ISTITUTO ITALIANO DI TECNOLOGIA, AND UNIVERSITY OF GENOVA

PHD PROGRAM IN BIOENGINEERING AND ROBOTICS

Robotic Grapevine Winter Pruning through Vision and AI

by

Miguel Ivo Ferreira Fernandes

Thesis submitted for the degree of *Doctor of Philosophy* (36° cycle)

December 2023

Prof. Darwin Caldwell	Supervisor
Dr. Claudio Semini	Supervisor

Thesis Reviewers:

Prof. Adriano Mancini, *Università Politecnica delle Marche*	External examiner
Prof. Joseph Davidson, *Oregon State University*	External examiner

Dibris

Department of Informatics, Bioengineering, Robotics and Systems Engineering

I would like to dedicate this thesis to my mother, for all the sacrifices she made so I could reach where I am right now.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Miguel Ivo Ferreira Fernandes
February 2024

Acknowledgements

I want to express my sincere gratitude to the many individuals who have played a crucial role, both active and passive, in completing this doctoral thesis. Their support, guidance, and encouragement have been invaluable throughout this challenging yet rewarding journey. In no specific order, I would like to thank:

Darwin Caldwell and Claudio Semini for their supervision, guidance, and support in developing the Vinum project.

Fei Chen, Carlo Rizzardo, Antonello Scaldaferri, Tao Teng, and Donatien Delehelle from *Active Perception and Robot Interactive Learning* of *IIT's Advanced Robotics* for their help in the initial phases of the system development.

Giuseppe Fiameni from *NVIDIA AI Technology Center* for the support provided in dataset creation and neural network training.

Juan Gamba from *IIT's Dynamic Legged Systems* research line for the support provided with the Summit XL, Arm motion planning, neural network output filtering and 3d point cloud creation.

Chundri Boelens, Angelo Bratta, Ylenia Nisticò, and Giovanni Dessy from *IIT's Dynamic Legged Systems* research line for all the support related to HyQReal, Summit XL, and mission organization.

Matteo Gatti, Paolo Guadagna, Alessandro Santamaria, and Filippo Graziosi from *Dipartimento di Scienze delle produzioni vegetali sostenibili* from *Università Cattolica Sacro Cuore di Piacenza* for all the agronomic knowlegde and support provided to this thesis.

Alessio Del Bue, Matteo Taiana, and Yonas Teodros Tefera for their help and inspiration in dealing with 3D point clouds.

Luís Alexandre for the mentorship provided during my bachelor's degree, master's degree, and initial steps in my research life.

Last but not least, I thank my closest friends, Carla, Valentin, Nuno, Anabela, Sep, and Louise, for their friendship and support over the most difficult year of my life.

This thesis is the culmination of the collective efforts of all those mentioned above. Without the support of all these persons, it would be nigh on impossible to achieve the results presented and discussed throughout this thesis.

Abstract

Grapevine winter pruning is a labor-intensive and repetitive process that significantly influences the quality and quantity of the grape harvest and produced wine. Because of its complexity and repetitive nature, the task demands skilled labor that needs to be trained, as in many other agricultural sectors. This thesis encompasses two developed approaches that target using a robotic system to perform grapevine winter pruning, using a vision system and artificial intelligence. The initial phase focuses on the 2D approach, harnessing the power of 2D RGBD images with segmentation neural networks. These networks enable the identification and classification of vital plant structures. The thesis details the post-processing steps applied to the 2D results. These procedures encompass mask merging to simplify outputs alongside error detection mechanisms to eliminate small, erroneous segments. The second phase of the thesis ventures into the third dimension, introducing point clouds. In the 3D approach, depth images and segmentation masks obtained in the 2D approach converge to create instance-segmented point clouds. The study does not only delineate the integration of 2D and 3D methods but also scrutinizes their efficacy in pruning point identification. The practical implications are evaluated through real-world performance analyses. The real-world performance was analyzed via the utilization of the created system in two different pruning seasons, one during the winter of 2021/2022 and the other in the winter of 2022/2023, where the system was used in a simulated vineyard to prune a set of test plants. Moreover, the thesis underscores a unique facet of adaptability, presenting a customizable framework that empowers end-users to fine-tune parameters according to their specific requisites. This adaptability extends to variables such as the number of nodes to retain on pruned canes and the preferred cane thickness, encapsulating the versatility of the 3D approach.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivations

Figure 1.1 Example of a manual pruning and mechanical pre-pruning. Manual pruning is still needed after the mechanical pre-pruning. (Images obtained from Ochs, 2021 and Betty, 2010)

An important task to perform in a vineyard is winter pruning, a complex operation that needs to be completed throughout the dormant season (Poni, Tombesi, et al., 2016). A balanced winter pruning allows a good compromise between remunerative yield and desired grape quality, maximizing the grower's income (Intrieri and Poni, 1995; Poni, Gatti, et al., 2018). This selective operation requires about 80 to 120 man-hours per hectare annually. Table 1.1 shows an overview of the man-hours needed for several tasks on the vineyard. Due

Table 1.1 Labor demand for the annual vineyard management for two different pruning techniques. Guyot* refers to Guyot + Mechanical Harvesting, and Spur pruned cordon** corresponds to Spur pruned cordon + Mechanical harvesting + Pre-pruning. The targeted type of plant for this thesis, highlighted in bold, is the spur pruned cordon, and winter pruning is the second most expensive task, with the first being harvesting. (Table from Palliotti et al., 2011)

	Guyot	Guyot*	**Spur pruned cordon**	Spur pruned cordon**
Fertilization	1	1	1	1
Weeding	2	2	2	2
Mowing	2-3	2-3	2-3	2-3
Plant protection (spray)	10-15	10-15	10-15	10-15
Desuckering	5	5	5	5
Shoot Thinning	30	30	50	50
Shoot Positioning	2-4	2-4	2-4	2-4
Trimming	2-4	2-4	2-4	2-4
Leaf Removal	-	-	5	5
Harvest	140	2-4	150	2-4
Winter Pruning (Including Cane Stripping)	100	100	**80**	-
Pre-pruning followed by hand-finishing	-	-	-	25
Cane Positioning	20	20	-	-
Total	324	188	319	118

to increased skill shortages and limited labor availability in the agricultural sector, automating winter pruning is a crucial step to increase efficiency and reduce production costs.

Havesting, although not the focus of this thesis, is the most expensive task in the needed man-hours when done manually. There are advantages and disadvantages to mechanical harvesting, as noted by Gilby, 2018, where the main advantages are the reduced time spent on harvesting and the capability of the machine to run 24 hours a day. The main disadvantages are machine access difficulty due to steep slopes, damage caused to the plants and the grapes, and the fact that machines tend to leave the stems on the vines, which limits winemaking choices.

Figure 1.2 Before and after mechanical pre-pruning, Manual pruning is still needed after the mechanical pre-pruning. (Images obtained from Betty, 2010)

Figure 1.1 shows an example of manual pruning and machine pruning. According to Betty, 2010, mechanical pre-pruning has some issues, where the mechanical pruner has instances where it damages the trellis wire that holds the grapevines in place or damages the T-posts that provide support to the grapevines. A before and after of the mechanical pruning can be seen in Fig. 1.2.

This project is included in the collaboration between Istituto Italiano di Tecnologia (IIT) and the Universitá Cattolica del Sacro Cuore, Piacenza (UCSC). The project VINUM has been addressing the automation of spur pruned cordon winter pruning of grapevines for the last few years. The VINUM team of IIT, which includes me as one of the members, is working on the robotics side of the project, as well as the vision, which is documented in this thesis. The members from UCSC provided their expertise in annotating data for dataset creation, evaluating the performance of the trained neural networks models and backbones in the field, evaluating the pruning points obtained in the 2022/2023 pruning season, and providing agronomic knowledge over the project.

1.2 Context of the Study

Grapevine winter pruning is an essential agricultural practice that significantly impacts crop yield and quality. The process involves removing growing points from grapevines in their dormant season, which helps to optimize photosynthesis and improve fruit production. However, manual, labor-intensive pruning can be time-consuming and costly, especially for large-scale vineyards. To address this challenge, researchers have been exploring the potential of robotic systems to perform autonomous grapevine winter pruning. This thesis presents a study on developing such a system using computer vision techniques and artificial intelligence algorithms. In the initial phase, the proposed approach involves segmenting grapevines from RGBD images using deep learning-based neural networks, which are then used to generate pruning points for each cane. This later evolves into using multiple views of segmentations to create 3D point clouds that see the grapevine from various angles to mimic the human behavior of examining an object from multiple angles. The results show that the developed system can examine the grapevine, identify pruning points, and prune the generated points. Moreover, the platform-agnostic framework allows end-users to customize various parameters according to their specific requirements, making it a versatile solution for different grapevine varieties and growing conditions.

The structure of a pruning region can be seen in Fig. 1.3, along with the correct pruning points and with the identified grapevine organs. Each class corresponds to a different visible organ over the winter pruning. The grapevine has several organs, the *cordon*, also called the permanent cordon, which is the oldest structure of the grapevine plant. Connected to the cordon, we may have at least three-year-old wood, commonly called an *arm*. Two-year-old wood is the *Spur*, and one-year-old structure is a *cane*, which generally appears vertically. The *cane* has multiple instances of *Nodes*, the points where future canes may grow. In the context of a pruning region, the *basal cane* is the *cane* closest to the *cordon*.

The selection of pruning points for grapevine plants is based on agronomic knowledge and expertise, aiming to maximize the quality and quantity in the next season and throughout the plant's life.

The pruning points can be categorized into four main types: clean cut, base-bud cut, replacement cut, and spur cut. The base-bud and spur cuts involve making two physical cuts on the grapevine. A description of the cuts can be read in Table 1.2.

A clean cut permanently removes the pruning region and is typically used for downward-growing canes. The base-bud cut is performed above the base bud and removes the old spur.

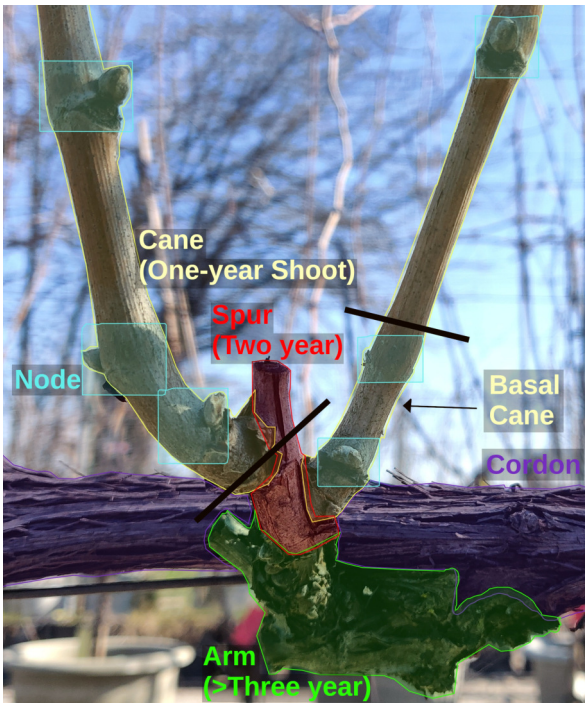

Figure 1.3 Semi-schematic representation of a grapevine pruning region area, illustrating the five grapevine organs: Cordon, Arm, Spur, Cane, and Node. The black lines indicate the desired pruning points. The basal cane is the cane closest to the cordon. The color overlay for each organ is created with data annotation created by agronomic experts. (Image captured at Università Cattolica del Sacro Cuore in Piacenza.)

The replacement cut involves removing the old spur and arm of the pruning region while retaining the replacement cane up to the second count bud.

The most commonly used pruning technique is the Spur cut, where the first cut is placed above the Nth count bud, and the second cut removes the old spur. The number of count buds to be kept is a parameter that can be adjusted depending on the variety of grapevine.

Table 1.2 Types of pruning cuts.

Cut type	Description
Clean cut	The corresponding organ is completely removed.
Base-bud cut	The base-buds are kept only, while the remaining (upper) part is removed.
Spur cut	The first *N* nodes of the basal cane are left, while the rest of it is removed. The *spur parent* organ is cut above the basal cane if it is not a new pruning region.
Replacement cut	The first *N* nodes of the basal cane are left, while the rest of it is removed. The *arm parent* organ is cut above the basal cane.

1.3 Objectives and Contributions

This thesis aims to develop a visual processing system to identify agronomically correct grapevine pruning points. For the duration of this work, several contributions have been made to the scientific community in the form of publications, events, and materials created.

The contributions of this thesis are as follows:

- Dataset for grapevine organ segmentation.
- Grapevine plant graph model for plant analysis.
- Segmented 3D reconstruction via point clouds of grapevines.
- Implementation of customizable agronomic rules for pruning point generation on 2D and 3D plant graph models.

1.3.1 Publication Record

Journal papers

- Guadagna*, P., **Fernandes*, M.**, Chen, F., Santamaria, A., Teng, T., Frioni, T., Caldwell, D. G., Poni, S., Semini, C., and Gatti, M. (2023). Using deep learning for pruning region detection and plant organ segmentation in dormant spur-pruned grapevines. Precision Agriculture, 24(4):1547-1569 (https://doi.org/10.1007/s11119-023-10006-y) (* equal contribution).

Conference papers

- **Fernandes, M.**, Scaldaferri, A., Fiameni, G., Teng, T., Gatti, M., Poni, S., Semini, C., Caldwell, D., and Chen, F. (2021). Grapevine winter pruning automation: On potential pruning points detection through 2d plant modeling using grapevine segmentation. In 11th IEEE International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (IEEE-CYBER 2021) (https://doi.org/10.1109/CYBER53097.2021.9588303) (Best Student Paper Award).

- Teng, T., **Fernandes, M.**, Gatti, M., Poni, S., Semini, C., Caldwell, D., and Chen, F. (2021). Whole-body control on non-holonomic mobile manipulation for grapevine winter pruning automation. In IEEE International Conference on Advanced Robotics and Mechatronics (ICARM 2021) (https://doi.org/10.1109/ICARM52023.2021.9536083) (Best Conference Paper Award Finalist).

- Guadagna, P., Frioni, T., Chen, F., Delmonte Incerti, A., Teng, T., **Fernandes, M.**, Scaldaferri, A., Semini, C., Poni, S., and Gatti, M. (2021). Fine-tuning and testing of a pre-trained dnn for detecting pruning regions of spur-pruned grapevines. In European Conference on Precision Agriculture (ECPA 2021) (https://doi.org/10.3920/978-90-8686-916-9_16).

- Milburn, L., Gamba, J., **Fernandes, M.**, and Semini, C. (2023). Computer-vision based real time waypoint generation for autonomous vineyard navigation with quadruped robots. In 2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC 2023) (https://doi.org/10.1109/ICARSC58346.2023.10129563).

Manuscript under preparation

- **Fernandes, M.**, Gamba, J., Graziosi, F., Caldwell, D. G., Gatti, M., Semini, C. Integrating 2D Segmentation with 3D Point Clouds for Pruning Point Generation in Grapevine Winter Pruning. Computers and Electronics in Agriculture, 2024 (under preparation).

1.3.2 Other Activities & contributions.

- Inauguration of the UCSC-IIT joint robotics lab, located at the Università Cattolica del Sacro Cuore in Piacenza, Italy in March 2021.

- Dataset creation and publication, as presented on (Guadagna, Fernandes, et al., 2023), and available at https://zenodo.org/record/5501784.
- Experimental trials with pruning robot prototypes over two pruning seasons, winter 2021/2022 and winter 2022/2023.
- Robot demos to the public during the 2023 United Nations FAO Science and Innovation Forum in Rome, Italy.

1.4 Overview of the Thesis

The chapters are organized as follows: Chapter 2 presents the related work, exploring neural networks, point clouds, and applications of these concepts in agriculture. Chapter 3 describes the system's first phase of development, using a prototype approach that delineates the essential components for the vision system, Chapter 4 presents the evolutions made to the system, replacing the simple pruning point approach with modules that attempt to output agronomically correct pruning points. Chapter 5 shows the implementation of three-dimensional point clouds into the system and pruning results. Chapter 6 presents the created material related to arm manipulation and motion planning, along with the system execution pipeline. Finally, Chapter 7 presents the conclusions regarding the developed systems and a final summary of the thesis and achieved results.

Chapter 2

State of the Art

Summary

Considering the research project of developing a system that uses vision to understand the structure of a grapevine, understanding the current state of the art in several domains is needed. The chosen fields for investigations were also split into three categories. The first category (Section 2.1) introduces neural networks, explaining backbones, models and frameworks, and some applications that are not related to agriculture. The second category (Section 2.2) focuses on 3D point clouds, and the third category (Section 2.3) contains applications of these methods to agriculture via fruit detection, yield estimation, weed removal, plant datasets, and pruning point generation. It is essential to state that all the images present in this chapter are obtained from the corresponding papers.

2.1 Neural Networks

A quick introduction to neural networks is given in this section by explaining important components, such as backbones, models, frameworks, and tasks. These concepts are important to understand both the rest of this chapter, as well as the thesis.

2.1.1 Backbones

A backbone in neural networks refers to the core architecture of neural network models, and they are situated normally after the input layer. These backbones are used by models such as Mask R-CNN (He, Gkioxari, et al., 2017) or YOLO (You Only Look Once) (Redmon and Farhadi, 2018), with different backbones allowing tradeoffs between accuracy and

computational efficiency. For example, ResNet-50 corresponds to the ResNet backbone with 50 layers, and ResNet-101 corresponds to the ResNet backbone with 101 layers. It is common to have a number after the backbone name, corresponding to the specific depth or number of layers in the model. By layers, ResNet counts all the layers introduced, such as convolutional layers, batch normalization, activation functions, and skip connections, with other backbones possibly having slight differences in their depth counting method.

ResNet (He, Zhang, et al., 2016), short for Residual Network, is an artificial neural network backbone designed to address the challenge of training deep neural networks. The main innovation of ResNet is the use of residual blocks. In traditional neural networks, each layer transforms the input data into a new representation. However, as the network gets more profound, it becomes harder to train because of issues like vanishing gradients or degradation, where adding more layers decreases accuracy. To mitigate the vanishing gradient problem, ResNet introduces skip connections, also known as shortcut connections, which allow the network to skip one or more layers in the course of training. These skip connections enable the flow of information directly from one layer to another, allowing the network to learn more effectively.

ResNeXt (Xie et al., 2017) is a deep neural network backbone that builds on top of ResNet. The term "ResNeXt" is derived from "Residual" and "Next." The key innovation in ResNeXt is the introduction of a "cardinality" parameter. The cardinality refers to the number of independent paths that information can flow through within a ResNeXt block. Traditional ResNet blocks use a simple stacking of layers, but ResNeXt introduces a split-transform-merge strategy. Instead of stacking many filters within a convolutional layer, the filters are divided into groups, and each group forms an independent path. Filters in the context of ResNext are the channels of the convolutional layer or the "feature map" of the convolutional layer.

2.1.2 Models

Mask R-CNN (He, Gkioxari, et al., 2017) is a region-based convolutional neural network architecture that extends the previous Fast/Faster R-CNN object detector. The critical innovation of Mask R-CNN is its ability to identify and locate objects in an image and generate a high-quality segmentation mask for each instance of the object. This means that, in addition to predicting bounding boxes around objects, Mask R-CNN can provide a pixel-level mask outlining the exact shape of each detected object.

YOLO (Redmon and Farhadi, 2018), which stands for "You Only Look Once," is a family of real-time object detection neural network models used in computer vision. The key idea behind YOLO is to perform object detection and classification in a single forward pass of the neural network, where this single pass can also be named single shot. Due to its single pass over the network, it becomes very efficient and suitable for real-time applications. Traditional object detection methods often involve sliding window approaches and region proposals, making them computationally expensive. YOLO simplifies this process by dividing the input image into a grid and directly predicting bounding boxes and class probabilities for each grid cell.

CenterMask (Wang et al., 2020) proposes a single shot object segmentation method to simultaneously predict object instances and pixel-wise segmentation masks for each instance in an image. The "center" in CenterMask refers to the model predicting a center point for each instance and then refining the segmentation mask around that center.

EOLO (Zeng and Sabah, 2020) proposes a single shot object segmentation method capable of being embedded in mobile devices. It achieves better inference using the same backbone as other methods, although with lower but acceptable accuracy.

A common thread for these models is that they perform object detection, where they find an object and draw a box around the found object. In images, pixel coordinates (X,Y) are used, where the starting point (0,0) is, by standard, placed on the top left of the image. The box placed by the neural network around the object, normally referred to as a bounding box, is generally defined by (X_0, Y_0, w, h), with X_0, Y_0 corresponding to the top left corner of the bounding box, and w, h to the width and height of the bounding box. The methods that also perform object segmentation provide the same bounding box information, along with a segmentation mask, usually referred to as just a mask. For a single object, a mask is defined by a matrix containing zeros and ones, the same size as the input image, where zero represents a pixel that does not belong to the object and one a pixel that belongs to the object.

2.1.3 Frameworks

A Framework is a library that provides a foundation for building, training, and deploying neural networks. Commonly, they provide all the essentials from data loaders to training scripts, configurations, and several pre-trained neural network baseline models, commonly referred to as a model zoo, along with result metrics for these models, for benchmark purposes. These metrics may include bounding box and mask Average Precision, train time

per iteration, inference time per image, and the required memory for the training and memory used for execution.

Detectron2 (Wu et al., 2019) is a framework from Facebook AI Research, containing implementations of state-of-the-art detection algorithms, using Pytorch and including a model zoo. It includes several models ready to use, such as Faster R-CNN, Mask R-CNN, RetinaNet, DensePose, Cascade R-CNN, Panoptic FPN, and TensorMask, ready to be used in tasks such as object detection, instance segmentation, panoptic segmentation and human pose prediction.

MMDetection (Chen, Wang, et al., 2019) is an open-source computer vision framework developed by the Multimedia Laboratory at the Chinese University of Hong Kong. It facilitates research and development in object detection, instance segmentation, and related tasks. The framework is implemented in PyTorch and provides a model zoo, tools, and utilities for building and training state-of-the-art object detection and segmentation models.

2.1.4 Tasks

The aforementioned neural networks are designed to execute certain tasks. In computer vision, the target tends to be the extraction of information from images, with these tasks being, for example, image classification, object detection, and image segmentation, among others. Image classification involves assigning a predefined label or category to an input image. The goal is to identify and categorize images into different classes accurately. For example, a model might predict that a given image contains a specific object or concept.

Object detection focuses on identifying and locating multiple objects within an image. Instead of assigning a single label to the entire image, the model must detect and classify individual objects while providing their spatial information. For instance, object detection aims to recognize and locate each person and vehicle in an image with people and cars.

Image segmentation entails dividing an image into distinct regions based on certain criteria, such as object boundaries or semantic content. Unlike object detection, segmentation provides a pixel-level understanding of the image. It outlines and differentiates various objects or areas within the image. For instance, in a medical image, segmentation could identify and delineate different organs or structures.

2.1.5 Applications

The authors in Casado García and Heras Vicente, 2018 present, in a guiding way, all the steps required to construct an object detector using deep learning techniques. They create a

set of Jupyter notebooks, which are interactive documents that allow sharing of live code, equations, visualizations, and narrative text. They can use several programming languages. In this case, Python programming language is used.

