mushroom stems.

5.1 Future Work

This project has successfully proven that computer vision can play an important part in the robotic harvest of shiitake mushrooms. However there have been a few limitations that would have affected the results predominantly the limited data, but also limited time spend optimising the current results. For these reasons their are the following suggestions for future work:

- More data should be collected and annotated, with different fruiting blocks, at different angles and in different environments in an attempt to make a system more generalisable to change.
- The tested segmentation models, Detectron2 and YOLOv8-seg should be adapted to accept a 4-channel RGB-D input instead of the default RGB. By customising these state-of-the-art models, a true evaluation of any advantages can be conducted and the costs, such as time and computing power, can be better estimated for practical implementation.
- The dataset should be migrated away from Roboflow and keypoint annotations should be added to the whole dataset. With Detectron this would would merge the segmentation and cut point models together. On inference, the model should be able to segmented the mushrooms and determine a suitable cut point in one go, utilising the multiple output capability.
- Hyperparameter tuning should take place so that the model can be fully optimised for a given processing and data limitation.
- Apply these techniques to other crops that grow in clusters such as: other mushrooms varieties, beans or berries.
- Apply a tracking algorithm so that the mushroom yield on a block can be properly estimated. Currently the model takes the input frame by frame without any continuity between frames, with a tracking algorithm every mushroom could be counted with a 360 degree spin.

References

[1] T. Rowland, *Shiitake Harvest*, Sep. 2023. available from: https://github.com/trowland/MSc-Dissertation-Shiitake-Harvest.

[2] United Nations, Department of Economic and Social Affairs, Population Division, "World population prospects 2019: Highlights," United Nations, Tech. Rep., 2019. available from: https://population.un.org/wpp/Publications/Files/WPP2019_Highlights.pdf.

[3] M. van Dijk, T. Morley, M. L. Rau, and Y. Saghai, A meta-analysis of projected global food demand and population at risk of hunger for the period 2010–2050, *Nature Food*, vol. 2, no. 7 2021, pp. 494–501, 2021.

[4] *Appharvest*, https://www.appharvest.com/, Accessed: 2023-03-30, Feb. 2023.

[5] World Wildlife Fund and Tesco, "Driven to waste: The global impact of food loss and waste on farms," World Wildlife Fund, Tech. Rep., 2021. available from: https://wwfint.awsassets.panda.org/downloads/driven_to_waste_summary.pdf.

[6] World Wildlife Fund, "Hidden waste: The food waste in our fields," World Wildlife Fund, Tech. Rep., 2022. available from: https://www.wwf.org.uk/our-reports/hidden-waste.

[7] H. Zhou, X. Wang, W. Au, H. Kang, and C. Chen, Intelligent robots for fruit harvesting: Recent developments and future challenges, *Precision Agriculture*, vol. 23, no. 5 2022, pp. 1856–1907, 2022.

[8] P. Stamets, *Growing gourmet and medicinal mushrooms*. Ten speed press, 2011, ch. 21.

[9] M. H. Buettgenbach, *Explain like i'm five: Artificial neurons*, Nov. 2021. available from: https://towardsdatascience.com/explain-like-im-five-artificial-neurons-b7c475b56189.

[10] Z. Qin, F. Yu, C. Liu, and X. Chen, How convolutional neural network see the world-a survey of convolutional neural network visualization methods, *arXiv preprint arXiv:1804.11191* 2018, 2018.

[11] Mathworks. "Convolutional neural networks." Accessed: 04/09/2023. (), available from: https://uk.mathworks.com/discovery/convolutional-neural-network-matlab.html.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems*, vol. 25 2012, 2012.

[13] O. Russakovsky, J. Deng, H. Su, *et al.*, ImageNet Large Scale Visual Recognition Challenge, *International Journal of Computer Vision (IJCV)* [online], vol. 115, no. 3 2015, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

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

[15] J. Redmon and A. Farhadi, Yolov3: An incremental improvement, *arXiv preprint arXiv:1804.02767* 2018, 2018.

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

[17] G. Jocher, A. Chaurasia, and J. Qiu, *YOLO by Ultralytics*, version 8.0.0, Jan. 2023. available from: https://github.com/ultralytics/ultralytics.

[18] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1653–1660.

[19] S. P. Wasser, Shiitake (lentinus edodes), *Encyclopedia of dietary supplements* 2005, pp. 653–664, 2005.