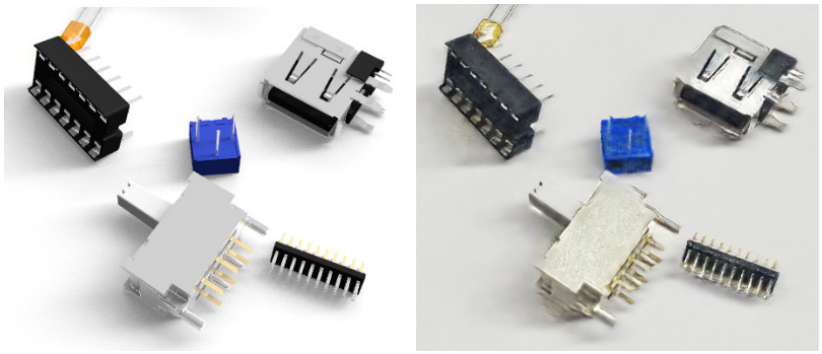

Figure 2.1 Refinement of synthetic images using generative adversarial Networks. The Left shows the input image and the right presents the refined image, where the components obtain a more realistic surface texture and color. (Obtained from Camaro Nogues et al., 2018)

The authors in Camaro Nogues et al., 2018 present the application of domain randomization and generative adversarial networks on Mask R-CNN for object detection. The main idea is that synthetic data can be refined by randomizing background textures, object textures, and object colors, along with refinement from the application of generative adversarial networks, seen in Fig. 2.1, which generates images closer to reality than the synthetic counterparts.

The authors in Yang and Soatto, 2020 use domain adaptation to adapt annotated images from a simulated dataset to a target "style" closer to reality. Domain adaptation in the context of semantic segmentation generally refers to adapting a model trained on a source domain to perform well on a target domain where the data distribution may differ. It uses Fast Fourier transformations on input images, replacing the low-level frequencies with those of the target images, as seen in Fig. 2.2. The authors report improvements in training several models with different backbones.

The authors in Mildenhall et al., 2020 present Neural Radiance Fields (NeRF), a system that aims to create a detailed and realistic 3D scene representation by training a neural network to model the radiance (color and intensity of light) at different points in a 3D space. Traditional methods for 3D scene representation often involve using meshes or voxels, but NeRF offers a continuous and neural network-based approach. The key idea behind NeRF is to represent a scene as a continuous function, where the neural network takes in 3D coordinates as input and outputs the radiance at those points. During training, the network learns from a set of images of the scene and corresponding camera poses, capturing the

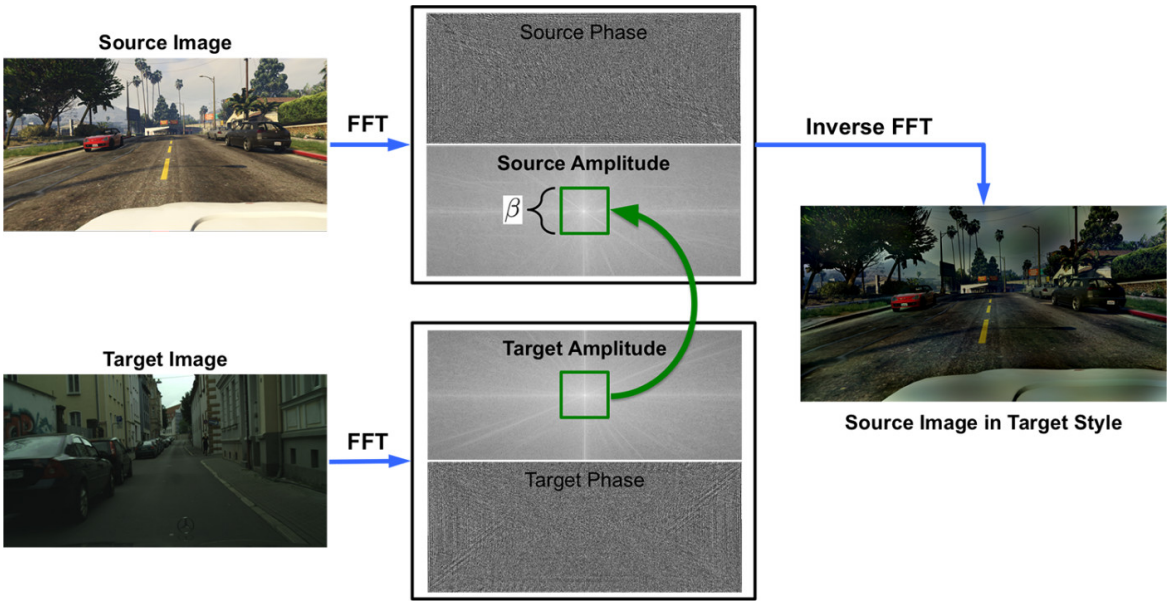

Figure 2.2 Pipeline for refinement of synthetic images using Fast Fourier Transforms. A randomly selected target image provides the style, and the source image replaces its style with the one from the target image. The resultant image has a smaller domain gap, resulting in better semantic segmentation model training. (Obtained from Yang and Soatto, 2020)

intricate details of the scene. NeRF has been applied to various applications, including 3D scene reconstruction, rendering, and virtual reality

The authors in Kirillov et al., 2023 present a new model and dataset for image segmentation. The model is designed and trained to be promptable, meaning it can transfer zero-shot to new image distributions and tasks. Zero-shot refers to a paradigm in which a model is trained to recognize and generalize to classes or tasks not seen during the training phase. In other words, the model can predict new, unseen classes without receiving explicit examples of those classes during training. The dataset was created in three stages. In the first stage, the annotations were manually created by a team of annotators who created segmentation masks without class labels. For the second stage, annotators receive annotated images and are requested to add segmentations for unannotated objects. The last stage is fully automated.

2.2 3D Point Clouds

One of the main issues in recognizing grapevine organs is the occlusion of Nodes at certain angles of vision. With this in mind, a natural step is the emulation of a human that tilts and moves its head to obtain a different angle over the grapevine. One way to merge the visual information from multiple points of view is via 3D point clouds.

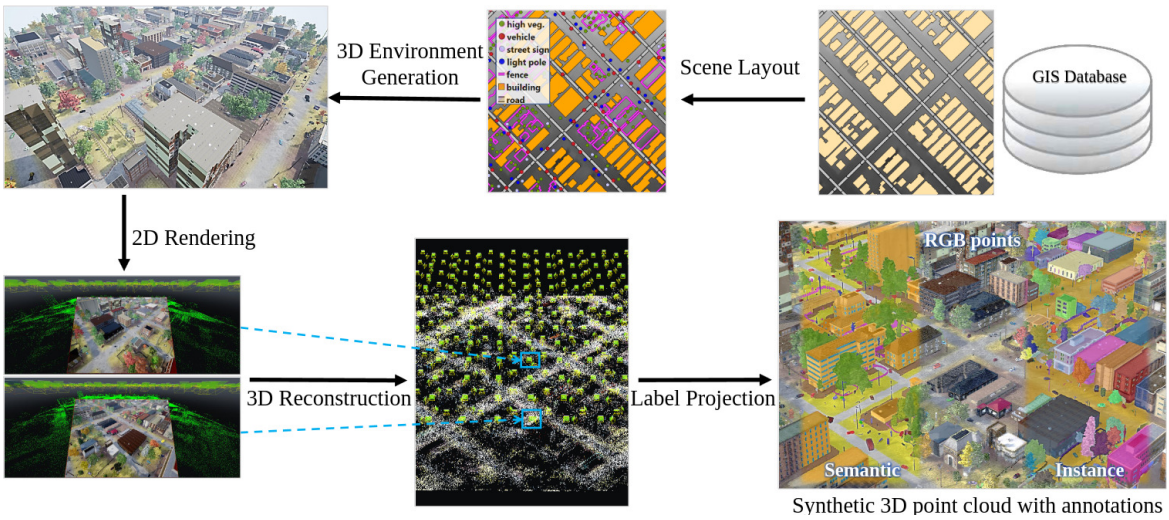

Figure 2.3 Proposed pipeline for the synthetic data generation. (Obtained from Chen, Hu, et al., 2022)

Regarding 3D point clouds, Chen, Hu, et al., 2022 presented large-scale aerial photogrammetry synthetic and natural datasets for semantic and instance segmentation. GIS data sources created realistic 3D virtual environments resembling city blocks, including building footprints and road networks. Objects like trees and vehicles were strategically placed for diversity and realism. Distance constraints ensured no intersections between objects. Trees were clustered to simulate forests, and individual trees were added around buildings. The study used a shape grammar for generating 3D buildings with varying types and heights for diverse environments. The proposed synthetic data generation pipeline is shown in Fig. 2.3.

3D-SIS (Hou et al., 2019) is a method for Semantic Instance Segmentation of RGB-D scans. It leverages multiple inputs of the same environment to reconstruct a 3D model. The process involves capturing a series of RGB-D frames, computing 6DoF rigid poses, and reconstructing the 3D model. The neural network employed in the approach combines features from both 2D color images and 3D geometry. It first detects and classifies objects and their classes using these combined features. Then, the network uses this information to detect instances inside each object's bounding box. This approach enables accurate and comprehensive semantic instance segmentation in 3D environments.

Softgroup++ (Vu et al., 2022) is a scalable method designed for 3D Instance Segmentation. It addresses semantic prediction challenges by allowing each point to belong to multiple classes. To enhance scalability, the authors identified the slowest part of the process as the k-NN grouping. To mitigate this limitation, Octree k-NN is utilized instead of the traditional "vanilla" k-NN approach. Figure 2.4 shows the results of the proposed system.

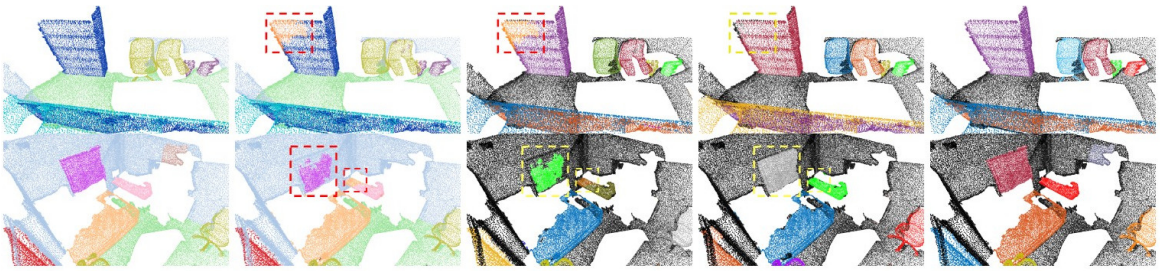

Figure 2.4 Results based on qualitative analysis of the ScanNet v2 validation set indicate that instance prediction without SoftGroup yields low-quality instance masks in areas with incorrect semantic predictions (identified by dashed boxes). In contrast, SoftGroup enhances the accuracy of instance masks in these regions. (Obtained from Vu et al., 2022)

2.3 Agriculture Applications

Some of these methods have also been applied to the agri-food field, such as fruit detection and segmentation (Borianne et al., 2019; Kuznichov et al., 2019; Santos et al., 2020) for yield estimation (Bargoti and Underwood, 2017), for weed removal (Di Cicco et al., 2017; Milioto et al., 2018), plant phenotyping (Grimm et al., 2018) and pruning (Botterill et al., 2017; Silwal et al., 2022).

2.3.1 Detection and Segmentation

Authors in Santos et al., 2020 present a new public dataset with grape clusters annotated in 300 images and a new annotation with interactive image segmentation to generate object masks, identifying background and occluding foreground pixels using scribbles. It has images, bounding boxes, and masks and an evaluation of two state-of-the-art methods for object detection, object segmentation, and a fruit counting methodology. This work provides a comparison between two current state-of-the-art methods, the MASK-RCNN, and YOLO versions 2 and 3, where the first method was tested for semantic segmentation, object detection, and instance segmentation, and both YOLO versions were tested only for object detection. In the semantic segmentation, only one class is considered: the grape. Each grape cluster is regarded as a rectangular bounding box in object detection. The instance segmentation follows the same logic as the object detection but uses areas and intersections of masks instead of bounding boxes. Figure 2.5 shows some detection results of the tested neural networks and the ground truth. In evaluating the methods, the authors consider that

Figure 2.5 Object detection results produced by the tested neural networks on an example of each grape variety, where the colors do not represent correspondence. (Obtained from Santos et al., 2020)

Figure 2.6 Left part of the figure shows the detection of mangoes in the "Keitt" tree, and the right part of the figure shows the three examples of cultivars targeted, where A corresponds to "Boucodiekhal", B corresponds to "Keitt" and C corresponds to "Kent". (Obtained from Borianne et al., 2019)

Mask RCNN presented superior results to the YOLO network. Still, they affirm that the bounding box annotation used to train the YOLO networks is faster to create.

The authors in (Borianne et al., 2019) studied the robustness of using the Fast R-CNN neural network in the identification of three different mango cultivars, the "Kent", the "Keitt" and the "Boucodiekhal" with a 90% score for fruit detection, but 56% for the cultivar identification, commenting that, although the network can identify the mangoes, it has issues differentiating their shape due to somewhat proximal in shape and color. Figure 2.6 contains an illustrative example of these mangoes, along with an example of the output of the neural network.

The authors in Kuznichov et al., 2019 present data augmentation techniques for leaf segmentation and counting. Data augmentation is a set of tools that can be applied to neural network training data, such as synthetic data generation and generative models to create

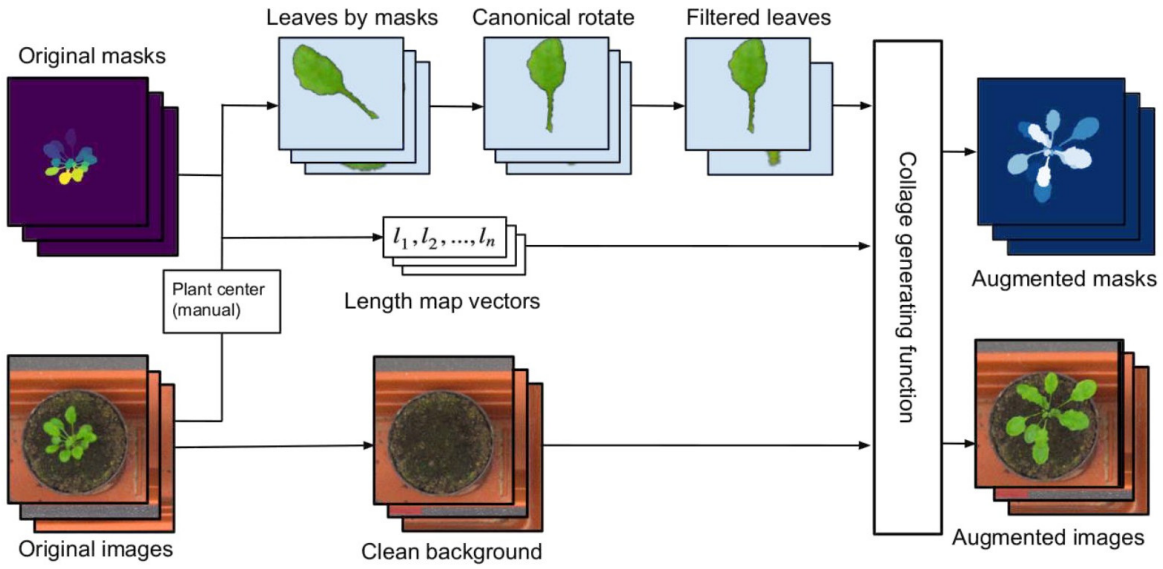

Figure 2.7 Structured collage generation pipeline. (Obtained from Borianne et al., 2019)

new data or the application of transformations like rotations, random cropping, and other operations over the training images. The authors propose a data augmentation method that attempts to keep the physical appearance of the generated objects as close as possible to the real captured plants. Figure 2.7 shows the proposed generative pipeline. This generative pipeline is achieved by extracting leaves from a dataset that look clear and focused. These leaves are then rotated to their canonical position and are filtered based on their distance to the plant center between other conditions. After that, clean background images of pots are created from the same dataset, where clean entails no leaves in the background image. Finally, a set of rules mimics a plant structure by merging the generated leaves and the created clean background images. An example of the rules is that the plant grows from a point close to the center of the background pot, with the leaves growing toward the perimeter of the pot.

2.3.2 Yield Estimation

Regarding yield estimation, the authors (Bargoti and Underwood, 2017) introduce an image processing framework that detects apples and estimates yield in apple orchards by performing image segmentation using a Multi-Layered Perceptron and a Convolutional Neural Network, which is later post-processed using Watershed Segmentation and Circular Hough Transform algorithms, and to count individual fruits. This method was tested on data captured by a monocular camera mounted on an autonomous ground vehicle, and it contains several apple

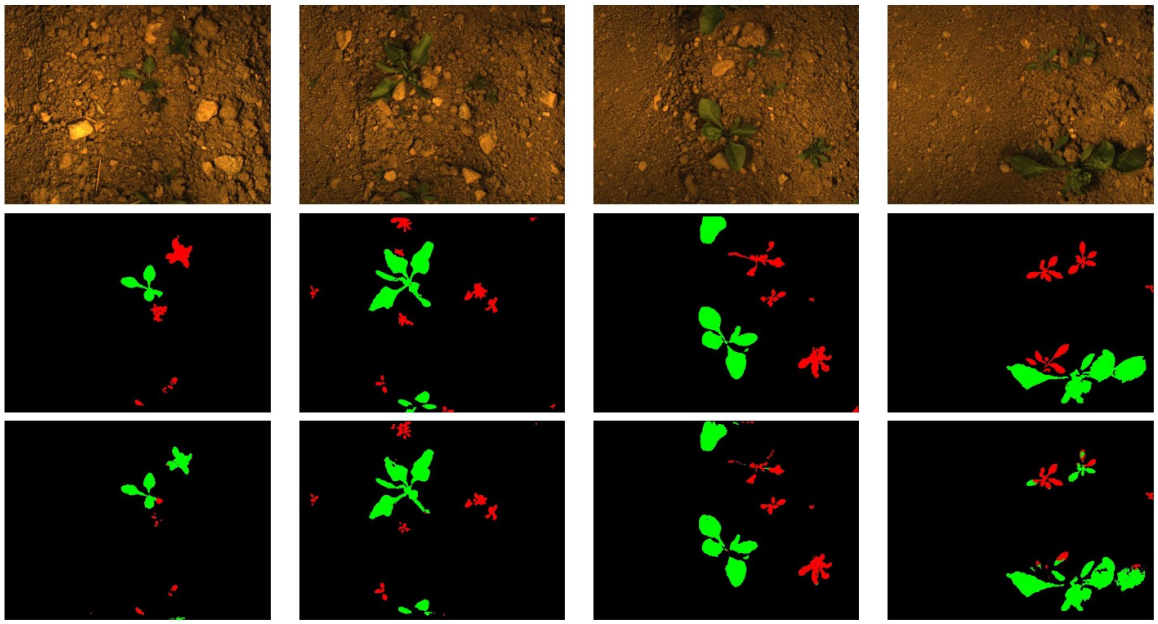

Figure 2.8 Examples of segmented images containing sugarbeets in green, and weeds in red. The first row is the input, the second is the ground truth, and the third is the segmentation results.(Obtained from Di Cicco et al., 2017)

varieties with different appearances. The fruit count is accumulated over individual orchard rows and compared to the post-harvest count.

The authors in Olenskyj et al., 2022 test and implement two different methods to predict grape yield estimation: object detection and regression models. This process starts by using a commercial yield monitor to create a large dataset of 23,581 yield points and 107,933 images in a mechanically managed commercial vineyard. The authors state that the object detection model shows poor performance when not considering the size of the grape clusters, and performance improves when using the grape cluster area instead of the cluster count. The regression-based deep learning model has a similar performance to the grape cluster in a variety of object detection methods, with the regression models not requiring data annotation to train and use.

2.3.3 Weed Removal

The authors in Di Cicco et al., 2017 present an approach to create synthetic data to segment weeds and sugar beet, minimizing the human effort needed to create datasets for training neural networks. Examples of the trained neural network, ground truth, and input can be seen in Fig. 2.8. The data is generated by modeling all the aspects without manually creating the

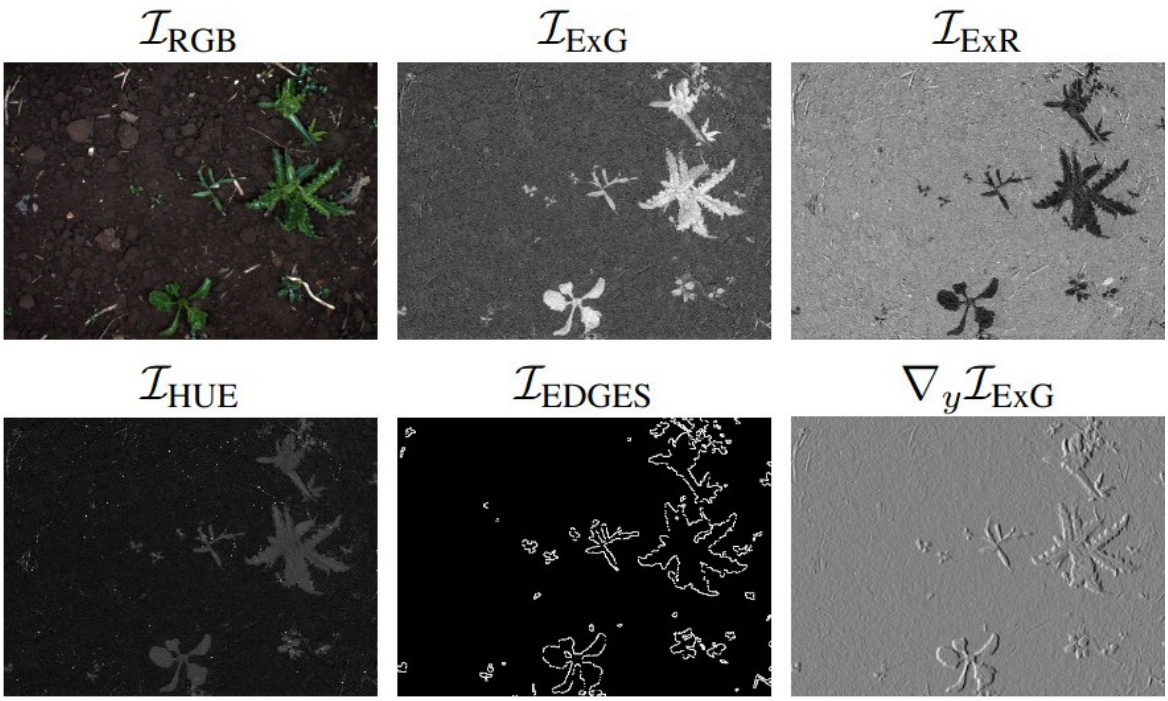

Figure 2.9 Some alternative representations are provided as input to the neural network. In order, on the first row, we have RGB image obtained by the camera, Execssive Green, Execssive Red, and on the second row Hue, Canny edge detector, and Sobel in the y direction on the Execssive Green. (Obtained from Milioto et al., 2018)

entire scene, along with randomizing and modulating parameters related to the generation pipeline, allowing a procedural dataset generation.

The authors in (Milioto et al., 2018) must deal with the challenge introduced by the variety of work regions in the fields, lighting, and weather conditions, which leads to difficulties in the semantic segmentation of crop fields. The authors present a convolutional neural network (CNN) to successfully allow a robot to distinguish sugar beet plants, weeds, and background dirt. This neural network, inspired by the SegNet and Enet, takes as input 14 different channels. These are the three color channels (Red,Green,Bue), Hue, valor and saturation from HSV color space. Four vegetation indices are added, including excess green, excess red, color index of vegetation extraction, and normalized difference index. Finally, operations on the excess green color index, such as the Sobel derivatives, Laplacian, and canny edge detector, Fig. 2.9 shows some of these layers. The input is resized to 512x384 pixels and has its contrast normalized channel-wise. The convolutional layers are defined as a composite function of a convolution followed by batch normalization and then rectified with a Rectified Linear Unit (ReLU). As a building block, a Residual Separable Bottleneck is proposed, in

which the convolution input is added to its result, helping with the degradation problem of very deep networks. To improve performance, this building block adds convolutions that reduce the depth of the input volume and then expand the result, reducing the number of FLOPs. The unpooling operations performed in the decoder are done by sharing the pooling indexes of the symmetrical pooling operation, allowing the network to have information on the spatial positions. Considering the output, the last layer is a linear operation followed by a softmax activation, where each element represents the probability of the pixel belonging to the class background, weed, or crop.

2.3.4 Plant Phenotyping

Authors in (Grimm et al., 2018) present a proof of concept for detecting and quantifying plant organs for yield estimation without using destructive means. This approach is based on automated detection, localization, count, and analysis of plant parts used to estimate yield, meaning the young shoots, inflorescences (plant part where flowers are formed), and grapes. Figure 2.10 illustrates the various plant parts considered. A CNN was created for the semantic segmentation and tested, along with object detecting and localization on six different datasets covering different grapevine growth stages. This neural network was composed of an encoder-decoder convolutional network, where the encoder uses VGG-Net-16 up to the fully connected layers, and the decoder uses FCN-8 (Fully Convolutional Network), replacing VGG-Net-16's fully connected layers. It receives the three or four color channels as input if the infrared is available. The decoder part uses the FCN-8 network, but uses concatenations instead of element-wise summations of layers, enabling end-to-end training. The final output is a 2-channel image, the size of the input, where the first channel contains the probability of an object in each pixel, and the second channel includes the likelihood of a background pixel.

The authors in (Esser et al., 2023) present a robotic system equipped with laser and camera sensors for high-resolution in-field plant scanning. The main contributions stated by the authors are the mobile robot shown in Fig. 2.11, methods to deal with outdoor lighting and localization, and comparison between laser scanner and camera. Both systems have positive and negative points regarding the precision of the laser scanner system versus the photo camera capture system. The laser reconstruction is more accurate but lacks color information and is less complete. The camera system has issues with reflective surfaces such as raindrops. In contrast, the laser system has issues with bright sunlight and plant movement due to wind, with the consideration that both systems complement each other.

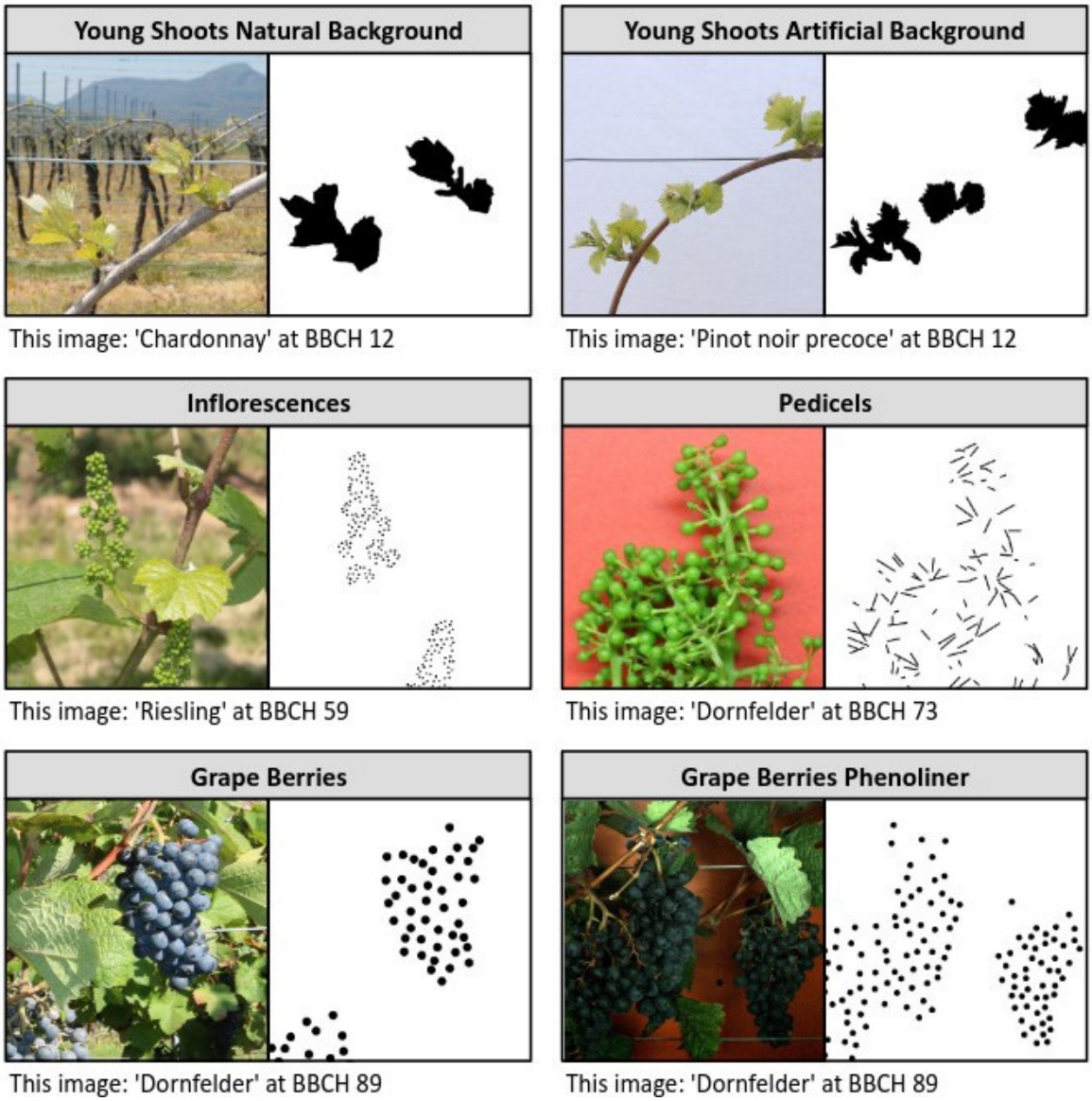

Figure 2.10 Examples of the various plant part datasets created, where in each, the left image corresponds to the input, and the right image corresponds to the annotation provided to the neural network to learn. (Obtained from Grimm et al., 2018)

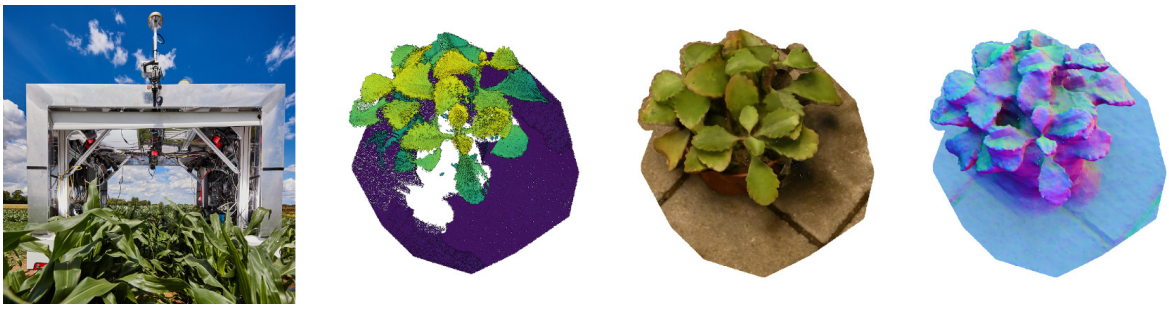

Figure 2.11 The first image presents the proposed mobile robot for 3D crop reconstruction, the second figure presents the laser-based reconstruction, and the third and fourth images present the reconstructed textured mesh. (Obtained from Esser et al., 2023)

2.3.5 Plant 3D Reconstruction

In Gentilhomme et al., 2023, the authors introduce ViNet, a network designed to identify nodes within a grapevine plant and reconstruct the plant's structure, where qualitative results can be seen in Fig. 2.12. The study also introduces the 3D2Cut dataset, offering annotated data containing node information and synthetic backgrounds. The approach employs a stacked hourglass network to reconstruct grapevine structure, encompassing node locations, branch types, and interconnections.

In Schneider et al., 2023, the authors introduce a novel skeletonization pipeline for grapevines. The authors propose a new skeletonization strategy that addresses the challenge of cycles in the structure graph. Furthermore, the paper showcases vigor estimation within vine growth by integrating 3D data and skeleton information. The platform used for data collection can be seen in Fig. 2.13.

The authors You et al., 2023 introduce a framework for the reconstruction of tree topology that can be used in tree pruning, using only RGB information in real-time. Using RGB information instead of point clouds, the authors avoid issues with thin branches that are not captured easily and can reach real-time performance. Figure 2.14 shows the system and example processing outputs, such as the 2D points of interest used to calculate the 3D points. The experimental validation demonstrates that the setup can generate primary branch models with an accuracy of 4-5 mm and secondary branch models with an orientation accuracy of 15 degrees relative to the ground truth model.

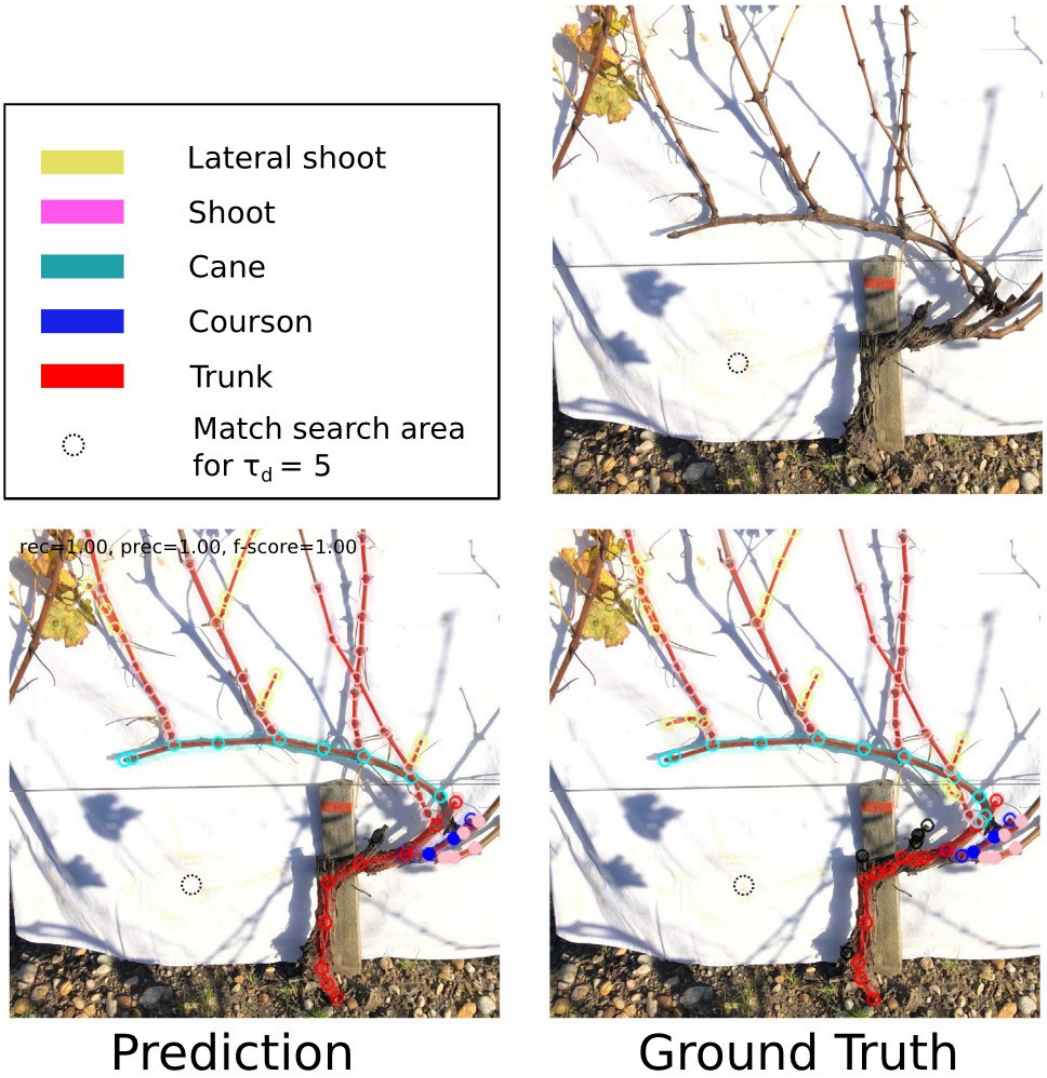

Figure 2.12 ViNet qualitative results, with the multiple organs identified. The top right presents the original image; the Bottom right is the ground truth; the Bottom left presents the predicted information. (Obtained from Gentilhomme et al., 2023)

Figure 2.13 (a): Data collection robot. (b) Point cloud with overlaid skeleton data. (c) Simple grapevine used for robotics research. (d) A complex example of vines in real situations. (Obtained from Schneider et al., 2023)

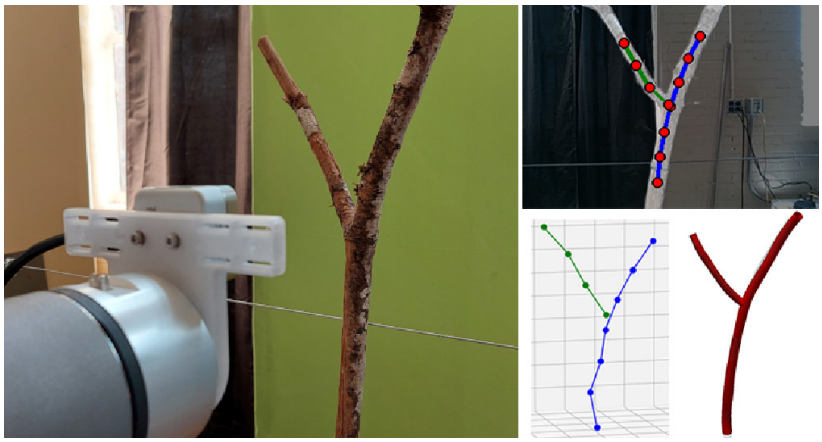

Figure 2.14 On the left, the robot along with the branch to be reconstructed, and on the top right, the extracted 2D points of interest, and bottom right presents the triangulation along with the branch 3D reconstruction. (Obtained from You et al., 2023)

2.4 Pruning Automation

2.4.1 Related Work

Authors in (Botterill et al., 2017) present a robotic system for automating the pruning of grapevines. Using a mobile platform, shown in Fig. 2.15, with trinocular stereo cameras that envelop the grapevine, the system captures images of the vines while a computer vision system constructs 3D models. An artificial intelligence system decides which canes to prune, and a six-degree-of-freedom robot arm executes the cuts. The integrated system is demonstrated in the vineyard, with the main contributions being the accurate 3D vine models

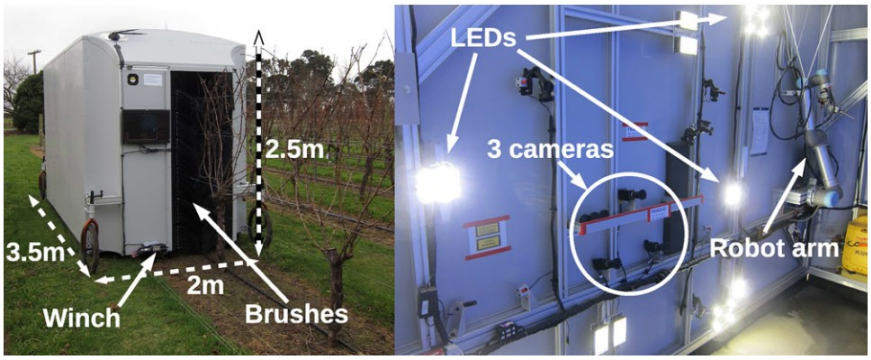

Figure 2.15 The proposed robotic system, seen from the outside and the inside. It is a platform that covers the grapevines and illuminates the interior, uniformizing the light for the vision system. (Obtained from Botterill et al., 2017)

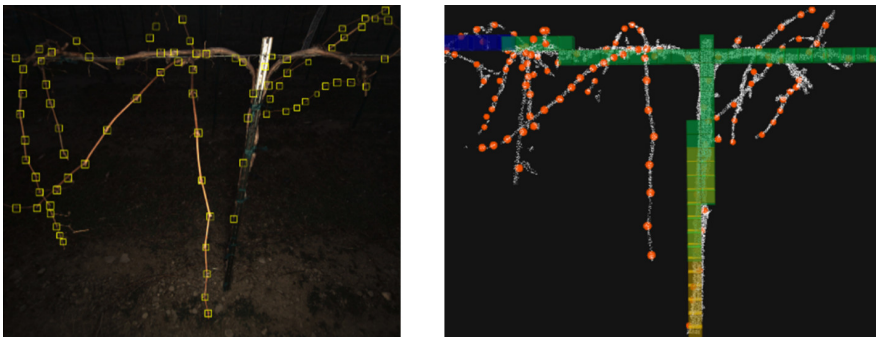

Figure 2.16 On the left, bud detection using Faster-RCNN. On the right, the squares represent the hard obstacles for motion planning, while the orange markers represent the detected buds. (Obtained from Silwal et al., 2022)

and the successful testing of the entire system. The system claims to have achieved a pruning time of 2 minutes per vine, comparable to human pruners.

Bumblebee (Silwal et al., 2022) is an autonomous robotic system that performs winter pruning, introducing novelty on several systems: camera, perception, manipulation, navigation, and robotic platform. The proposed system targets bilateral cordon grapevines after mechanical pre-pruning, as presented in Fig. 2.16, and prunes a grapevine in 213 seconds with a total pruning accuracy of 87%. The authors also mention lessons learned, such as the need for a secondary camera system to reduce occlusion, which brings such problems as the constant recalibration and manual tuning of parameters to achieve real-time point cloud processing.

Apart from these, there are other platforms available related to pruning made by companies to target vineyards. One of these commercial platforms is the *Vitis* mechanical platform from Billo (Billo, 2022) seen in Fig. 2.17 that allows the pruning people to sit in a heated

Figure 2.17 Vitis platform from Billo s.r.l. (Obtained from Billo, 2022)

Figure 2.18 On the left YV01 platform from Yanmar (Obtained from Yanmar, 2021). On the right Bakus robot from VitiBot (Obtained from VitiBot, 2022)

and illuminated vehicle that gets pulled by a tractor through the vineyard. Other examples of vineyard robots target spraying, weeding and other vineyard operations, e.g. YV01 of Yanmar (Yanmar, 2021), shown in Fig. 2.18 (left); as well as the Bakus robot of VitiBot (VitiBot, 2022) shown in Fig. 2.18 (right).

2.4.2 Our Robot Prototype Systems

The robot prototype systems documented throughout this thesis depend on a mobile manipulator to perform perception and pruning.

The mobile robot must meet several essential requirements to perform its tasks effectively:

- Weather-resistant
- The platform must be able to easily traverse challenging terrains.
- The platform has to include an arm for manipulation, which is crucial
- Stability

In detail, the mobile robot should be weather-resistant, ensuring its functionality is not compromised by environmental conditions. Additionally, the mobile robot must be able to

easily traverse challenging terrains, as found in vineyards. Furthermore, the mobile robot has to include an arm for manipulation, which is crucial, allowing the robot to interact with its surroundings and perform winter pruning. Finally, stability is paramount to prevent any interference with the correct functioning of the manipulation subsystem.

In the course of the development of the vision system, several mobile robots were used. The remainder of this subsection describes these robot prototypes.

Rolling Panda

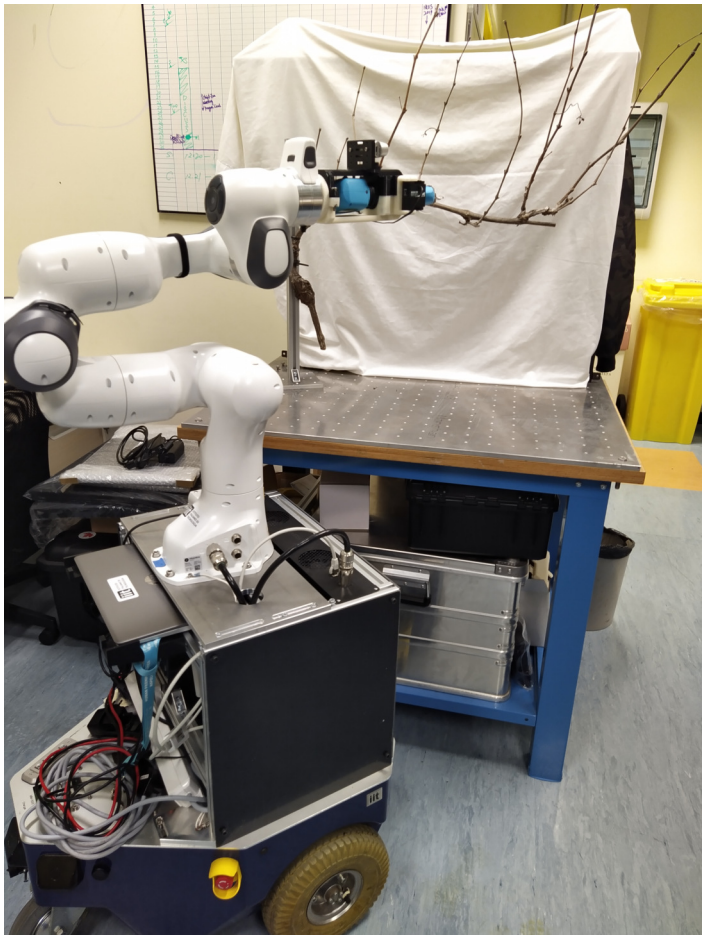

Figure 2.19 First Robot prototype used on the project, dubbed the *Rolling Panda*, composed by a Neobotix *MP-500* and a *Franka Emika Panda*, equipped with an Intel RealSense D435 camera for the vision and custom-made pruning shears that are activated via a servo motor.

The initial mobile manipulator robot used is APRIL Lab's Rolling Panda platform, seen in Fig. 2.19. This wheeled robot, composed of a Neobotix *MP-500* and a *Franka Emika Panda*, was the first robotic system, where both robots were separated on the software level,

meaning that the wheeled robot was controlled by its computer, inside the base, and a separate computer controlled the robotic arm. This was due to the old software running on the wheeled robot, which is running *ROS Indigo*, versus the computer controlling the arm that runs *ROS Kinetic*. Another issue with the wheeled robot is that it is meant for lab research since it is not weatherproof and has difficulty navigating on rough terrain such as a vineyard. I configured the wheeled robot to run *ROS navigation stack*, using *hector slam* (Kohlbrecher et al., 2011) as the SLAM method, using a *SICK LiDAR* mounted on the front of the robot, and *amcl* for localization on a map. Due to the different versions of ROS that are incompatible with each other, the adopted SLAM method was not used in the outdoor pruning trials, although it was tested.

The shears that this platform uses is a commercially bought electric pruning shears. To mount these shears on the *Panda* arm as an end effector, a 3D-printed structure was designed to hold the pruning shears and allow the mount of an Intel RealSense D435 camera for perception. The pruning shears are remotely operated using a servo motor that physically activates the trigger.

HyQReal

Figure 2.20 IIT's quadruped robot *HyQReal* (Semini et al., 2019) with Kinova Gen3 with seven DoF arm and pruning end effector.