[20] R. Kohsaka, M. Tomiyoshi, O. Saito, S. Hashimoto, and L. Mohammend, Interactions of knowledge systems in shiitake mushroom production: A case study on the noto peninsula, japan, *Journal of Forest Research* [online], vol. 20 5 Oct. 2015, pp. 453–463, 5 Oct. 2015, ISSN: 16107403. DOI: 10.1007/s10310-015-0491-4.

[21] *Shiitake Mushroom market, the brainy insights*, https://www.thebrainyinsights.com/report/shiitake-mushroom-market-12716, Accessed: 2023-03-20, Apr. 2022.

[22] A. M. Research, *Agritech market - global industry size, share, growth, trends, competitive analysis, and forecast*, https://www.adroitmarketresearch.com/industry-reports/agritech-market, Accessed: 2023-03-30, 2023.

[23] *Join the future of farming*. available from: https://root-ai.com/#intro.

[24] *Meet Sprout - Muddy Machines*, https://www.muddymachines.com/technology, Accessed: 2023-03-30, 2023.

[25] S. Birrell, J. Hughes, J. Y. Cai, and F. Iida, A field-tested robotic harvesting system for iceberg lettuce, *Journal of Field Robotics* [online], vol. 37 2 Mar. 2020, pp. 225–245, 2 Mar. 2020, ISSN: 15564967. DOI: 10.1002/rob.21888.

[26] AppHarvest. "Appharvest acquires agricultural robotics and artificial intelligence company root ai to increase efficiency." Accessed: 09/08/2023. (2021), available from: https://www.appharvest.com/press_release/appharvest-acquires-agricultural-robotics-and-artificial-intelligence-company-root-ai-to-increase-efficiency/.

[27] Agrobot. "Agrobot e-series - the first pre-commercial robotic harvesters for gently harvest strawberries." Accessed: 09/08/2023. (2023), available from: https://www.agrobot.com/e-series.

[28] F. Robotics. "Transforming agriculture with selective and autonomous harvesting robots." Accessed: 09/08/2023. (2023), available from: https://fieldworkrobotics.com/.

[29] A. Williams, *Company makes significant progress with raspberry harvesting robots - university of plymouth*, https://www.plymouth.ac.uk/news/company-makes-significant-progress-with-raspberry-harvesting-robots, Accessed: 2023-03-30, Jun. 2022.

[30] *Agrobot e-series, agrobot*, https://www.agrobot.com/e-series, Accessed: 2023-03-30, 2023.

[31] A. Gongal, S. Amatya, M. Karkee, Q. Zhang, and K. Lewis, Sensors and systems for fruit detection and localization: A review, *Computers and Electronics in Agriculture*, vol. 116 2015, pp. 8–19, 2015.

[32] R. Horaud, M. Hansard, G. Evangelidis, and C. Ménier, An overview of depth cameras and range scanners based on time-of-flight technologies, *Machine vision and applications*, vol. 27, no. 7 2016, pp. 1005–1020, 2016.

[33] L. Fu, F. Gao, J. Wu, R. Li, M. Karkee, and Q. Zhang, Application of consumer rgb-d cameras for fruit detection and localization in field: A critical review, *Computers and Electronics in Agriculture* [online], vol. 177 Oct. 2020, Oct. 2020, ISSN: 01681699. DOI: 10.1016/j.compag.2020.105687.

[34] A. Arefi, A. M. Motlagh, K. Mollazade, and R. F. Teimourlou, Recognition and localization of ripen tomato based on machine vision, *Australian Journal of Crop Science*, vol. 5, no. 10 2011, pp. 1144–1149, 2011.

[35] K. Kusumam, T. Krajník, S. Pearson, G. Cielniak, and T. Duckett, "Can you pick a broccoli? 3d-vision based detection and localisation of broccoli heads in the field," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 646–651. DOI: 10.1109/IROS.2016.7759121.

[36] S. Bargoti and J. Underwood, "Deep fruit detection in orchards," in *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 3626–3633.

[37] S. Tu, J. Pang, H. Liu, *et al.*, Passion fruit detection and counting based on multiple scale faster r-cnn using rgb-d images, *Precision Agriculture*, vol. 21 2020, pp. 1072–1091, 2020.

[38] X. Liu, D. Wang, Y. Li, X. Guan, and C. Qin, Detection of green asparagus using improved mask r-cnn for automatic harvesting, *Sensors* [online], vol. 22, no. 23 2022, 2022, ISSN: 1424-8220. DOI: 10.3390/s22239270. available from: https://www.mdpi.com/1424-8220/22/23/9270.