The second mobile manipulator robot used was IIT's quadruped robot *HyQReal* with *Kinova Gen3 with seven DoF*. A new version of the pruning end effector was created, allowing better manipulation dexterity. The overall shears are shortened, and the shears are remotely triggered by electric signals instead of a physical servo. The two reasons for the change from the *Franka Emika Panda* to *Kinova Gen3* robotic arm are the following: The control box required by the *Panda*, which is considerably large and challenging to install onto mobile robots such as *HyQReal*, in addition, *Panda* arm's weight of 18 kilograms surpasses the 8 kilograms of the *Kinova Gen 3*. The second reason is related to software, where the *Panda* requires a dedicated computer that runs a real-time kernel, whereas the *Kinova Gen3* does not have such a requirement. In simple terms, a real-time kernel is a modification that allows predictable and low-latency response times for time-sensitive tasks, such as ensuring the deterministic execution of processes. This type of execution is needed for low-level control of a robotic arm. Still, it interferes with other hardware components, such as graphics cards, that have issues during real-time execution.

Summit XL

Figure 2.21 *Summit XL* with *Kinova Gen3 with seven DoF* and pruning end effector.

The third mobile manipulator robot used was *Summit XL* with *Kinova Gen3 with seven DoF*. This mobile manipulator is agile and more appropriate to outdoor environments in

comparison to the above-mentioned *Rolling Panda*. A recurring issue with the previous pruning end effectors was related to the calibration of the depth camera to the used robotic arm. To simplify the calibration process, a third version of the pruning end effector was created that simplifies the depth camera calibration by placing the camera in line with the arm end effector mount. Moving the shears underneath the end effector allows the tip of the shears to be closer to the arm end effector mount, which benefits the overall dexterity of the robot arm.

2.5 Conclusions

The related work presented in Section 2.3.5 and Section 2.4.1 are closely related to this thesis' end goal of automating the winter pruning of grapevines, although in one way or another, each suffers from some limitation. A common thread in Botterill et al., 2017; Gentilhomme et al., 2023 is that both require a controlled environment for the perception part of the system to be robust, be it by having a static background Gentilhomme et al., 2023 or by enveloping the entire plant with a structure that keeps out natural light, and illuminating with synthetic uniform light Botterill et al., 2017. To address this limitation, this thesis targets the creation of a dataset for segmentation and related neural networks, that are robust against the background, regardless of the environment where it operates.

Additionally, Botterill et al., 2017 uses AI to decide where to place the pruning points, which can be seen as a limitation since the pruning for a specific variety may not be the same for other varieties. To address this limitation, this thesis presents a plant model able to be used for deterministic analysis to generate the pruning points based on adjustable, parametrized rules. Our solution also addresses the need for "advanced sensing capabilities to assess cane health and vine size for balance pruning" mentioned by Silwal et al., 2022.

Both the proposed solutions to these limitations are discussed in Section 1.3.

Chapter 3

Three-Class Segmentation Neural Network with Simple Pruning Points

Summary

This chapter presents the first version of the created system prototype, which uses a simple three-class neural network with a dataset that I annotated, and proof-of-concept pruning points, along with the lessons learned in the course of this initial stage of the thesis. This chapter is split into two main sections: the segmentation neural network and the plant graph generation as well as the potential pruning point generation.

3.1 Segmentation Neural Network

The initial version of this system started as a prototype solution to understand the viability of using a single Intel RealSense camera in an environment where we have no control over the light conditions or the background. Considering that it was a prototype, a simple dataset annotated by me was used, and pruning points were placed above the second detected node starting from the cordon, without following any agronomically defined rule. With this, a simple system was created, although this system has several limitations and performance issues, as explained in Section 3.4.

3.1.1 Dataset

A dataset is needed to train the neural network for grapevine segmentation. I created a dataset with three classes: the cordon, the cane, and the node. An example of this annotation concept

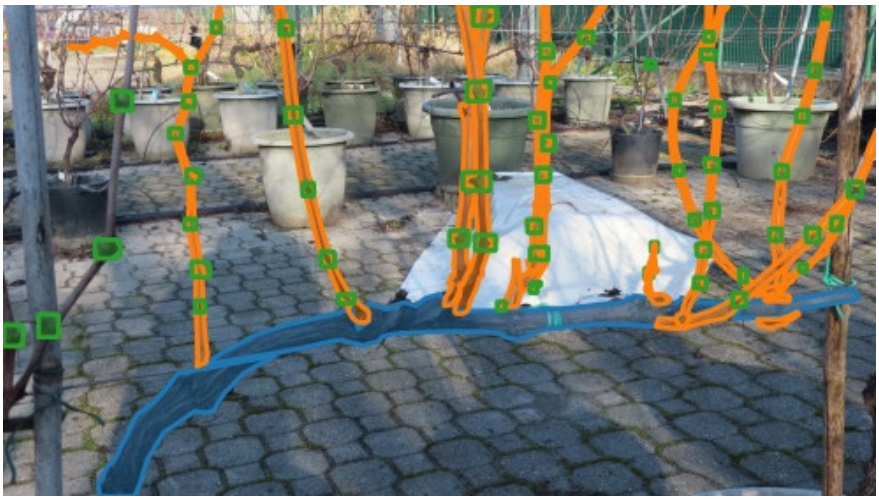

Figure 3.1 Example of an annotated grapevine, where blue represents the cordon class, orange is the cane class, and green is the node class.

can be seen in Fig. 3.1. The cordon is the horizontal, permanent static structure, and the canes are the plant organs generally showing vertical orientation. The nodes are structures present on the canes where future canes may grow. Dividing the grapevine into these three main categories allows us to generate potential pruning points. In other words, the cordon consists of the permanent thickest "branch" of the grapevine; the cane is a branch that starts from either the cordon or another cane; and the node is a region present only on canes, from where the next year canes will grow.

3.1.2 Data Acquisition

The data was acquired in an experimental vineyard at Università Cattolica del Sacro Cuore in Piacenza, see Fig 3.2. Data was captured using a typical 4k Canon compact camera by recording a video closeup of a grapevine row segment. The frames from the video were extracted using `ffmpeg` into a total of 171 frames. These frames have a resolution of 3840×2160 pixels. These images were the only ones used in this stage of the project, with considerations to add more via additional capture or by performing data augmentation. It is important to note that the camera used to capture the video that originates the dataset, and the camera placed on the robotic arm for inference purposes are different, with the first being used for data capture being a handheld 4k Canon compact camera, and the latter being an Intel RealSense D435, which is used for inference.

Figure 3.2 Outdoor simulated vineyard, used throughout the system's development, for data capture, platform testing, and pruning testing. It is located in Università Cattolica del Sacro Cuore in Piacenza.

3.1.3 Data Annotation

The dataset was annotated following the standard defined by the COCO segmentation dataset, with the three classes mentioned above: cordon, cane, and node. We chose the COCO format for the annotation since it is a standard format for segmentation annotation, which contributes to the higher availability of annotation tools that use this format. Some neural network frameworks provide built-in processing of this format, along with the ease of sharing this dataset with the scientific community.

The annotation tool used is COCO annotator, a web-based tool designed to label images efficiently. This tool also includes the possibility of connecting to an external neural network, allowing automatic annotation of an image using a pre-trained neural network. This feature has been used to aid in annotating more samples. The 171 captured images were annotated and split into 136 training and 35 evaluation images. We are not using a test set since the actual test results are the pruning results obtained on the experimental vineyard. Even though the quantity of data was low and the was annotated by me instead of experts in the agronomic field, these images and annotations were sufficient to kickstart the development of the desired solution and evaluate the feasibility of the approach.

The images are annotated in their original size, allowing the downscaling of the image as required without losing information, which happens when an image is downscaled and then reupscaled again.

3.1.4 Neural Network

To solve the proposed task, we used a deep neural network that can perform object detection and segmentation. Models such as Mask R-CNN are considered instead of YOLO (Versions 2 and 3) due to the latter not proving object segmentation. A counter-argument for using Mask R-CNN is the lack of performance or difficulty reaching real-time performance in a "simpler" scenario. Simple here means that the target scenario of winter pruning has fewer classes than Mask R-CNN originally classifies on datasets as ImageNet, which contains 1000 object classes, or COCO, which contains 81 classes. Considering it is one of the current state-of-the-art methods, Mask R-CNN was selected for the vision system.

Table 3.1 Detectron2 Mask R-CNN hyperparameters differences.

	Original	Modification
Batch size	16	4
Training Steps	(60000, 80000)	(25000, 48000)
Maximum Training Iterations	90000	50000

The currently used framework is Detectron2 (Wu et al., 2019), which was described in Section 2.1.3. The base model being used is the Mask R-CNN with several backbones tested. The first being tested was R50-FPN, a ResNet with 50 layers combined with Feature Pyramid Networks (FPN). For experimentation, two additional models were tested, the R101-FPN and the X101-FPN, where the first is a Residual Network with 101 layers with the same structure as the R50-FPN and the second is a ResNeXt with 101 layers.

The network is being trained using the default training procedure of Detectron2. This procedure creates a model, optimizer, scheduler, and dataloader with the default configurations provided along with the model. It then loads the pre-trained model weights, initializes logging functions, and follows a standard training workflow with a single-optimizer single-datasource iterative optimization.

The training hyperparameters are the default ones, with two changes, as presented in Table 3.1. The first is the batch size, or the images per training iteration, changed to 4 from the original value of 16. The reason for this is the lack of graphics card memory available;

Figure 3.3 An output example of potential pruning points generation overlaid on the segmentation output, where blue represents the cordon class, orange is the cane class, and green is the node class, with the red markers representing the generated potential pruning points.

Detectron 2 models are trained with NVIDIA V100 that have 16 gigabytes of RAM; the new training performed by me was completed on NVIDIA 1080Ti, which only has 8 gigabytes of RAM. The number of training iterations was also changed, going to 50000 from the original 90000, to prevent overfitness of the network over the low quantity of data.

3.2 Plant Graph Generation and Potential Pruning Points Detection

After obtaining the inference produced by the trained neural network, an additional processing layer is needed to find the desired potential pruning points. The method we decided to use was the creation of a processing layer that interprets this inference by understanding how the several segments are related to each other. While my focus was on the training of the neural network, Antonello Scaldaferri worked on the post-processing of the neural network's output, which is presented in this subsection. My contributions to this subsection are integration with the rest of the created system, bug fixing, and general supervision.

3.2.1 Plant Graph Generation

The data structure used to store the computer representation of the grapevine is a tree-shaped graph, with the cordon as the root element. Then, canes are connected to the cordon or other canes. The leaf elements are nodes or canes that have no nodes on them. The critical concept that we want to analyze is the relation between canes and nodes on the canes to obtain accurate potential pruning points. Figure 3.3 presents an example of a created plant graph overlaid on the segmentation output. This tree-shaped graph connects the various graph items that are inferred from the grapevine segmentation, showing the topographical structure of the plant. Each grapevine item consists of the following elements:

- Unique identifier number
- Bounding box coordinates
- Probability according to the neural network
- Segmentation mask
- Class identifier, name, and color
- Item color
- Center of the object
- Thickness of the element
- Depth information
- Distance from the parent organ
- A link to the parent organ

The object's thickness is calculated by averaging the pixel coordinate distance between the shortest extremities, which, for the canes, corresponds to measuring the distance between the minimum and maximum width along the cane height. The distance from the parent organ is calculated between the center of the object and the center of the parent's object.

The grapevine item is based on the common aspects of the three neural network classes and extended with additional class-specific elements. The cordon item, for example contains a list of canes sorted by their distance to the parent, and the canes have a list of nodes sorted using the same metric. In the end, the graph's root node is the cordon item.

There are three sets of connections in the graph structure: the "cordon to canes" set, relating the cordon to its connected canes; the "cane-to-cane proximity" set, relating each cane to its proximal canes; and the "cane to nodes" set relating each cane to its connected nodes.

Algorithm 1 presents a generic method to associate the canes to their respective cordon. It achieves this by using an auxiliary matrix containing all cordon instances, with shape

Algorithm 1: Get connections between two sets of masks

```
parameters : dilation, max_iter, n
input      : masks_A → Mask group A
input      : masks_B → Mask group B
output     : connections → Map of connections
foreach mask ∈ masks_B do
    iter = 0
    while iter ≤ max_it and not(connected) do
        if iter > 0 then
            Dilate mask
        end
        Get the indices of mask's non-zero values
        Obtain the corresponding values on masks_A
        if There are non-zero correspondences then
            Get the lowest correspondence
            Get the intersection with mask
            if Intersection is in the n^th slot then
                Add connection to connections
                Set connected to true
            end
        end
        iter++
    end
end
```

$n_{MC} \times H \times W$, where n_{MC} is the number of cordons inferences, H and W is the height and width of the input image. As equation (3.1) shows, each cordon segmentation mask (*mask*) is inserted into the matrix ($masks_A$) with the corresponding identifier number (ID_{MC}), increased by 1, leaving 0 as the background class, at its non-zero values.

$$\begin{gathered} \forall mask \in masks_{MC} : \\ masks_A[i] = mask * (IDs_{MC}[i] + 1), \\ i = 1, ..., n_{MC} \end{gathered} \tag{3.1}$$

this page is very hard to follow. I think you could give example images of how the dilation is used to make non-connected canes intersect with the cordon etc. maybe a figure with 2 rows of 5 images like this: first row: a sequence of images with 0, 1, 2, 3, 4 dilation steps for cane to cordon connection. second row: same as above but with cane to cane connections. Also an example of the map showing the keys and values, e.g. For each cane segmentation mask, the row and column indices of its non-zero values are searched and then used to get

the unique values in the matrix at these indices across all the cordon segmentation masks. This way, we can know which is the overlapping cordon. Then, dividing the cane mask into n different slots, arranged vertically, we select the cordon that overlaps the cane in the n^{th} slot, if it is present. It may happen that the cane does not coincide with any cordon inference due to the imperfection of the inference. As such, a solution was found by performing an incremental dilation of the cane segmentation mask until a cordon was found or a maximum number of dilations was reached. The number of dilations, the size of the dilation, and the number of vertical slots are user-defined parameters for this algorithm.

The same method can be iteratively used to find the connections between the proximal canes. The initial search field comprises only the canes already connected to the cordon. New connections are searched for nonconnected canes among the connected canes (the search field), using the same overlapping and dilation concepts at each iteration. These newly found connections are then added to the final connections set, and the two groups of masks are updated. The iteration stops when no new connections are found or all the canes have been connected. The algorithm starts by considering the output of Algorithm 1 as $masks_A$ and the set difference between all cane masks and $masks_A$ as $masks_B$. It also takes as input the size of the dilation ($dilation$), the maximum number of dilations (max_it), and the number of vertical slots (n), and it outputs a map whose keys are cane identifiers and values are lists of cane identifiers.

The last set of connections is obtained using a variation of Algorithm 1, where, instead of checking if the intersection is contained in a specific part of the mask, the most overlapping cane is considered as connected cane. Moreover, the node masks are approximated by their bounding boxes without any need for dilation. The algorithm takes as input the list of cane masks ($masks_A$) and the list of node bounding boxes ($masks_B$), and it outputs a map whose keys are cane identifiers and values are lists of node identifiers.

3.2.2 Potential Pruning Points Detection and Localization

Pruning points can be generated with the previously created structure. We decided to use a crude approach for detecting potential pruning points, which are points on canes, either between two nodes of the same cane, between the bases of two canes growing from the same cane, or between the base of a cane and its first node. This example can be seen in Fig. 3.3, indicated by the red markers. As equation (3.2) shows, by default, a potential pruning point ($\vec{pp}$) is the midpoint between two of the previously mentioned points ($\vec{p}_1$ and $\vec{p}_2$). To select the final pruning point, we chose the pruning point above the second node of a cane

during this initial work. Due to the possible curvature of the canes, this midpoint may not be contained in the cane mask, and if this happens, the point is moved to a point inside the mask. For this, it is first verified if the pruning point is contained in the corresponding cane mask. If not, it is moved in its cane mask, ensuring it is not transferred outside the correct region, i.e., the rectangular area enclosed by the two points. An orientation angle is needed to orient the pruning tool roll angle to perform the cut correctly. This orientation angle (α) is computed considering the slope angle between the straight line connecting the two points and the horizontal direction of the image.

$$\begin{gathered} \forall(\vec{p_1}, \vec{p_2}) \\ \vec{pp} = \frac{\vec{p_1} + \vec{p_2}}{2} \\ \delta_{x,y} = p1_{x,y} - p2_{x,y} \\ \alpha = \begin{cases} 0, & \text{if } \delta_x = 0 \\ \frac{\pi}{2}, & \text{if } \delta_y = 0 \\ \arctan\frac{\delta_y}{\delta_x} - \text{sign}\,\frac{\delta_y}{\delta_x} * \frac{\pi}{2}, & \text{otherwise} \end{cases} \end{gathered} \tag{3.2}$$

3.3 Neural Network Results

This section is split into two parts, one presenting the quantitative results of the neural network via the validation results during training and the second giving a qualitative evaluation of the plant graph generation algorithm and the subsequent potential pruning points generation.

3.3.1 Quantitative Evaluation

Regarding a quantitative evaluation of the neural network, Fig. 3.4 shows the various Average precision (AP) metrics calculated amidst the training process. The AP metric presented in the figure is on par with the Detectron2 COCO instance segmentation baseline, where the segmentation AP is around 40% in our training where the baseline value is 37.9% (As of the baseline present in the Detectron2 model zoo Wu et al., 2021). The difference between the various classes is also noticeable, which can be explained by the difficulty of each class, starting from the easiest. The nodes are closer to bounding boxes than segmentation since the node is a specific structure along the cane. Due to the nature of the training process performed on the grapevine by the grower, the cordon is a well-defined structure present on the grapevine, mainly in the same place, with a lesser variation of its shape. The canes are an

object that is harder to perceive due to being more prone to occlusion by other canes, blending with the background due to their thinness and randomness. Table 3.2 shows the comparative results of the three tested models after training, presenting both Average Precision and Average Recall.

Table 3.2 Average Precision and Average Recall after training for the three tested models. Both metrics are calculated with intersection over union IoU=0.50:0.90, considering all area sizes and a maximum detection count of 100.

	Resnet 50	Reset 101	ResNeXt 101
Average Precision	39.0%	41.1%	41.6%
Average Recall	47.0%	48.7%	48.2%

3.3.2 Plant Graph Creation and Potential Pruning Points Generation

The potential pruning points detection performance is related to the graph generation performance, which depends on the segmentation network performance. This is because each connection is related to a potential pruning point, except for the connections between cordons and canes. In particular, each missed connection leads to a missed potential pruning point, i.e. a false negative sample. Non-existent connections may be established, leading to extra-detected pruning points, i.e. false positive samples. In the case of cane connections, the actual connection rule is based on checking if the masks are intersected in their bottom part since, as said, canes tend to grow upwards. With this, canes that grow downwards, although rare, are not detected.

When considering the performance of the several created components, it is also essential to understand how they perform on the field. An example of this can be seen in Fig. 3.5, where although it does not find every single cane, the graph generation still manages to find viable potential pruning points that can be used. It is important to note that the images were not used to train the network or acquired with the same camera as the one used on the robot. These results may not represent the complete challenge the neural network using images captured by the robot may face, but they are still essential to analyze.

Figure 3.4 The validation results for the network model ResNet-50 along the training process, following the COCO evaluation method, presenting the Average Precision (AP) of each singular class and the mean Average Precision of all classes.

3.4 Conclusions

Although an initial stage prototype, it brought many lessons that shaped the following years of development. The data acquisition was performed far from the plant, which made annotating specific organs difficult. At the same time, since it is not the same point of view of the robot, it also led to detection issues. The annotation brought the problem of not discriminating between different organs of the plant structure, which is essential for selecting the pruning points. Mask-RCNN's speed performance is not the best, taking around one-tenth of a second to infer the input images when using ResNet-50. Although not time efficient, the system does not need to achieve real-time execution, allowing the use of slower models and backbones while not needing to compromise the neural network accuracy. Finally, the platform used is not meant to be used in outdoor applications due to its lack of weatherproofing and the

Figure 3.5 Two pairs of images, where the first image of the pair shows the inference output and the second image shows the plant graph and potential pruning points generation. As in the previous figures, blue represents the main cordon class, orange represents the various detected canes, and green represents the nodes. The red markers represent the generated potential pruning points.

abovementioned issues related to navigation in rough terrain. Nevertheless, this platform worked on the simulated vineyard in Fig. 3.2. The software separation also leads to a situation where the wheeled platform does not assist the arm in planning movements when the platform is too far from the plant or when moving the platform would give an advantage to the arm motions.

Due to the imperfect nature of the obtained inference, the detected grapevine items, created and used for potential pruning points generation, may not be accurate. Multiple factors, such as the lack of data on grapevine specimens of various ages or the capture conditions, cause this imperfection. To mitigate this, data augmentation may be considered to expand the existing dataset and capture new images from different grapevines. Other issues are caused by misunderstandings related to how grapevines grow, where, for example, some pruning regions appear to grow from the bottom of the cordon rather than the top, which causes some organs not to be related to each other. However, we can create a 2D plant structure model even considering imperfect segmentation. The plant model, and consequently the pruning points, is heavily dependent on the segmentation, which could lead to the non-detection of potential pruning points. Furthermore, the algorithm that establishes the connections can be seen as a crude method that does not solve certain anomalies that may happen, leading to false connections being created.

These initial results were published in a conference paper *Grapevine winter pruning automation: On potential pruning points detection through 2d plant modeling using grapevine segmentation* (Fernandes et al., 2021), which was used as an introductory conference paper about the Vinum project that got the Best Student Paper Award.

Chapter 4

Five-Class Segmentation Neural Network with Agronomic Pruning Points

Summary

This chapter presents a direct evolution of the work presented in the previous chapter. It shows the improvements in the data capture and annotation for the neural network, moving from three classes to five classes. Changes to the pruning point generation system, going from a prototype version that places pruning points above every second node to making it follow agronomic rules.

4.1 Segmentation Neural Network

With the lessons learned during the execution of the work presented in the previous chapter, the existing system was upgraded by tackling the issues mentioned previously. The areas worked on in this section are the dataset, data annotation, and neural network training. The first issue that needs special attention is the fact the dataset presented in Section 3.4 does not consider the correct organs for calculating pruning pints that follow agronomic knowledge.

4.1.1 Dataset

A dataset is an essential component of the implementation of a machine-learning method. Since no dataset with the plant organ segmentation of spur-pruned grapevines was available online, there was the need to create a new dataset with the annotation of five selected organs: cordon, arm, spur, cane, and node. A semi-schematic view of the grapevine plant

Table 4.1 Number of annotations per class category in each plant group.

	Control	Shoot Thinning
Cordon	79	84
Cane	440	341
Node	110	912
Arm	103	154
Spur	116	144
Total	1838	1635

can be seen in Fig. 1.3, along with an explanation of the individual organs in Section 1.2 An example of these annotation concepts can be seen in Fig. 4.1. Since it is one of the contributions to the scientific community, the created dataset is shared in open access format at https://zenodo.org/record/5501784.

4.1.2 Data Acquisition

The data was acquired in the simulated vineyard at UCSC, Piacenza, Italy, as mentioned in the previous chapter and shown in Fig. 3.2. It comprises two groups of grapevine specimens, where the first group contains seven plants that act as a control group, and the second group containing eight grapevines is subjected to shoot thinning.