[39] R. Courtney and T. Mullinax, *Washington orchards host robotic arms race, good fruit grower*, https://www.goodfruit.com/washington-orchards-host-robotic-arms-race/, Accessed: 2023-03-30, Dec. 2019.

[40] J. Rong, G. Dai, and P. Wang, A peduncle detection method of tomato for autonomous harvesting, *Complex & Intelligent Systems* 2021, pp. 1–15, 2021.

[41] Q. Sun, X. Chai, Z. Zeng, G. Zhou, and T. Sun, Multi-level feature fusion for fruit bearing branch keypoint detection, *Computers and Electronics in Agriculture* [online], vol. 191 2021, p. 106 479, 2021, ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2021.106479. available from: https://www.sciencedirect.com/science/article/pii/S0168169921004968.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Dec. 2015, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[43] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Cham: Springer International Publishing, 2016, pp. 483–499, ISBN: 978-3-319-46484-8.

[44] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 466–481.

[45] *Bristol robotics laboratory - centre for machine vision*, https://www.bristolroboticslab.com/centre-for-machine-vision, Accessed: 29-03-2023, n.d.

[46] J. Reed and R. Tillett, Initial experiments in robotic mushroom harvesting, *Mechatronics*, vol. 4, no. 3 1994, pp. 265–279, 1994.

[47] N. L. Baisa and B. Al-Diri, Mushrooms detection, localization and 3d pose estimation using rgb-d sensor for robotic-picking applications, *arXiv preprint arXiv:2201.02837* 2022, 2022.

[48] J. Rong, P. Wang, Q. Yang, and F. Huang, A field-tested harvesting robot for oyster mushroom in greenhouse, *Agronomy* [online], vol. 11, no. 6 2021, 2021, ISSN: 2073-4395. DOI: 10.3390/agronomy11061210. available from: https://www.mdpi.com/2073-4395/11/6/1210.

[49] J. Deng, Y. Liu, and X. Xiao, Deep-learning-based wireless visual sensor system for shiitake mushroom sorting, *Sensors*, vol. 22, no. 12 2022, p. 4606, 2022.

[50] P. Cong, H. Feng, K. Lv, J. Zhou, and S. Li, Myolo: A lightweight fresh shiitake mushroom detection model based on yolov3, *Agriculture*, vol. 13, no. 2 2023, p. 392, 2023.

[51] Y. Wang, L. Yang, H. Chen, A. Hussain, C. Ma, and M. Al-gabri, "Mushroom-yolo: A deep learning algorithm for mushroom growth recognition based on improved yolov5 in agriculture 4.0," in *2022 IEEE 20th International Conference on Industrial Informatics (INDIN)*, IEEE, 2022, pp. 239–244.

[52] A. 3. T. Limited. "Xtion pro live." Accessed: 18/08/2023. (), available from: `http://xtionprolive.com/asus-3d-depth-camera/asus-xtion-pro-live`.

[53] Microsoft. "Azure kinect dk." Accessed: 18/08/2023. (), available from: `https://azure.microsoft.com/en-gb/products/kinect-dk`.

[54] OpenNI. "Openni 2 sdk binaries docs." Accessed: 18/08/2023. (), available from: `https://structure.io/openni`.

[55] Microsoft, *Azure kinect dk documentation*, `https://learn.microsoft.com/en-us/azure/kinect-dk/`, Accessed: 18/08/2023, 2022.

[56] S. Tomar, Converting video formats with ffmpeg, *Linux journal*, vol. 2006, no. 146 2006, p. 10, 2006.

[57] W. Liu, D. Anguelov, D. Erhan, *et al.*, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, 2016, pp. 21–37.

[58] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, *Detectron2*, `https://github.com/facebookresearch/detectron2`, 2019.

[59] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, Springer, 2014, pp. 740–755.

[60] *The opencv reference manual*, 2.4.9.0, Itseez, Apr. 2014.

[61] Q.-Y. Zhou, J. Park, and V. Koltun, Open3D: A modern library for 3D data processing, *arXiv:1801.09847* 2018, 2018.

[62] CVAT.ai Corporation, *Computer Vision Annotation Tool (CVAT)*, version 2.2.0, Sep. 2022. available from: `https://github.com/opencv/cvat`.

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