The images were acquired with a typical 4k Canon compact camera, by taking pictures from both sides of the grapevine plant. Similar to the previous chapter, the camera used to capture the video that originates the dataset, and the camera placed on the robotic arm for inference purposes are different, with the first being used for data capture being a handheld 4k Canon compact camera, and the latter being an Intel RealSense D435, which is used for inference.

Shoot thinning is a summer pruning technique that reduces disease pressure, improving canopy microclimate, vine balance, and grape quality to increase the sustainability of viticulture (Poni, Gatti, et al., 2018). Moreover, this practice allows more efficient shoot positioning in Vertical Shoot Positioned trained vines due to the reduced shoot number, making the management of their growth direction and orientation easier and, in turn, facilitating winter pruning operations.

Each plant has around five spurs, and the photos are taken from both sides of the grapevine specimen, meaning that there is one picture where the plant grows from left to right and the other side where the plant grows from right to left. The resolution of the images is 4608x3456 pixels, and we have a total of 149 images.

4.1.3 Data Annotation

The dataset is annotated following the COCO segmentation dataset, with the five classes mentioned: cordon, arm, spur, cane, and node. It follows the same rules as defined in Section 3.1.3 by using the COCO format for the annotation and using the same annotation tool, with the only difference being the using the five classes rather than just three classes. The 149 captured images were annotated and then split into 80% (119) for training and 20% (30). The images are annotated in their original size, allowing us to downscale them as required. Two agronomy experts from Università Cattolica del Sacro Cuore (Paolo Guadagna and Alessandro Santamaria) annotated these images, with Prof. Matteo Gatti reviewing the annotations.

Figure 4.1 Example of a grapevine organ annotation with five classes. Purple represents the cordon class, green is the arm class, red is the spur class, orange is the cane class, and light blue is the node class. This picture was taken inside the simulated vineyard at UCSC, Piacenza, Italy.

4.1.4 Neural Network

This section describes the neural network architecture and backbone used for this work. The framework that was used is the same as the previous chapter, which is Detectron2 (Wu et al., 2019), Facebook AI Research's implementation of state-of-the-art detection algorithms. The neural network training is performed the same way as described in Section 3.1.4, with the only difference being the training of only one backbone. The selected backbone was ResNet-50. The selection of this backbone was defined by results presented in Table 3.2, where the selected backbone presented similar results to the other backbones evaluated. Being a smaller backbone allows a quicker execution time and model training time.

4.2 Grapevine Plant Model Structure & Modeling Algorithm

In the previous chapter, I introduced a 2D plant modeling algorithm, which considers only three types of organs. As mentioned in Section 3.4, this model type can lead to inconsistencies in the overall model, resulting in unreliable information. Understanding that there were ways to improve the model, using depth information, we present a new modeling algorithm that we consider more reliable and that has stricter relationships between the different detected organs of the plant, considering the new five-class categorization. As with the previous chapter, Antonello Scaldaferri worked on the post-processing of the neural network output, which is presented in this subsection. My contributions related to this subsection are integration with the rest of the created system, bug fixing, and general supervision.

4.2.1 Grapevine Plant Model Structure

To create this model, we use a tree-based data structure since its form is suitable to represent the relationship of the different detected grapevine organs. The base structure is the same as presented in Section 3.2.1, with the only difference being the change from three grapevine organs to five grapevine organs. Based on the new five-class categorization, we defined a new set of more specific and stricter relationships between the different types of organs to create the new plant model, listed in Table 4.2.

Table 4.2 Relationships between the different types of organs.

Parent Organ Type	Children Organs Types
Cordon	*Arm*, *Spur*, *Cane*
Arm	*Spur*, *Cane*
Spur	*Cane*
Cane	*Node*

Grapevine Plant Modeling Algorithm

For the modeling algorithm, we use an upgraded version of Algorithm 1 present in Section 3.2.1, to compute the connections between two sets of organs of different types, whose pseudo-code is given in Algorithm 2. It uses the same concept of overlap between masks as the previous one and uses the same mask dilation and subdivision heuristics. The difference is that now we consider both upward and downward growing directions of the organs, i.e., organs that grow either from the upper side of the parent organ or the lower side, fixing a grouping issue mentioned in Section 3.4. This algorithm is executed for each type of *parent-children* connection to compute all the final connections for the entire model, according to Table 4.2. Each time, it takes as input the two corresponding sets of items, the first for items of type *parent* and the second for items of type *children*. For the last type of connections, where the *nodes* are considered, the set of the children masks is replaced with their corresponding bounding boxes for simplicity.

4.2.2 Pruning Points Detection

After obtaining the grapevine plant model, it is used to compute the pruning points based on previously described rules. One of the main contributions of this thesis is to provide a set of pruning points that is as reliable as possible. To obtain such a set of points, we defined a set of assessments to simplify the reasoning process of the pruning points detection phase. The set comprises the following evaluations: pruning region location, pruning region canes number, basal cane growth direction, basal cane vigor, basal cane origin, and adjacent pruning region distance. In the created model, each pruning region is a grapevine item, and its type can be either *arm*, *spur* or *cane*, according to Table 4.2.

The pruning region location is the area from which *arm* or *spur* grow in respect to the *cordon*. For its assessment, three different angles α_V, α_I, and α_D are considered to classify the pruning region as ventral, intermediate, or dorsal, which means that the organs present in the pruning region grow from the bottom, middle or top side, respectively. We define

Algorithm 2: Compute connections between two types of organs.

parameters : $dilation, max\_iter, n, include\_top$
input : $items_A \rightarrow$ Set of type A items
input : $items_B \rightarrow$ Set of type B items
output : $connections \rightarrow$ Map of connections
$masks_A \leftarrow$ set of masks of $items_A$
$IDs_A \leftarrow$ set of instance IDs of $items_A$
foreach $i, mask_A \in$ ***enumerate***$(masks_A)$ **do**
  $masks_A[i] \leftarrow mask_A * (IDs_A[i] + 1)$
foreach $item \in items_B$ **do**
  $iter \leftarrow 0, connected \leftarrow False$
  $mask \leftarrow item$ segmentation mask
  while $iter \leq max\_it$ ***and not***$(connected)$ **do**
    if $iter > 0$ **then**
      Dilate $mask$
    $indices \leftarrow mask$ non-zero values indices
    Obtain $masks_A$ unique values at $indices$
    if *There are non-zero correspondences* **then**
      $ID_A, mask_A \leftarrow [lowest]$ correspondence
      $inter \leftarrow$ **intersection**$(mask_A, mask)$
      if $inter$ *is in the mask* $[n^{th}]$ *slot* **then**
        $item_A \leftarrow items_A(ID_A)$
        Add $(item_A, item)$ to $connections$
        $connected \leftarrow True$
    $iter++$
  if ***not***$(connected)$ ***and*** $include\_top$ **then**
    Repeat the previous **while** loop considering the $[highest]$ correspondence and the $[1^{st}]$ slot

(x_{PR}, y_{PR}) as the pruning region origin image coordinates and d_{MC} as the diameter of the *cordon*, which can be estimated from its segmentation mask, at the column x_{PR}, in pixels. Moreover, we define y as the first non-zero row in the *cordon* segmentation mask. With these definitions, the pruning region location is classified as:

$$\begin{cases} dorsal & \text{if } y_{PR} < y + \frac{d_{MC}}{2} * (1 - \cos\frac{\alpha_D}{2}) \\ ventral & \text{if } y_{PR} > y + d - \frac{d_{MC}}{2} * (1 - \cos\frac{\alpha_V}{2}) \\ intermediate & \text{otherwise} \end{cases} \tag{4.1}$$

For the pruning region canes number assessment, a Depth-first search algorithm is used, counting the items of type *cane* in the sub-tree represented by the pruning region. In a pruning region, the basal cane is the *cane* closest to the *cordon*. For its growth direction assessment, two other different angles, α_L and α_C are considered to establish whether it is vertical in both the lateral and cross sections. We define (x_o, y_o, z_o) and (x_e, y_e, z_e) as the basal cane origin and endpoint in the 3D space, that can be computed thanks to the depth image and camera intrinsic parameters. With these definitions, the growth direction can be classified as:

$$\begin{cases} vertical & \text{if } \frac{|x_o - x_e|}{|y_o - y_e|} \leq \alpha_L \ \ \& \ \ \frac{|z_o - z_e|}{|y_o - y_e|} \leq \alpha_C \\ not\ vertical & \text{otherwise} \end{cases} \tag{4.2}$$

The vigor of the basal cane is its thickness, which can be estimated from its segmentation mask and the depth image. Based on two threshold values, it establishes which canes to keep and which to remove if they are too thick or thin. We define *rows* and *cols* as the row and column indices of the basal cane segmentation mask. For simplicity, we suppose that the basal cane is vertical or horizontal based on its bounding box aspect ratio. The subsequent considerations are valid for vertical basal canes. In the case of horizontal basal canes, swap the rows with the columns.

Since the depth value or the distance between the camera and the points located at the basal cane segmentation mask boundaries may not be accurate, for each *row* in *rows*, we consider an estimation of the depth value at *row*, considering the midpoint between the mask columns in that specific *row*. This *row* depth estimation is then used to obtain the 3D points at the *row* boundaries, to compute the *row* thickness as:

$$\begin{gathered} thicknesses \leftarrow \text{empty list} \\ \forall row \in rows \\ cols_{row} \leftarrow \text{get } cols \text{ values where } rows \text{ is } row \\ col_{mean} \leftarrow mean(cols_{row}) \\ depth \leftarrow estimate\_real\_depth(\langle col_{mean}, row \rangle) \\ col_m, col_M \leftarrow \min(cols_{row}), \max(cols_{row}) \\ \vec{p}_{row_1} \leftarrow to\_3D(\langle col_m, row \rangle, depth) \\ \vec{p}_{row_2} \leftarrow to\_3D(\langle col_M, row \rangle, depth) \\ thickness_{row} \leftarrow |\vec{p}_{row_1} - \vec{p}_{row_2}| \\ \text{add } thickness_{row} \text{ to } thicknesses \end{gathered} \tag{4.3}$$

The final estimation of the vigor of the basal cane is computed considering the mean value of all the *thicknesses* among all the *rows*.

Another critical point is the basal canes origin assessment, which is necessary for two evaluations. The first is to classify the pruning region as new or not, verifying whether the *parent organ type* is *cordon* or not, respectively. The second is to verify if the pruning region concerns a replacement cut or not, verifying whether the parent organ type is *arm* or not, respectively.

The last considered assessment is the adjacent pruning region's distance assessment. This is necessary when multiple new pruning regions are close together. Based on a threshold value, it is verified whether these pruning regions are too close or not to remove or keep them, respectively. We define *PRs* as the set of all the grapevine plant pruning regions. For each pruning region *PR* in *PRs*, its distance d_{PR} to adjacent pruning regions is computed as:

$$\begin{aligned} PR_{-1} &\leftarrow \text{get the previous pruning region from } PRs \\ PR_{+1} &\leftarrow \text{get the next pruning region from } PRs \\ \vec{p}_{PR} &\leftarrow \text{get the 3D origin of } PR \\ \vec{p}_{PR_{-1}} &\leftarrow \text{get the 3D origin of } PR_{-1} \\ \vec{p}_{PR_{+1}} &\leftarrow \text{get the 3D origin of } PR_{+1} \\ d_{PR} &\leftarrow \max(|\vec{p}_{PR} - \vec{p}_{PR_{-1}}|, |\vec{p}_{PR} - \vec{p}_{PR_{+1}}|) \end{aligned} \tag{4.4}$$

Furthermore, to perform the final pruning correctly, different types of cuts for the grapevine organs are defined, the details of which are given in Section 1.2.

In particular, each pruning point $\vec{p}p$ is computed as a point between two other points $\vec{p}_1$ and $\vec{p}_2$. These points $\vec{p}_1$ and $\vec{p}_2$ can be *nodes* locations, *canes* / *spurs* / *arms* origins or endpoints. The estimated pruning points may be too far from their corresponding organ origin. This can happen when *nodes* are not detected. In these cases, it is helpful to compute the pruning point using a distance parameter d that can be set by the user based on the vineyard grapevine type.

Note that if d is 0, the pruning point is placed at $\vec{p}_1$ and if d is $\frac{1}{2}$, the pruning point is the midpoint between $\vec{p}_1$ and $\vec{p}_2$ and if d is 1, the point is placed at $\vec{p}_2$.

With these definitions, a pruning point $\vec{pp}$ can be computed as:

$$\begin{aligned} D &\leftarrow |\vec{p}_1 - \vec{p}_2| \\ w_1 &\leftarrow \frac{|D-d|}{D} \\ w_2 &\leftarrow 1 - w_1 \\ \vec{pp} &\leftarrow w_1 * \vec{p}_1 + w_2 * \vec{p}_2 \end{aligned} \tag{4.5}$$

And its corresponding orientation angle α as:

$$\begin{aligned} \delta_x &\leftarrow p_{1x} - p_{2x} \\ \delta_y &\leftarrow p_{1y} - p_{2y} \\ \alpha &\leftarrow \begin{cases} 0, & \text{if } \delta_x = 0 \\ \frac{\pi}{2}, & \text{if } \delta_y = 0 \\ \arctan\frac{\delta_y}{\delta_x} - \operatorname{sign}\frac{\delta_y}{\delta_x} * \frac{\pi}{2}, & \text{otherwise} \end{cases} \end{aligned} \tag{4.6}$$

Since the pruning point is computed as a point between two other points along a straight line, it may occur that it is not on the corresponding grapevine organ, which means it is not contained in the corresponding grapevine item segmentation mask. We need to move the estimated points on their corresponding organs. This correction is applied to each pruning point, moving it horizontally or vertically until it reaches its corresponding segmentation mask or a meaningful area of the depth image.

4.3 Results

4.3.1 Neural Network Evaluation

This section presents the used criteria and results from the quantitative evaluation performed on the neural network. While I provided the raw data, these results were mainly created by Prof. Matteo Gatti and Paolo Guadagna, from Universitá Cattolica del Sacro Cuore, with these results being presented in the produced journal paper Guadagna, Fernandes, et al., 2023.

Criteria

For each of the five classes used, the output of the grapevine segmentation network was compared to the annotated images. The correctness of a detected grapevine organ was assessed through the IoU overlap with the corresponding ground truth labeling. The IoU overlap was defined according to the following equation:

$$IoU = (A \cup B)/(A \cap B) \tag{4.7}$$

Within every class, a detected object was assumed as a true positive when its IoU overlapping with the ground truth annotation was higher than 0.5. In parallel, the dataset was also analyzed with an IoU of 0.2 and 0.7. The output was classified as a false negative (false negative) when a detected organ did not reach the minimum IoU threshold. The output was classified as false positive in the case of no overlap with the corresponding ground truth annotation. For false positives, the misclassed grapevine organ or other element was described and considered for further analysis.

The neural network performances were evaluated through the percentage of false positives among the overall inferences, recall, precision, and F1 scores, which were calculated for the overall object population according to the following equations:

$$FDR = FP/(TP+FP) \tag{4.8}$$

FDR: False Discovery Rate, TP: True Positive, FP: False Positive

$$Recall = TP/(TP+FN) \tag{4.9}$$

TP: True Positive, FN: False Negative

$$Precision = TP/(TP+FP) \tag{4.10}$$

TP: True Positive, FP: False Positive

$$F1 = 2*(TP*FP)/(TP+FP) \tag{4.11}$$

TP: True Positive, FP: False Positive

Evaluation

The general performances of the segmentation network were described by a recall of 0.81 and a precision of 0.97 with an F1 score of 0.88, as seen in Table 4.3.

Table 4.3 Overall performance of the neural network for grapevine segmentation with an IoU of 0.5, with TP standing for True Positive, FN for False Negative, and FP for False Positive.

Count	TP	FN	FP	Recall	Precision	F1 score
1359	1069	258	32	0.81	0.97	0.88

Table 4.4 Performance measures of the neural network for grapevine segmentation depending on five different grapevine organs with an IoU of 0.5, with TP standing for True Positive, FN for False Negative, and FP for False Positive.

Organ	Count	TP	FN	FP	Recall	Precision	F1 score
Cordon	75	61	14	0	0.81	1.00	0.90
Arm	89	71	17	1	0.81	0.99	0.89
Spur	108	77	30	1	0.72	0.99	0.83
Cane	343	229	107	7	0.68	0.97	0.80
Node	744	631	90	23	0.88	0.96	0.92
Total	1359	1069	258	32			

The most recurrent grapevine organ in the testing dataset was a node, followed by cane, spur, arm, and cordon, as provided in Table 4.4. False positives related to each class were generally low. Nodes scored the highest recall value (0.88), followed by cordon and arms (0.81), while spur and cane classes revealed a recall of 0.72 and 0.68, respectively. The precision values ranged from 0.96 (node) to 1 (cordon), with arm and spur segmentations showing intermediate performances.

The most represented category for canopy management was control, followed by shoot thinning and light pruning (Table 4.5). True positives were 493 in control, 487 in shoot thinning, and 89 in light pruning, with the highest recall values calculated for grapevine organs subjected to shoot thinning (0.85) and relatively lower performances described in control (0.80); moreover, the segmentation of the grapevines subjected to light pruning led to the lowest recall. With only five wrong inferences, precision was highest in control (0.99), with similar responses described for shoot thinning organs despite the higher number of false positives (15). Conversely, although the false positives in light pruning (12) were relatively similar to shoot thinning, the precision was much lower (0.88).

To investigate if and how vineyard management influences dormant canopy segmentation, the model was tested against each treatment × grapevine organ combination, with the

Table 4.5 Performance measures of the neural network for grapevine segmentation depending on canopy management with an IoU of 0.5, with TP standing for True Positive, FN for False Negative, and FP for False Positive.

Treatment	Count	TP	FN	FP	Recall	Precision	F1 score
Control	619	493	121	5	0.80	0.99	0.89
Shoot Thinning	590	487	88	15	0.85	0.97	0.90
Light Pruning	150	89	49	12	0.64	0.88	0.74
Total	1359	1069	258	32			

quantitative results being presented in Table 4.6. Treatment here corresponds to one of three: control, shoot thinning, and light pruning. With Control corresponding to unthinned grapevines, shoot thinning to the studied technique mentioned in Section 4.1.2. Light pruning is an undesirable treatment in normal circumstances since it favors acrotony because the node count per spur is higher than two. Acrotony refers to a plant trait where the upper branches exhibit more pronounced growth than the lower ones.

In control vines, the cordons were detected with a recall of 0.80. Arm segmentation was described with higher recall (0.87), while the model resulted in poorer performance in identifying spurs and canes. No false positives were counted in these grapevine organs, giving a precision of 1. The node class had the highest recall (0.89), having 296 true positives and 37 false negatives. The model returned five wrong classifications (false positives), lowering the precision to 0.98.

As expected, shoot thinning showed a lower count than the control for annotated canes and nodes and a similar number of annotated elements for cordons, arms, and spurs. When compared to the control group, in shoot thinning grapevines, the recall values increased for cordon (0.91), spur (0.77), and cane (0.76) with no or minor changes for nodes (0.89) and arms (0.85), respectively.

Although correct inferences in shoot thinning proportionally increased as compared to control, the model errors also increased, affecting the precision for most of the classes such as arm, cane, and node, showing the following values: 0.97, 0.94, and 0.97, respectively, as seen in Table 4.6.

The light pruning presented fewer annotated grapevine organs, as noted in Table 4.6. In most cases, false negatives were similar to or higher than true positives. This condition was mirrored by performance metrics such as recall and F1 score, revealing the lowest values within the experiment. The recall and F1 scores identified poor segmentation performances

Table 4.6 Performance measures of the neural network for grapevine segmentation depending on canopy management and grapevine organs with an IoU of 0.5, with TP standing for True Positive, FN for False Negative, and FP for False Positive.

Treatment	Organ	Count	TP	FN	FP	Recall	Precision	F1 score
Control	Cordon	30	24	6	0	0.80	1.00	0.89
	Arm	39	34	5	0	0.87	1.00	0.93
	Spur	43	31	12	0	0.72	1.00	0.84
	Cane	169	108	61	0	0.64	1.00	0.78
	Node	338	296	37	5	0.89	0.98	0.93
Shoot Thinning	Cordon	34	31	3	0	0.91	1.00	0.95
	Arm	41	34	6	1	0.85	0.97	0.91
	Spur	56	43	13	0	0.77	1.00	0.87
	Cane	141	103	32	6	0.76	0.94	0.84
	Node	318	276	34	8	0.89	0.97	0.93
Light Pruning	Cordon	11	6	5	0	0.55	1.00	0.71
	Arm	9	3	6	0	0.33	1.00	0.50
	Spur	9	3	5	1	0.38	0.75	0.50
	Cane	33	18	14	1	0.56	0.95	0.71
	Node	88	59	19	10	0.76	0.86	0.80
Total	1359	1069	258	32				

for arms and spurs (0.33 and 0.38 recall, respectively) and higher sensitivity for node detection (0.76 recall). Precision was affected mainly by count varying between 0.75 (spur) and 1 in the case of cordons and arms, where no false positives were detected.

Several elements of the grapevine or the surrounding environment were associated with wrong predictions, such as arms, spurs, canes, and nodes, as seen in Table 4.7. Nodes were the most wrongly attributed class since 3.6% of the inferences were false positives. The second most frequent incorrect class attribution concerned canes (3.03%), while wrong arm and spurs segmentation was limited to 1.39% and 1.28%, respectively.

Table 4.7 Performance measures of the neural network for grapevine segmentation depending on canopy management and grapevine organs with an IoU of 0.5. Det. Class stands for Detected Class, F. D. R. stands for False Discovery Rate

Det. Class	True Class	Count	Mean Confidence	F. D. R. (%)
Arm	Cane	1	0.91	1.39
Cane	Other object	3	0.89	1.29
Cane	Arm	2	0.95	0.87
Cane	Other grapevine organ	2	0.96	0.87
Node	Other object	11	0.97	1.71
Node	Arm	2	0.99	0.32
Node	Other node	7	0.98	1.10
Node	Other grapevine organ	3	0.99	0.47
Spur	Other grapevine organ	1	0.99	1.28
Total		32		

Neural Net Discussion

The current study allowed the fine-tuning and testing of a novel neural network for grapevine organ identification at 0.5 IoU and 0.7 confidence, resulting in the following performance metrics: Recall of 0.81 and a precision of 0.97, as demonstrated previously in Table 4.3 As already mentioned about PR detection, the current results suggest that assuming lower confidence would increase the network sensitivity towards the grapevine organs identification; as a matter of fact, the general improvement of the detection process would lead to an increased recall at the expense of precision because of the higher number of inferences (true positive and false positive) regardless of their correctness as presented in Table 4.3.

Because the current segmentation network is expected to support grapevine organ identification from a winter pruning perspective, developing highly performant systems is required to limit the risk of missing pruning regions and cutting points. Indeed, this is well known as spur-pruning over dormancy requires specific cuts (i.e., renewal cuts, cane shortening) to be applied to all the pruning regions along the cordon and as an automated pruning system should exclude any manual follow-up of unpruned PRs randomly spread through the vineyard.

When considering its sensitivity in detecting the five organ classes featuring the grapevine canopy over winter, the neural network resulted in different performances as reported in Table 4.4. With a recall of 0.88 (i.e. 88% of the specific annotations identified), nodes were the best-detected class, corresponding to the most represented class in the test dataset due to the grapevine structure. Consequently, further improvement of the current neural network version work will provide more training examples of the under-represented classes, such as cordons, arms, and spurs, to have a more balanced dataset and consistent results among the five classes. In addition to the different abundance of training data, the heterogeneous performances describing our segmentation process can be explained by the different grapevine organ sizes (i.e. thickness and width) characterizing a grapevine canopy over dormancy.

Data reported in Table 4.4 describe a higher segmentation rate for more prominent organs such as cordons and arms (recall = 0.81), while spurs were less detected (recall = 0.72) because of their thinner structure. The importance of the size of target organs is also confirmed when comparing segmentation performances described for arms and spurs; indeed, because a spur might be considered as the natural continuation of an arm, and the ratio between their count approaches 1 in both training and test datasets, the higher detection described for the arms might depend on the more complex structure characterizing a permanent organ older than two years as compared to a 2-year-old spur. Canes, despite being the worst detected organ by the present algorithm (recall = 0.68), were associated with a higher recall value compared to the results reported by Botterill et al., 2017 with the 2D cane detector (0.49).

The generally worse segmentation results obtained for spurs and canes can be linked to the higher probability of occlusions. More prominent and isolated organs such as cordons and arms are much less subject to occlusion than spurs, relatively thin and short elements surrounded by canes, and canes that often cross each other or are self-occluding. Precision values in experiment 2 are significantly high because of the low number of false positives for each of the five classes.

Canopy management greatly affected the segmentation results, showing the best detection performances in shoot thinning grapevines where only one shoot per node was kept (Table 4.5).

Consequently, a shoot thinning canopy has fewer elements to be detected, fewer potential occlusions, and a more standardized canopy that leads to better results when applying computer vision algorithms. However, the three treatments revealed different outcomes regarding grapevine organ segmentation, shown in Table 4.6. Despite slightly improving the overall performances, control followed the same ranking already described in Table 4.4

with recall values decreasing in the following order: node > arm > cordon > spur > cane. Segmentation of shoot thinning canopies revealed the highest recall values; precisely, cordons (0.91) were followed by nodes (0.89) and arms (0.85). The same recall value describes node segmentation. The reason recall does not decrease in control treatment is probably due to a lower frequency of occlusion since nodes could only be masked by very thin organs such as canes. In parallel, nodes were successfully segmented in light pruning because their morphology did not differ among treatments. At the same time, segmentation performances dramatically decreased for the other organs in response to altered growth patterns and pruning region morphology induced by highly variable spur length.

Shoot thinning becomes a quite promising practice in vineyards that will be subjected to automated robotic pruning because of the following reasons: (i) better performances of perception modules such as PR detection and grapevine organ segmentation due to the increased proportion of simple spurs and limited frequency of occlusions; (ii) better performances of the manipulation module, by facilitating the motion planning to reach cutting points as well as the end-effector operability; (iii) a significant decrease in cut number per meter of row impacting on robot capacity. On the other hand, such a pivotal role assumed by canopy management supports the idea that robotic solutions in agriculture must be coupled with a "robot-ready" orchard to reach their maximum efficiency.

The segmentation network detected few false positives compared to correct inferences, as Table 4.7 noted. The most recurring error consisted of labeling the Other objects as a node, such as various small, round, and point-like objects of the image background. The "other nodes" segmentation as "nodes" mainly included blind buds at the base of longer spurs retained in light pruning treatment. Due to acrotony, distal shoots of an upward spur show preferential growth during the season, inhibiting bud breaking of the lower nodes that lose the possibility of developing new shoots in the next season, even if keeping a relatively similar morphology. The risk of this segmentation is that if the old nodes were counted as real, a pruning algorithm could schedule a wrong cut, targeting a spur instead of a cane.

4.3.2 Pruning point Model

While a quantitative and qualitative evaluation was done on the neural network, we do not have a quantitative analysis of the current grapevine model or pruning point generation module, but it is still possible to perform a qualitative evaluation. Figure 4.2 shows output examples of the modeling algorithm on the left, and on the right shows the estimated pruning points. We can notice how the modeling algorithm performance strictly depends on the

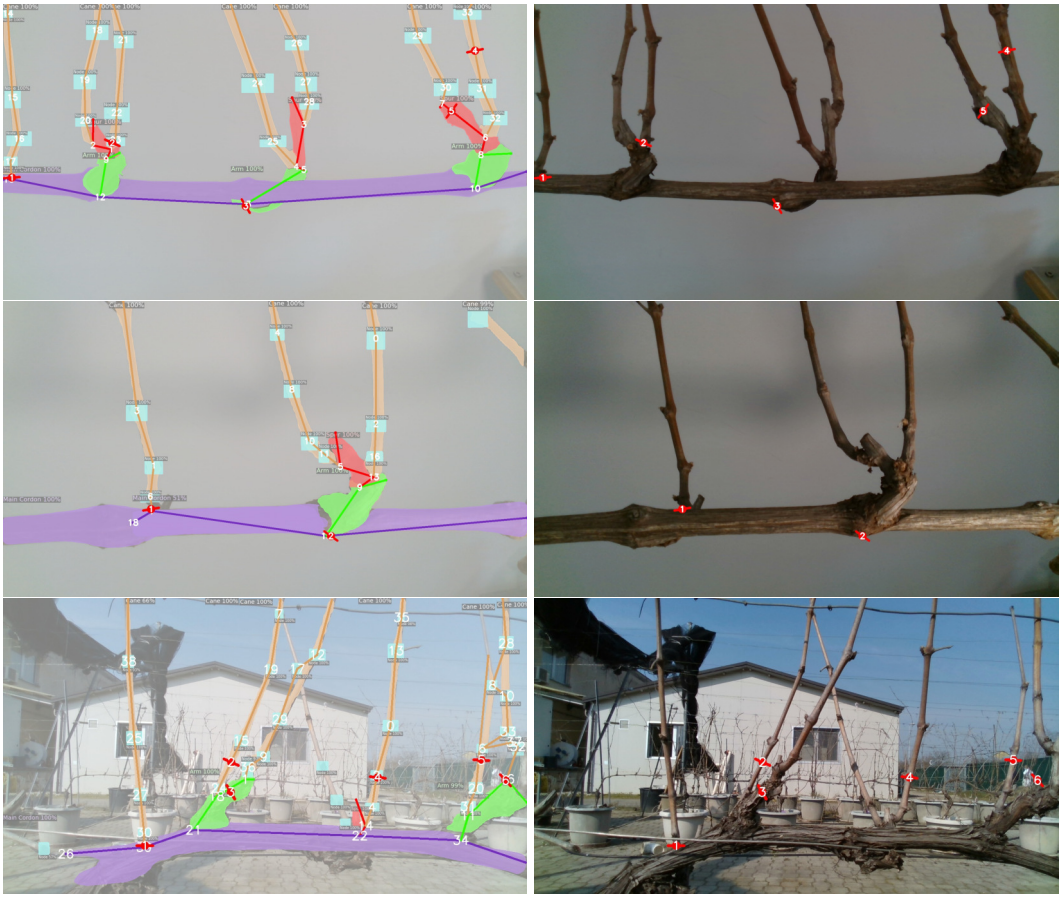

Figure 4.2 Modeling Algorithm Results: in left side of this figure, the models are drawn on the color image to show that it contains all the information needed to compute the pruning points, using a simplified structure and discarding the complications of the raw images. The top left figure uses the ground truth, while the center left figure uses the neural network output, with both cases being in the lab environment. The bottom left figure shows the segmentation neural network operating on the simulated vineyard. In the right side of the figure, we can see how the pruning points' location accuracy decreases as we move from perfectly known cases in the top figure to unknown and challenging cases in the bottom right figure.

detection, recognition and segmentation inference of the grapevine segmentation neural network. In the top left part of Fig. 4.2, we can see how the modeling algorithm operates using the ground truth annotation. As the segmentation masks deteriorate, the plant modeling accuracy also decreases and this can be noted in the center and bottom parts of the figure. However, the modeling algorithm has issues even with ground truth segmentation annotations. This is due to organs being occluded since it is difficult to define the geometries of grapevine organs from a single point of view.

The pruning point detection performance depends on both the segmentation inference and the plant modeling algorithm. This algorithm relies on a set of rules, with its performance being affected by how these rules are understood and implemented. It is also strongly dependent on the depth information, since some of the rules implemented depend on understanding how thick some of the organs are. Its performance heavily depends on the type of depth sensor, since we are targeting small-scale objects, which can be thin to the point that depth information may be unavailable or unreliable.

4.4 Conclusions

We reached closer to an automated pruning system prototype with this system version. In this version, there is a need for a human operator to evaluate the results obtained by the system, including the neural network output, pruning point generation, and the planning of the robotic arm. This includes manually moving the arm into the desired pruning points when the planner cannot find a solution.

Reliable segmentation of grapevines was achieved through a Mask R-CNN network specifically trained to identify five different grapevine organs: cordons, arms, spurs, canes, and nodes. Nodes, arms, and cordons were the best-detected grapevine organs with more than 80% of correct inferences. The overall network's performance massively improved when tested on shoot-thinned grapevines, highlighting the critical role of canopy management in facilitating the introduction of robotic solutions in agriculture.

Although reliable in a way, there is a limitation on the dataset side. The neural network has generalization problems that require the robotic arm to be in a quasiperfect position to detect the grapevine organs properly. This generalization problem is caused by the lack of variety on the dataset since most of the annotation images were taken at a close range, with the cordon close to the bottom of the image and the captured specimens themselves being limited to eight similarly aged plants. The age similarity causes issues with detecting older plants, which have thicker cordons, and younger plants, which have thinner cordons. Due to

the heavy influence of humans in the performed robotic pruning, the results obtained cannot be used scientifically. Regardless, some pruning that the human operator did not influence or "help" was achieved, which can be seen as a development milestone. The neural network, although improved, still has some generalization problems.

The main contribution of this chapter is the conference paper *Fine-tuning and testing of a pre-trained dnn for detecting pruning regions of spur-pruned grapevines.* ((Guadagna, Frioni, et al., 2021)) and the journal paper *Using deep learning for pruning region detection and plant organ segmentation in dormant spur-pruned grapevines* (Guadagna, Fernandes, et al., 2023), with the journal paper being a shared first author. as previously mentioned in Section 1.3.1.

Chapter 5

Three-Dimensional Grapevine Representation with Point Cloud and Pruning Points

Summary

This chapter presents several changes in the system's processing pipeline. The first section introduces three-dimensional point clouds. These point clouds are visual representations of the grapevines that are constructed from multiple points of view. The second section provides results about the pruning season. The third section presents a point clustering method based on the inferences obtained from the neural network.

5.1 Three-Dimensional Segmented Point Cloud

As explained in the previous Chapter, one of the issues of the presented vision pipeline was related to difficulties in detecting some nodes if they are not visible from a frontal point of view. Taking inspiration from humans, a decision was made to implement a method that allows the integration of multiple points of view into just one structure. Point clouds were chosen for this purpose.

It is important to note that the work related to point cloud creation was initially developed by Juan Gamba, which corresponds to the initial code for filtering and unsegmented point cloud merge, while I worked on performance improvement (Section 5.1.3), segmented point cloud merge (Section 5.1.4). In addition, the plant graph and pruning point generation system

developed by Antonello Scaldaferri (Section 5.1.6) was refactored from working in 2D images into working in 3D point clouds.

Before introducing the point cloud, it is essential to introduce the additional annotated data created for the training of the 2D segmentation neural network introduced in Section 4.1.4.

5.1.1 Data Acquisition and Annotation

Considering the previous dataset of only 149 images, we decided to extend it to reach a dataset of 800 annotated images. The 800 captured images were annotated and then split into 80% for training (640) and 20% for validation (160) images, where the validation images are used to monitor the training performance. The additional images are of the same original plants, but the acquisition is slightly different. Instead of a uniform capture of the grapevines, the pictures were acquired from different angles, at different times of day, and with varying weather conditions, as shown in Fig 5.1. No agronomic operations that modify the plants such as cuts were made to the plants during the data acquisition. These new images were captured with an Intel RealSense D435 camera, and the camera placed in the robotic arm is an Intel RealSense D405. The change in the inference camera is related to the depth capabilities of each camera, where the D435 camera is a generalist camera and operates up to 10 meters, and the D405 camera is meant for short-range, operating between 7 and 50 centimeters. The difference in depth capabilities is crucial when working with millimetrically thin organs that need to be measured accurately. The training process of the neural network was not changed except for the additional data.

Figure 5.1 Examples of the added images to the Dataset, with different weather conditions and angles. The first image uses an angle different from normal, where we see the grapevine from underneath it, and it is captured in sunny weather. The second image was captured in cloudy weather, having a more diffuse light present, and the last image, while captured during a sunny day, also contains the sun in the background.

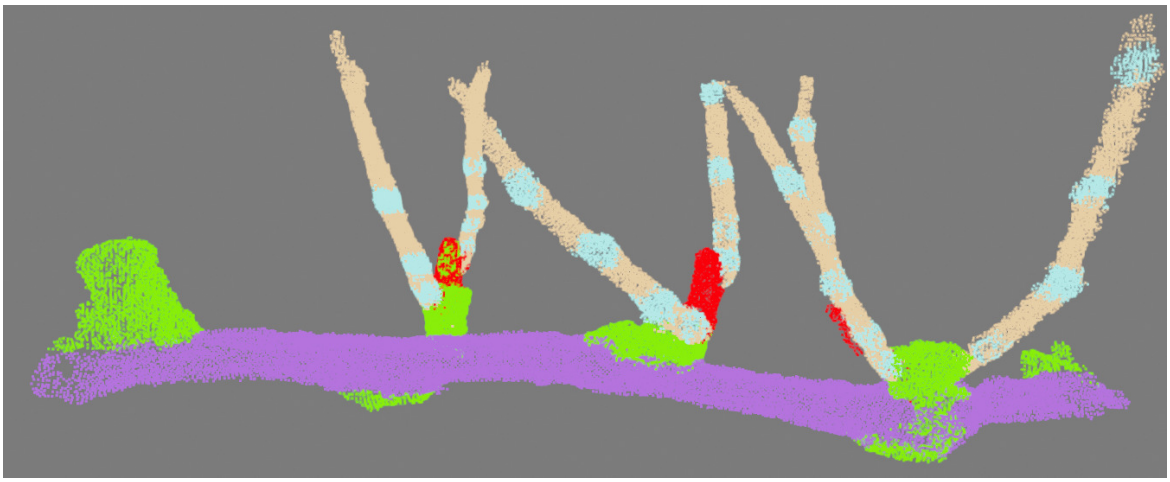

Figure 5.2 Example of a final point cloud. The colors represent the individual classes, with purple representing the cordon class, green the node arm class, red the spur class, orange the cane class, and light blue the node class.

5.1.2 Point Cloud Creation

Perception systems face different difficulties outdoors, like shadow effects, varying light conditions, occlusions, and reflections, which makes it challenging to understand the entire plant's structure from a single perspective. Therefore, collecting and fusing information from different perspectives is suitable for dealing with the commented difficulties. In this sense, we merge the segmented instances from the neural network at every collected view to obtain a 3D representation of the vine. Figure 5.2 presents an example of a merged point cloud created by the system, with purple representing the cordon class, green the arm class, red the spur class, orange the cane class, and light blue the node class.

To create the point clouds, we employ a data structure that includes the following components: a timestamp, an RGB image, a Depth Image, and the Transformation between the camera and a fixed point in the world. These components are synchronized in time. The process begins by applying a segmentation neural network to the first RGB image. The output from the network then undergoes further processing. This processing phase encompasses two primary areas of focus. Firstly, within the segmentation masks domain, several tasks are executed. Initially, overlapping masks of the same class are merged to simplify the output, thereby reducing the number of resulting masks. Secondly, areas of the mask considered too small and indicative of errors are removed. Finally, the depth image is subject to processing. It is filtered using all the segmentation masks, with any value in the depth image falling outside a segmentation mask set to zero. The camera position and orientation are then changed by moving the robotic arm as explained in Section 6.2.1.

5.1.3 ICP & Pose Graph optimization

The segmentation merge is a process done to the data, as mentioned earlier, in which the captured data instances are processed in a way that allows them to be merged into a single point cloud or discarded. Utilizing depth information, we project visual data into a point cloud. By incorporating the transformation information between the camera and the world, we determine the point cloud's spatial relationship to the world. Since we have a matching between the depth image and the color image, it is possible to create a relationship between the color image, the object segmentation, the depth image, and the position of the camera in the world. With this related data it is possible to build a point cloud that only includes the grapevine and also includes each point the object class it belongs to.

We first calibrated the intrinsic camera parameters using the "Dynamic Calibration Tool" from Intel RealSense. Then, we mounted the camera on the robotic arm and calibrated the extrinsic camera parameters using the "Hand-Eye Calibration" package from MoveIt (Coleman et al., 2014). In this sense, it is possible to project a colored point cloud referenced to the camera frame using the intrinsic camera parameters and the depth information. Consequently, we can transform the colored point cloud to the arm base using the extrinsic camera parameters and the arm kinematics. This process is repeated for every captured frame to obtain a "global" point cloud of the plant. However, due to system imprecisions, whether on the camera or robot side or changes in the world, the alignment of point clouds may deviate, making it challenging to overlap multiple point clouds accurately.

To address the commented misalignment issue, we use a two-step strategy consisting of a local optimization to find an approximate homogeneous transformation between frames one and two and successively until closing the loop, and then a global optimization (pose graph optimization) is run on the closed loop to refined the locally optimized transformations from the previous step. To address this, we use a "pose graph" structure to achieve optimal point cloud overlap. The pose graph consists of nodes representing individual point clouds and links indicating the transformations between these nodes. A circular graph needs to be formed to optimize the pose graph; in other words, a graph that consists of a closed chain connecting all the vertices. To achieve this, we employ the Point-to-Plane Iterative Closest Point (ICP) algorithm, (Chen and Medioni, 1992) to calculate the transformations between different point cloud nodes.

To build the pose graph, a pairwise calculation between all the point clouds is normally done. The main issue is that the complexity of the pairwise registration is quadratic ($O(n^2)$). Due to the large quantity of data that is acquired over scanning, a simplified approach to build the pose graph was created.

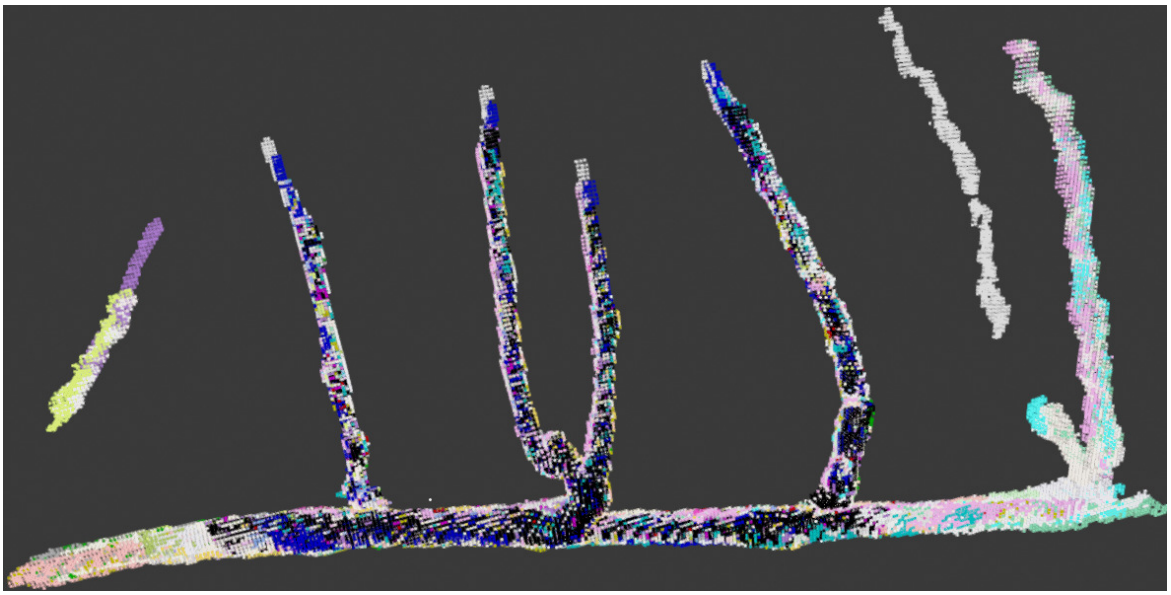

Figure 5.3 Demonstration of the point selection process, where each color represents a different view, where the points in black belong to the first view, and every other random color represents subsequent iterations.

To add nodes to the pose graph we execute the following steps: The first point cloud is added by default and serves as the "root" node. Starting from the second point cloud, we check the overlap between the current and last point clouds in the graph, using the ICP algorithm to determine the fitness metric. If the fitness exceeds 50%, indicating substantial overlap, the current point cloud is added to the pose graph. If not, the current point cloud is discarded, and we move on to the next point cloud and continue the process.

By only creating a closed loop, the complexity of the pairwise registration becomes linear ($O(n)$), with the drawback being that additional checks are needed to understand if all point clouds are fully aligned with each other.

An optimization process is executed after adding all possible point clouds to the pose graph. This process brings all the point clouds closer to the root point cloud, improving the overall fitness of the point clouds.

5.1.4 Point Selection-Based Point Cloud Merging

After obtaining the optimized pose graph containing all the matching point clouds, the next step is to merge them based on segmentation information. To accomplish this, we create several lists that store essential information for each point. These lists include point

coordinates, class, class probability, and node probability, and they are collectively called the final point lists.

This selection of the points that compose the merged point cloud is important to avoid duplicated information. This is caused by the fact that a single point can be seen from multiple points of view. By selecting the more desired points, such as the ones where the neural network is more confident in the result, we can ensure that we have the best available information in a space-efficient manner.

The point selection is executed iteratively, using the vertices of the optimized pose graph. The first element, the root node of the pose graph, is used as the base of the point cloud. The information in this element, the data structure mentioned in Section 5.1.2, is converted into several point clouds, creating one with depth information, a second with class information, a third with probability information, and the last with node probability information, and these point clouds, for performance reasons, are downsampled via voxelization to 0.01 meters.

By creating these four different point clouds, we have the various types of information mapped against the depth information. These point clouds are converted into four separate lists named *positions*, *classes*, *probabilities*, and *node probabilities*. With the first element of the pose graph being used as a base for the final point cloud, the initial lists contain all the points in the element. Starting on the second iteration, initially, the normal of the translation is checked to be inside a threshold, which, if it is not, means that the instance is not correctly aligned with the rest of the vertices and is discarded.

While the fitness of the point cloud may be high enough to keep this point cloud inside the pose graph described in the previous subsection, there is a possibility that this current point cloud is not properly aligned with the rest of the point clouds present in the pose graph, due to the simplification made in the creating the pose graph, as explained in Section 5.1.3.

If it is inside the threshold, a filter based on the point-to-point distance is used, where points are divided into two: points not present in the final point list and those already present. The points that are not present are added to the final point lists. For the present points, we check if the new probability provided by the neural network inference is higher than the current one. If it is higher, we replace the existing point information with further information; else, we discard the new information. This process ensures that the final point cloud contains the most probable information for each point. An example of the final point cloud can be seen in Fig. 5.3.

5.1.5 Point Clustering

Once we have obtained the final point lists through merging, the next step is to organize the points into distinct instances using a clustering method. This allows us to differentiate different objects of the same class in the final created point cloud.

Initially, we implemented the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) from the Open3Dlibrary. DBSCAN relies on two parameters: "*eps*" controls the distance to neighbors for forming a cluster, and "*min_points*" specifies the minimum number of points required to create a cluster. With the parameter "*eps*", it is possible to define how close the points are for them to be considered a single cluster. However, this leads to information being deleted because certain points are not associated with a cluster, for example, in areas where the list of points is more sparse.

We mitigated this issue by transitioning to HDBSCAN (Hierarchical Density-Based Spatial Clustering). HDBSCAN is an enhanced version of DBSCAN that only requires the "min_points" parameter. It automatically adapts the "eps" value based on the data distribution, which allows the clustering to be more adapted to the created point cloud. However, this does not solve the issue entirely since some points are still considered outliers and discarded, while some clusters are wrongly merged into a single cluster.

By clustering the points, we can identify different instances of organs and generate individual point clouds for each organ instance. This step is crucial as it enables us to analyze each organ separately, facilitating estimations such as organ thickness, orientation, and position relative to the rest of the plant. These estimations provide valuable insights for deciding which parts of the plant to retain or remove.

5.1.6 Pruning Point Generation

This subsection explains how the pruning points are generated based on the agronomic rules introduced in Section 1.2 and the rules defined in Section 4.2.2. Differently from before, instead of two-dimensional images, three-dimensional data is used. The above-mentioned rules include the evaluations of several properties: pruning region location, pruning region canes number, basal cane growth direction, basal cane vigor, basal cane origin, and adjacent pruning region distance. The organs are correlated to each other as defined in Table 4.2, where the *cordon* connects to *arms*, *spurs* and *canes*, the *arm* connects to *spurs* and *canes*, the *spurs* connect to *cane*, and the *cane* contains *nodes*.

Following this, important metrics are extracted from the grapevine model to apply the mentioned pruning rules. These metrics include the thickness (also referred to as "vigor") of

the canes, which indicates their health and growth. Another critical metric is the verticality of the canes, which assesses their angle and alignment with the desired pruning pattern.

The cane thickness is determined by dividing the cane element into 20 vertical segments. For each segment, we calculate the Euclidean distance between two points, $(mean_x, min_y, mean_z)$ and $(mean_x, max_y, mean_z)$. Distances that are considered outliers are eliminated and the remaining values are averaged for the final cane thickness estimation. Figure 5.4 shows an example of the thickness being measured in a cane.

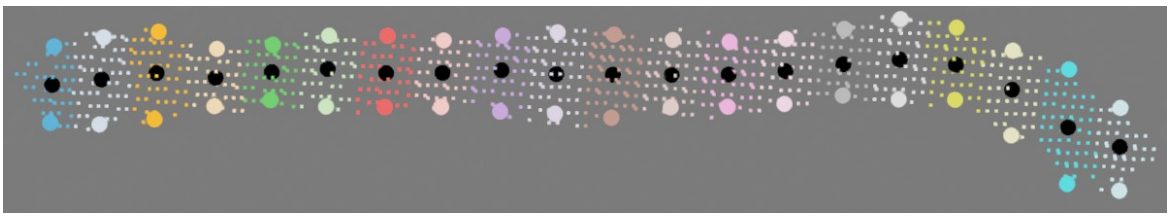

Figure 5.4 Thickness estimation of a cane. The multiple colors are the twenty subdivisions made of the cane, with the spheres corresponding to the extremities ($(mean_x, min_y, mean_z)$ and $(mean_x, max_y, mean_z)$) of the corresponding color segment. The black spheres correspond to the center of the segment, located at $(mean_x, mean_y, mean_z)$. The thickness value of a segment is given by the Euclidean distance between the two extremities.

The verticality of the cane is determined by calculating the angle between two lines. The first line is formed by connecting the center point of the lowest thickness segment with the same point moved vertically by a certain distance. The second line connects the lowest-thickness segment's center point with the highest-thickness segment's center point. By comparing these two lines with a vertical line, we can calculate the angle that represents the verticality of the cane. Figure 5.5 demonstrates the two lines that are used to calculate the verticality of a cane.

The proposed system offers the advantage of flexibility through parameter customization. End users can adjust various parameters to meet their specific requirements. For instance, parameters such as the number of nodes retained on pruned canes, desired cane thickness, and spacing between adjacent pruning regions can be easily modified.

For now, the placement of the pruning point is obtained by selecting the middle point of the line created by the elements being considered for the pruning. For example, in the case of pruning a cane, the elements being considered are the configured count node and the node above it, with the pruning point being placed on the midpoint between the two nodes.

In another example, the spur cut, the elements being considered are the two canes, where the pruning point is placed on the midpoint between the lower extremities of the two canes.

A visual explanation can be seen in Fig. 5.6.

Figure 5.5 The estimation of the verticality of the left cane is given by the angle defined by the two red lines, with the left line connecting the both extremities of the cane, and the right line starting on the lowest extremity and moving upwards an arbitrary distance.

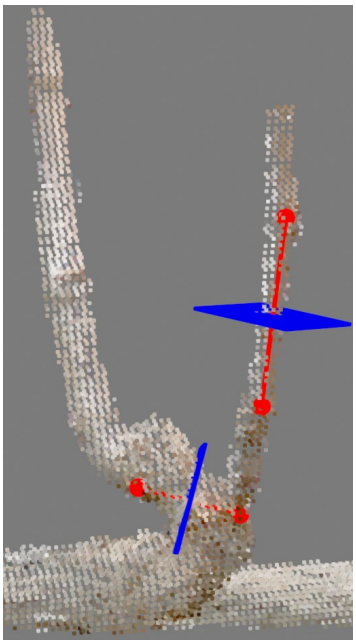

Figure 5.6 Demonstration of the pruning point placement, where the pruning points are shown by the blue rectangles, and the red spheres show the organ points considered. For the upper pruning point, the red spheres are located in the nodes' center. For the lower pruning point, the red spheres are placed in the lower extremity of the two canes of the pruning region.

5.2 Pruning Experiments and Results

The vision pipeline and pruning point selection explained in Section 5.1 has been experimentally validated in the pruning season of winter 2022/2023. The experiments were performed with the robot prototype Summit XL presented in Section 2.4.2. We used the robotic system in the simulated vineyard to test the system in a real-life environment to prune several grapevines, some that were hand-pruned the previous year and others that were pruned with the previous version of this system throughout the pruning season in winter 2022/2023. As mentioned in the previous chapter, since the plants were robot-pruned, without considering the correctness of the pruning, some of the evaluated pruning regions are considered more complex than usual. The evaluation of the pruning results was done by Filippo Graziosi and Prof. Matteo Gatti from Università Cattolica Sacro Cuore di Piacenza.

5.2.1 Experimental Setup

The execution process for the experiments was as follows. The robotic arm is positioned so that the camera faces the pruning regions, with the camera field of view centered on either the arm or spur structure of the pruning region. The scanning movement (See Section 6.2.1) builds the grapevine point cloud model. The grapevine point cloud model is analyzed and the pruning points are generated. Although the system can perceive the pruning regions surrounding the current pruning region, we only consider the pruning points that affect the scanned pruning region.

It is important to note that the target plants are part of an ongoing study comparing robot and human-pruned plants. The plants are pruned regardless of the correctness of the system-provided pruning points, which can cause irregular plant growth over the next season and affect the following year's pruning season.

5.2.2 Experimental Results

The main result we have to present in this section is the evaluation of the pruning performed by the system in the pruning season of winter 2022/2023. The pruning methodology can be categorized into four main types: clean cut, base-bud cut, replacement cut, and spur cut. The base-bud cut and spur cut involve making two physical cuts on the grapevine. A description of the cuts can be found in Section 1.2. The pruning points obtained by the system are compared with the correct pruning methodology (Table 1.2) that would be applied by the

Table 5.1 Results obtained by the system during the March 2023 pruning season, along with information on whether additional manual pruning can fix the case.

	Number of Cases	Percentage Overall	Can be fixed?
Correct Cut	7	28.2%	No Need
Incorrect Length	3	12.0%	Yes
Incorrect Cane	5	20.0%	No
Incorrect Case	9	36.0%	No
Incomplete Pruning	1	4.0%	Yes
Total	25	100%	

experts, where we compare if the selected methodology is correct and if the placement of the physical cuts is correct.

These results are classified into five different classes. *Correct cut*, *Incorrect length*, *Incorrect Cane*, *Incorrect Case* and *Incomplete Pruning*.

As the name implies, the *correct cut* is an agronomically correct pruning performed on the pruning region, as seen in Fig. 5.7. The *Incorrect length* is similar to the correct cut, where the error was the incorrect shortening of the cane, where the system keeps three nodes instead of two nodes, generally due to the incorrect clustering or detection of the nodes present in the cane, which can be seen in Fig. 5.8. The *Incorrect cane* happens when the incorrect cane is kept on the plant, and the correct cane is removed. This case occurs typically when the verticality of the cane is wrongly calculated, or the position where the cane starts to grow is incorrectly placed. Figure 5.9 presents this case. *Incorrect case* is where the system misclassifies the pruning region, leading to the wrong pruning points. This case has multiple causes, some instances by an incorrect plant segmentation that leads to an incorrect point cloud reconstruction, others by incorrect cane thickness estimation; one such example is seen in Fig. 5.10. Lastly, *Incomplete pruning* is an edge situation where the system does a correct cut but does not process the entire pruning region, which is caused by a glitch implementation of the pruning point generation algorithm. Table 5.1 contains the quantitative results obtained by the system in the performed field tests over the pruning season, along with the information if the obtained case can be fixed by additional, manual pruning. It is essential to note that the number of cases does not correspond to the number of cuts since some of the cases, such as the *spur cut* or the *base-bud cut* imply performing two cuts, as mentioned in Section 1.2.

Several conclusions can be extracted from the results that explain what triggered those wrong results. On the incorrect length cases, all the instances are faulty due to not cutting

Figure 5.7 Example of a correctly calculated cut, with the expert evaluation on the left side of the image, with the plant designation on the top left, with the cut type designation on the bottom left, in this case, Spur Cut. The numbers next to the pruning points are designations for each pruning point, where the format is *PlantNumber.PruningRegion.CutOrder*. The cut order is important, with the pruning order allowing a fail-safe in case of an accident while performing the pruning. The right image shows the point cloud output, with the blue markers representing the cuts. Text close to both blue markers states the name that the system gave to the pruning points.

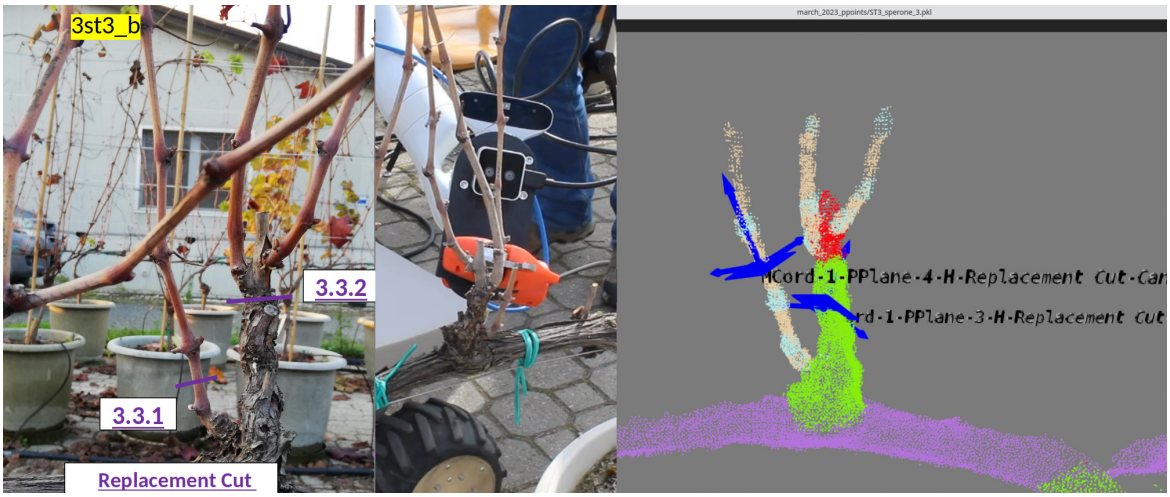

Figure 5.8 Example of a wrong length cut, with the expert evaluation on the left side of the image, with the plant designation on the top left, with the cut type designation on the bottom left, in this case, Replacement Cut. The numbers next to the pruning points are designations for each pruning point, where the format is *PlantNumber.PruningRegion.CutOrder*. The cut order is important, with the pruning order allowing a fail-safe in case of an accident while performing the pruning. The right side shows the point cloud output, with the blue markers representing the cuts. Text close to both blue markers states the name that the system gave to the pruning points. The center part of the image shows the robot right after performing the cut with the wrong length.

Figure 5.9 Example of a wrong cane cut, with the expert evaluation on the left side of the image, with the plant designation on the top left, with the cut type designation on the bottom left, in this case, Spur Cut. The numbers next to the pruning points are designations for each pruning point, where the format is *PlantNumber.PruningRegion.CutOrder*. The cut order is important, with the pruning order allowing a fail-safe in case of an accident while performing the pruning. The right shows the point cloud output, with the blue markers representing the cuts. Text close to both blue markers states the name that the system gave to the pruning points. This cut, in particular, is wrong due to selecting the wrong cane to be kept. Although not counted as such, this specific example could be fixed since the cut performed on the spur is correct, which removes the correct cane, with the fix being the shortening of the remaining cane.

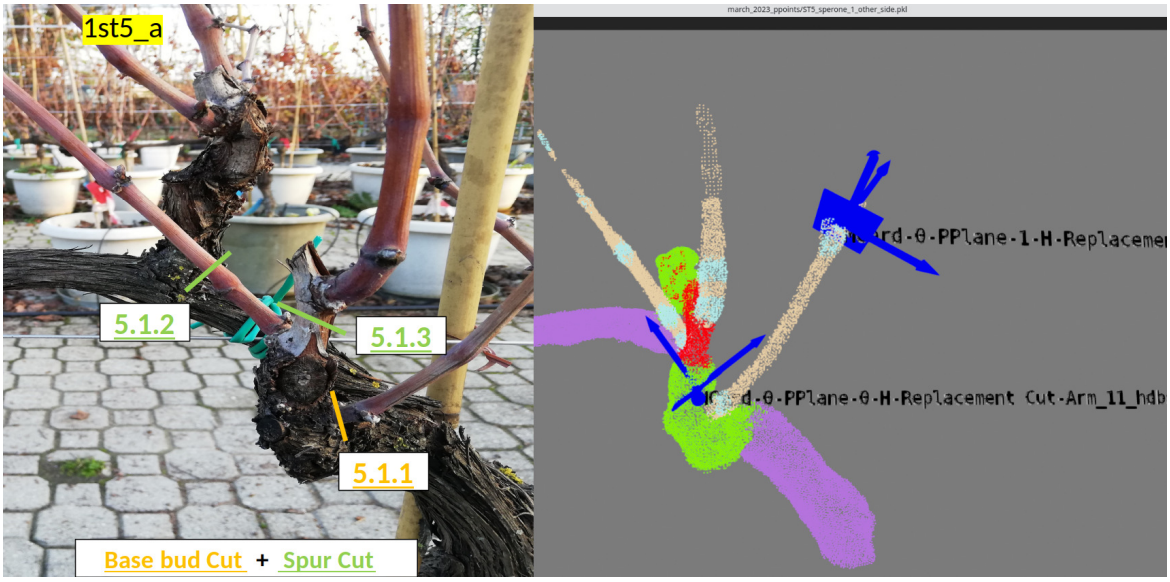

Figure 5.10 Example of an incorrect case, with the expert evaluation on the left side of the image, with the plant designation on the top left, with the cut type designation on the bottom left, in this case, Base bud Cut and Spur Cut. The numbers next to the pruning points are designations for each pruning point, where the format is *PlantNumber.PruningRegion.CutOrder*. The cut order is important, with the pruning order allowing a fail-safe in case of an accident while performing the pruning. The right side shows the point cloud output, with the blue markers representing the cuts. Text close to both blue markers states the name that the system gave to the pruning points. This cut, in particular, is wrong due to the inaccurate thickness estimation of the lower cane, which the system estimates as 10mm but in reality it is thinner than 7mm.

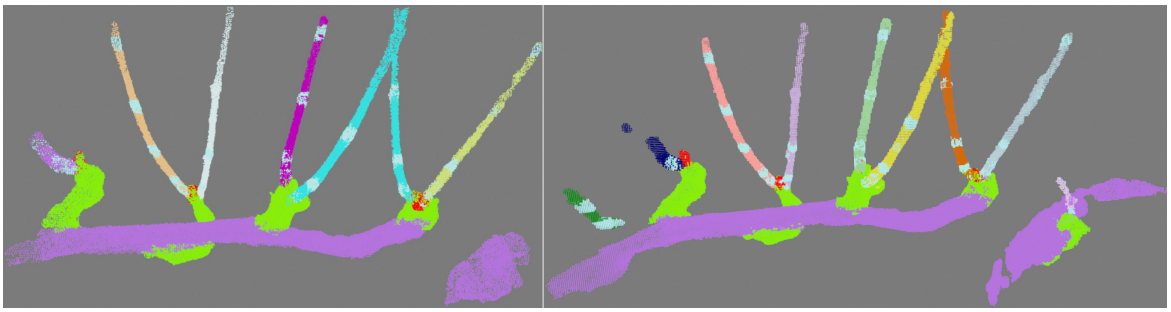

Figure 5.11 Demonstration of the difference in the clustering method. Both images represent the same grapevine specimen, where the left grapevine uses the point-based clustering method, and the right specimen uses the new inference-based clustering, along with some improvements related to logic rewrite. To aid the visualization, the cane elements of both grapevines are randomized by instance, and on the left grapevine, the cane in cyan is just one element; with the new clustering method, the same cane is identified correctly with yellow and dark orange colors.

enough of the cane rather than removing too much of the cane, which allows all these cases to be corrected via additional manual pruning. We can also identify that the problem in these specific cases is either the wrong clustering of nodes close to the cane base that leads to the false identification of two nodes as just one instance or the nonidentification of one of the present nodes. Regarding the cases where the system selected the wrong cane to keep, most of the situations come from an incorrect distance measurement between the individual canes and the cordon. More specifically, we considered the absolute height of the element instead of its distance to the corresponding cordon element.

5.3 Inference-based Clustering

The incorrect clustering of points independent of the object class causes some pruning points to be considered wrong during the March 2023 pruning season. On the nodes, the clustering merges separate nodes, while on the canes, the problem is the merging of overlapping canes that belong to different pruning regions. This issue is mentioned in Section 5.1.5, where points are clustered together due to being close to each other, which is the expected behavior for the clustering method. The issue stands on organs, normally canes, that naturally can touch each other and overlap. When this happens, a normal point clustering method just considers everything as one group.

To correct this outlier situation, I implemented inference-based clustering, by taking advantage of instance information already provided by the neural network. The challenge is

that the identifier for an object changes between every instance. For example, a cane with identifier 5 in the next frame might be cane 9. The way to overcome this is by merging instances with a high enough Intersection over Union in the 3D space and creating new instances if the current instance is not present in the final point cloud.

In pursuing this solution, the function that manages the addition of new points to the final point cloud was rewritten, simplifying the merge of new points to the final point cloud. Using tensor point clouds offered by Open3D, it is possible to add attributes to each point. By allowing the addition of attributes, point cloud simplification methods such as downsampling do not affect the overall creation of the point cloud.

This change also helps with memory management and speeds up the point cloud creation. An example of the result created by the new inference-based clustering can be seen in Fig. 5.11.

5.4 Conclusions

As a global evaluation of the results obtained from the winter 2022/2023 pruning season, as shown in table 5.1, they present a favorable evolution from the previous year's results since the system performed automated pruning without human aid, where aid refers to finding the best favorable situation for the pruning region. The only aid provided was to the spur cuts, where the arm, due to overzealous collision constraints, could not place the shears in the provided pruning point. In those cases, the robotic arm was manually moved to the exact position offered by the system and then pruned.

Considering the qualitative evaluation of the results, although the positive success was only 30.2%, considering the entirely correct cuts and the incorrect length cuts that can be fixed. By analyzing the wrong cuts and the reason to lead to the incorrect evaluation, the system can be improved on the logic of how to obtain the right pruning points by correcting the code logic that generates them.

One of the issues that caused wrong pruning is the error of the camera itself, which has a depth estimation error of 2% at a distance of 50cm (according to the datasheet). This causes the system to have issues with the estimation of the thickness of the canes, leading to several cases of performing the wrong cut altogether. Using the same error of 2% at the normal range of operation of 30cm, corresponds to an error of +/- 6 millimeters. The *canes* are one of the organs that need to be measured to decide if they are meant to be kept or removed from the plant, where we keep the canes that are between 7 and 18 millimeters. In the specific situation of the presented results, four of the nine incorrect cases were caused by

the inaccurate cane thickness estimation, by keeping a cane thinner than seven millimeters and pruning the rest of the region. An example of this case can be seen in Fig. 5.10. With this information, a future study on mitigating or removing this estimation error is needed, either by further calibration or by changing the way the thickness estimation is done.

The development of inference-based clustering methodology addresses the challenges related to incorrect point clustering, particularly on nodes and canes and demonstrates promising results in the test in the laboratory. It will be evaluated in the next pruning season. By leveraging instance information from the neural network, the system now merges instances with a significant Intersection over Union in the 3D space. This adaptive approach ensures accurate clustering even when object identifiers change between frames.

I am currently preparing a manuscript about the results presented in this chapter to be submitted to the journal Computers and Electronics in Agriculture with the tentative title: *Integrating 2D Segmentation with 3D Point Clouds for Pruning Point Generation in Grapevine Winter Pruning.*

Chapter 6

Arm Manipulation

Summary

This chapter presents the pipeline created for the system, along with the manipulation tasks created. This pipeline is used as the execution principle of the system used throughout the thesis, having two major different versions, the first one for Chapters 3 and 4, and the second for Chapter 5. In addition, work performed related to the arm planning and movement was created by Juan Gamba, with me later fixing minor issues and implementing some features (Section 6.2.4).

6.1 Execution Pipeline

With the modules presented in Chapter 4, an execution pipeline is devised to be used for autonomous pruning. Figure 6.1 shows a schematic representation of the pipeline. This pipeline starts by navigating the platform inside the row to the first plant to be processed. The robotic arm is then deployed, and the system performs the spur perception and detection. The potential pruning regions are found via the trained segmentation neural network, and selecting the inferences that are either *spurs* or *arms*. The robotic arm then approaches the first detected pruning region and performs segmentation of the pruning region itself, this time being able to discern better the smaller nodes present in the canes. With this information, the grapevine model is created, the pruning points are generated and converted into world coordinates, and the arm approaches those points. After performing the pruning, the system continues to the next pruning region and continues the process until the grapevine is pruned. Due to the change from 2D segmented images to 3D point clouds, presented in Chapter 5,

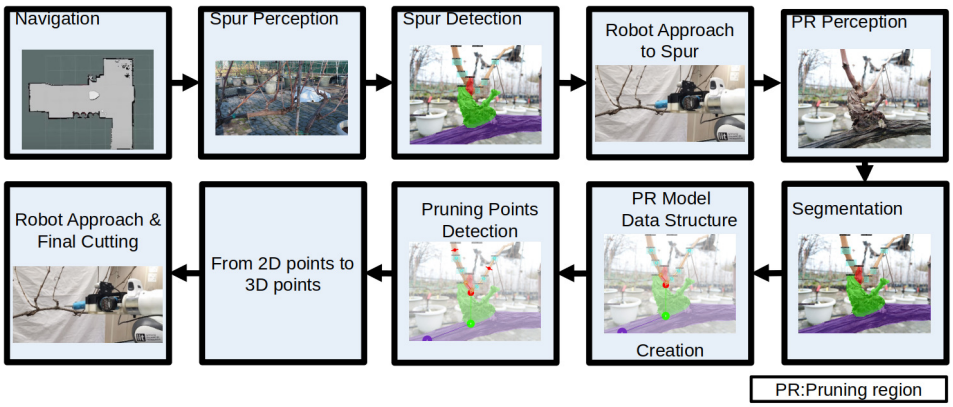

Figure 6.1 Execution pipeline for the complete system, contemplating the navigation, perception, and robotic arm planning.

the pipeline needs to be updated. Considering the modules mentioned earlier and additions, the current pipeline does not consider the navigation of the platform, starting instead by deploying the arm and performing the pruning region detection, done the same way as before, by using the neural network and selecting the inferences that are either the spurs or arms, since these compose the pruning regions. The robotic arm then approaches the first detected pruning region and executes the *plan to scan* task described in Section 6.2.1, and, while the arm is moving, the system captures data for the point cloud creation module. After finishing the movement, the captured data is processed as described in Section 5.1.2. With the creation of the segmented point cloud, the system then uses the *plan to cut* and the *plan back* tasks to prune the obtained pruning points.

6.2 Arm Planning and Movement

This section explains the arm motion planning and movements. A framework named MoveIt Task Constructor (Görner et al., 2019) that simplifies the process of planning, executing, and monitoring complex manipulation tasks for robots was used. It achieves this by providing a high-level interface for task planning and execution. Our pruning pipeline uses four different tasks. The first task, seen in Fig. 6.3, scans a pruning region to aquire the information needed to capture the data for building the point cloud. The second task, shown in Fig. 6.4, uses a pruning point and plans a movement that allows the engagement of the pruning shears in the desired point. The third task entails only the disengagement from the plant and moving into a known joint position, such as fold position, where the arm is folded on top of the mobile robot safely, allowing the prototype platform to move without damage to the robotic

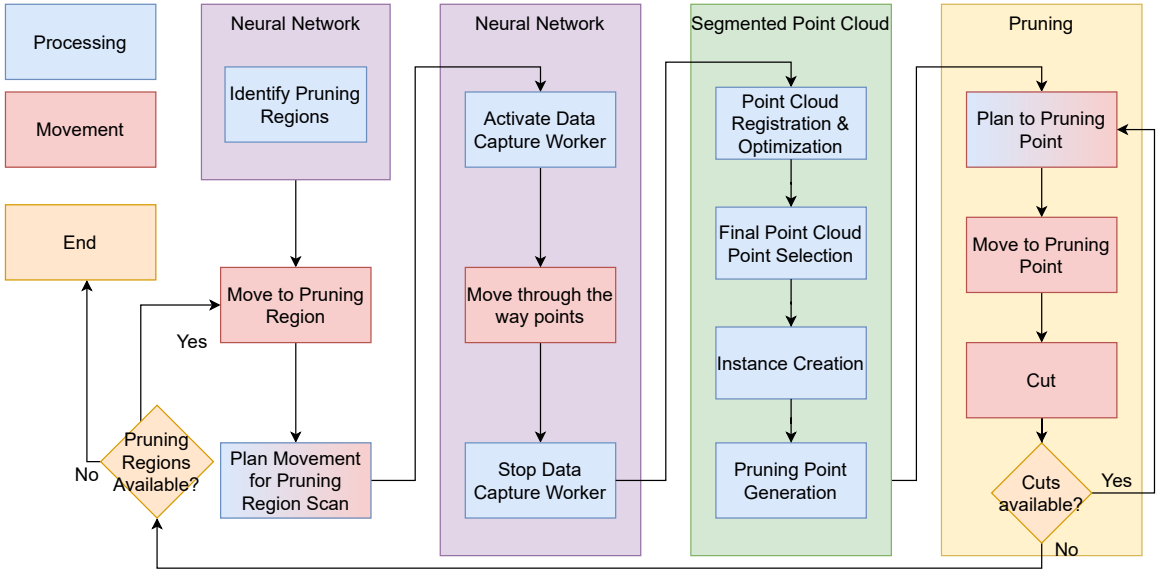

Figure 6.2 Pipeline of the system execution loop, from the initial pruning region identification step up to the cutting.

arm. The fourth task is a composition of the third task, where instead of returning to a known joint position, the arm moves to the next provided pruning point. Each task will be more thoroughly explained in the following subsections, but they share some parameters; for example, their cost function is not a metric of time but rather a metric of how much the end effector moved. This metric was selected to prevent the arm from planning and executing motions that move the shears away from the plant and towards the human operator or other bystanders. I consider any movement that keeps the shears near the grapevine and its working area better than a faster movement that can potentially threaten a nearby person.

6.2.1 Task N°1 - Plan to Scan

This task creates a plan that scans the selected pruning region. The scanning motion, which can be seen in Fig. 6.3 involves the following poses: The robotic arm initially orients itself toward the designated pruning area and executes a square motion pattern centered around the target object within the image. This motion comprises a sequence of distinct poses involving rotational adjustments. It commences with a 10-degree leftward rotation, a 5-degree downward tilt, then a 10-degree rightward rotation. Subsequently, it repeats a 10-degree rightward rotation, followed by a 5-degree upward tilt, and concludes with a final 10-degree leftward rotation to reach the initial position. Considering that Cartesian planning can sometimes fail in more complicated motions, MoveIt Task Constructor allows the usage

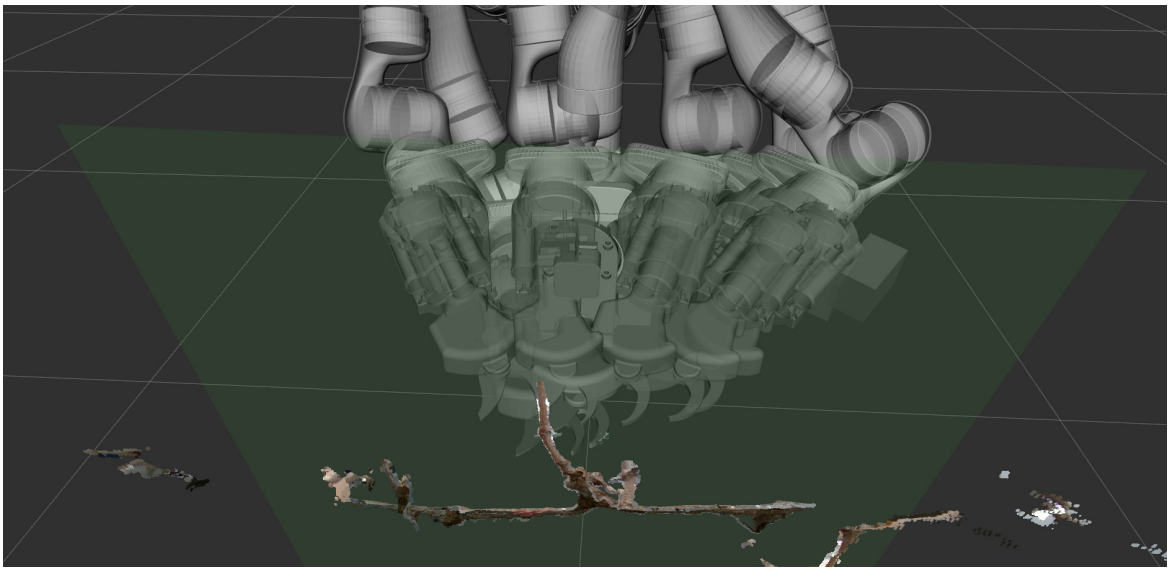

Figure 6.3 An example of the planned movement, with a green plane as an obstacle preventing the planning of movements that hit the grapevine and shadows representing positions where the robotic arm will move through during the scanning motion.

of *Alternatives*, which enables the planning with multiple types of planners, where in this case, it plans using the Cartesian planner, joint interpolation planner, and sampling planner. By using these planners in parallel and selecting the one with the lowest cost, the final motion plan has a combined lower cost. In short, Cartesian planning is used with desired positions and orientations of the end effector using Cartesian coordinate space (X,Y,Z). Joint interpolation planning involves specifying the desired positions for each joint of the robot arm. The planner then computes a trajectory that smoothly interpolates between the initial and final joint configurations. Finally, the sampling planner randomly samples configurations and constructs a graph or tree connecting the sampled configurations to find feasible paths for a robot to move from its initial state to the goal state.

6.2.2 Task N°2 - Plan to Cut

Plan to Cut is the task used to plan a movement to a pruning point, as seen in Fig. 6.4. It receives the pruning point we desire to prune. It is a pruning point rather than a pruning pose because, from the vision side of the system, there is an abstraction of the used arm and end effector. The difference between a point and a pose in this situation is the fact that a point has (X,Y,Z) coordinates, and a pose has the same coordinates (X,Y,Z) but it also has an orientation, in quaternion format (X,Y,Z,W). In robotics quaternions have advantages such

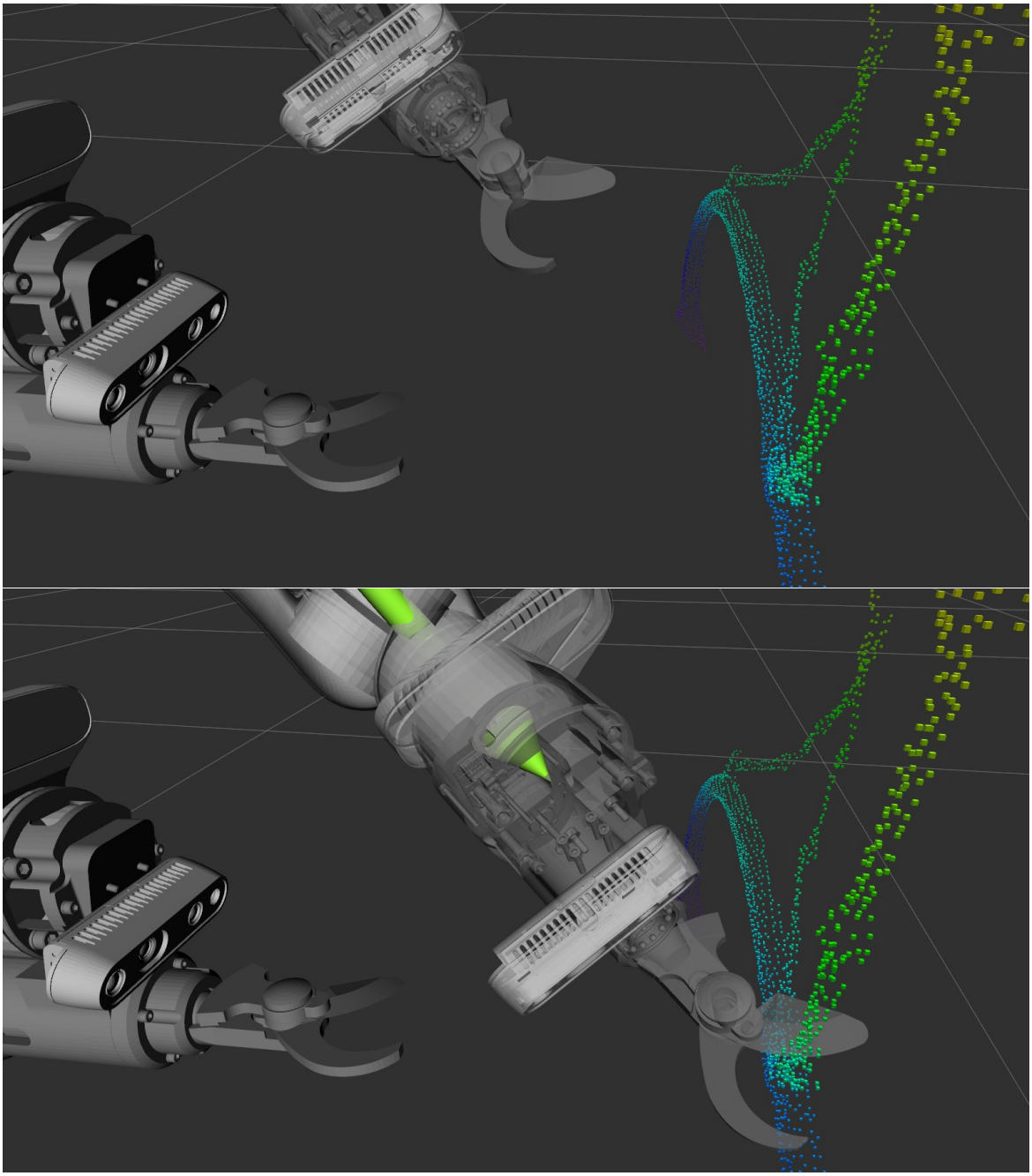

Figure 6.4 Task *plan to cut* in parts, where the ghosts are positions where the robot will pass. The top part of the figure is where the robot aims towards the pruning point, and the bottom is where the robot engages the pruning point.

as to avoid singularities, situations where a specific configuration of the robot's joints causes a loss of one or more degrees of freedom.

With this abstraction, it is possible to use the same mechanics and concepts as the grasping research community uses, applied to the pruning task. This is because pruning is just a grasping task; instead of grasping, we cut the intended target. Due to the need to validate the feasibility of the end of the motion, the plan is calculated inversely from the execution. With this, the first thing this task does is verify via the inverse kinematics if the end effector, or, more specifically, the center point of the shear blades, can be inserted into the provided pruning point. This discards the positions of the end effector that either touches the plant or tries the pruning point from the opposite side of the plant. These end effector poses are obtained by searching around the provided point, with an angular delta around the three coordinate frames, X,Y and Z, and by using code from MoveIt Task Constructor related to grasping tasks. After finding valid positions, a relative engagement movement controls the arm to the validly generated pose, with a distance from the pose between 5 and 15 centimeters. This is first planned with Cartesian planning, which falls into joint interpolation if the former fails. The last part plans the motion from the current state of the arm to the position of the beginning of the relative engagement movement.

6.2.3 Task N°3 - Plan Back

This task uses as input a set of joint positions, which are generally the joint positions when the robotic arm is deployed into the detect position or is folded on top of the robot, where the former is used for moving to the next pruning point or next pruning region and the latter to stow the arm during platform navigation. The task comprises two phases, where a relative movement is done backward, in a range of 5 to 15 centimeters, followed by the move to the provided joint positions.

6.2.4 Task N°4 - Plan to Cut Sequential

This task consists of a variant of the second task, *Plan to Cut* from Section 6.2.2. It is meant to be used when already engaged in a pruning point, and the system plans a movement that disengages from the current pruning point, aims to the next pruning point, and re-engages it. This change saves time by removing the unneeded movement of the arm to and from the detect position, along with reducing the search space of the planning.

One of the recurring situations with the presented tasks, is the Cartesian planning failure, where the motion planner was unable to fully complete a motion from point A to point B due

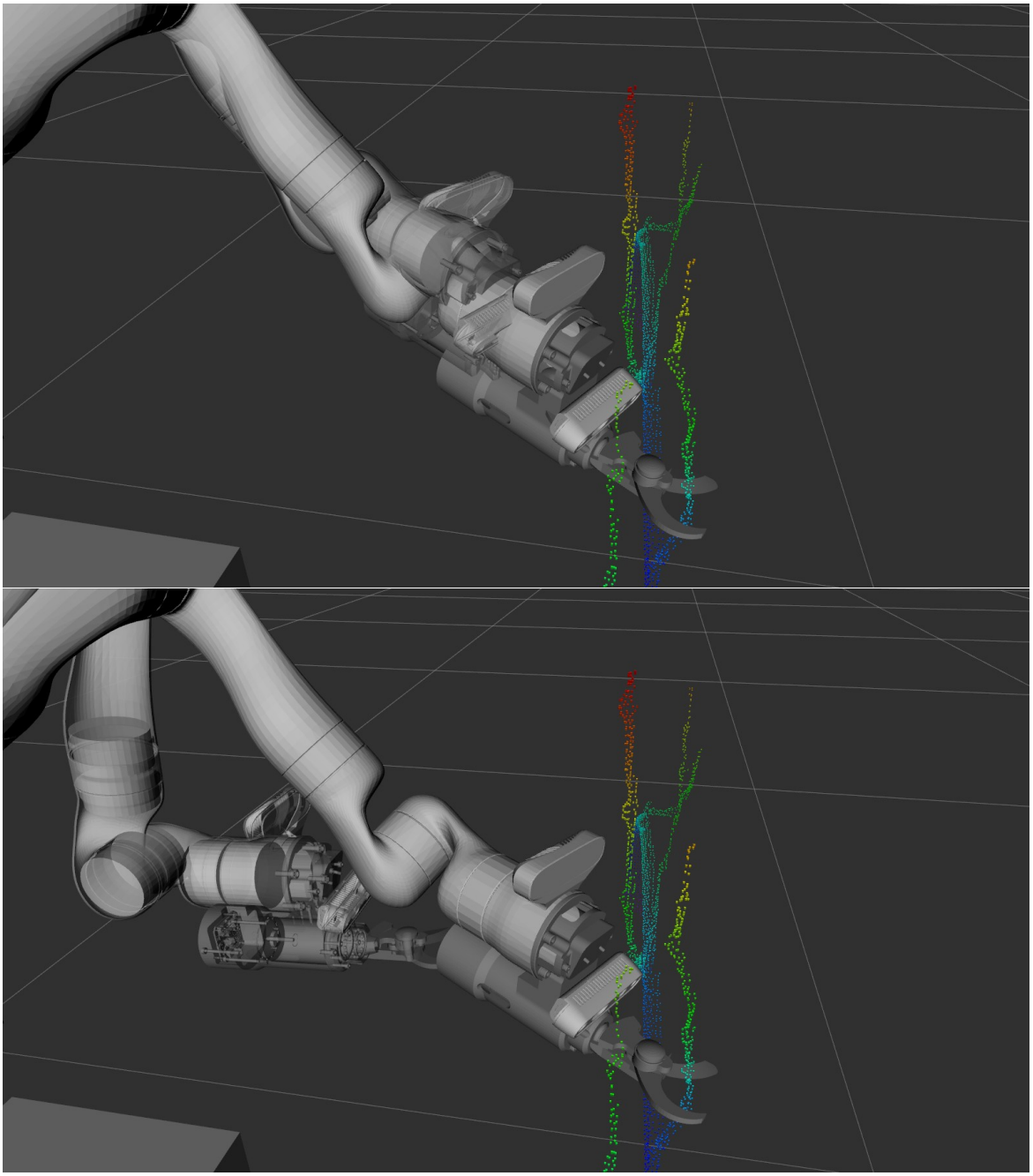

Figure 6.5 Task *plan back* in parts, where the ghosts are positions where the robot will pass, the top part of the figure is where the robot disengages from the pruning point, and the bottom is where the robot returns to the detection position.

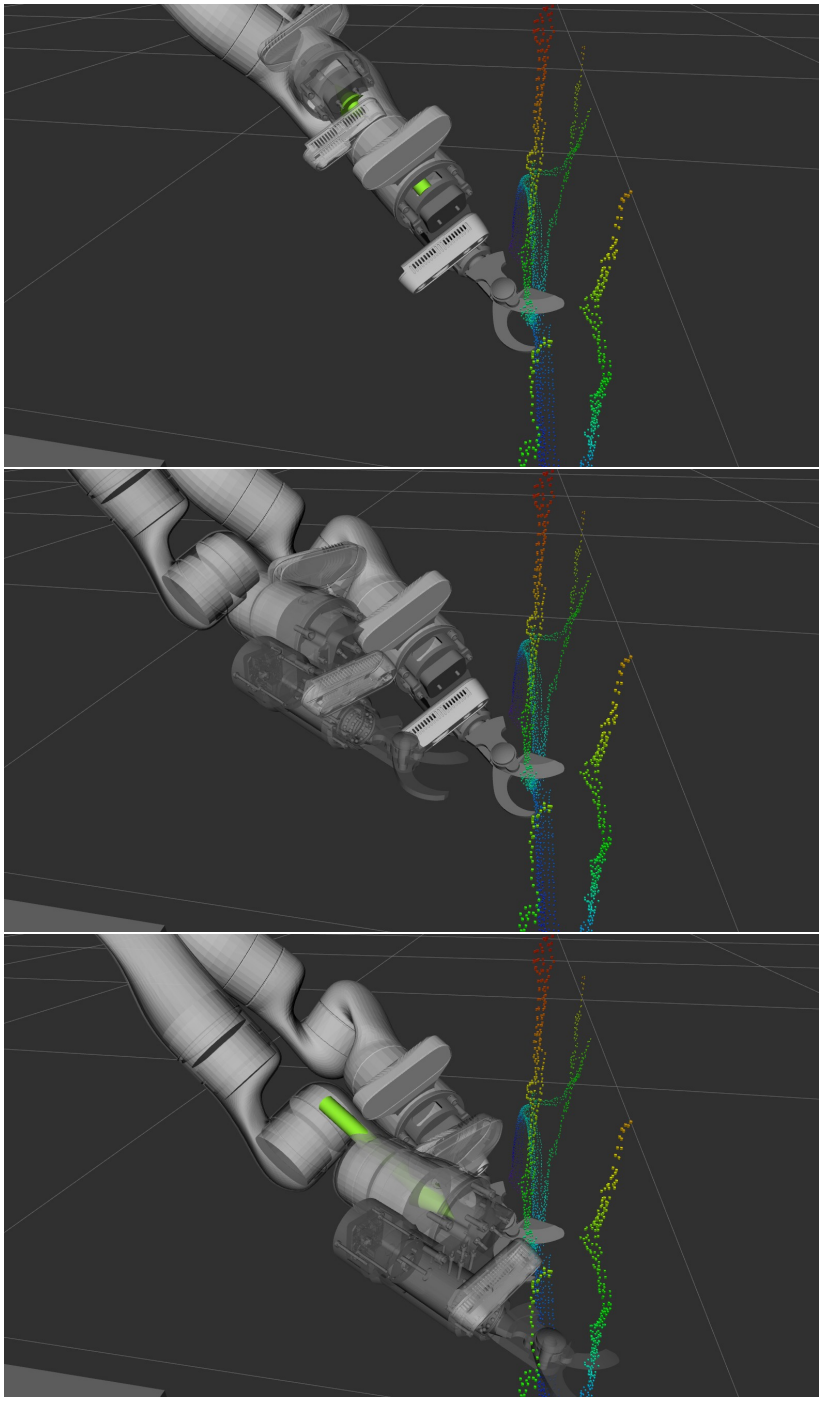

Figure 6.6 Task *plan to cut sequential* in parts, where the ghosts are positions where the robot will pass, the top part of the figure is where the robot disengages from the current pruning point, the middle shows the re-aiming towards the new point, and the bottom is the engagement to the new pruning point.

to either planner limitations or other robot limitations, a situation also reported by the authors in Silwal et al., 2022. With this, I decided to add fallback planners to every task mentioned before and any future task created. Now, by default, any task that used Cartesian planning, in case of failure, falls back into a joint interpolation planner. With this change, the system is more robust, allowing the planning functions to reach successful plans that can be executed.

6.3 Conclusions

This chapter has defined the two versions of the execution pipeline, with one relating to the 2D version of the system (Chapters 3 and 4) and the second version for the 3D point clouds (Chapter 5). It defines arm planning tasks, that define a set of movements essential for scanning and for moving to the pruning points.

Issues were found with the generation of movements using Cartesian planners, with these issues being mitigated by implementing a fallback that uses the interpolation planner. When the Cartesian plan fails to complete a motion, the fallback to joint interpolation planning ensures task execution. Additionally, the new planning task, *Plan to Cut Sequential*, optimizes arm movements by reducing unnecessary arm movements and streamlining the planning process, resulting in improved time performance. Both these improvements contribute to the overall robustness and efficiency of the system.

Regarding the results presented in Table 5.1, an initial evaluation of the pruning was made from the manipulation side. From the 25 cases, all the cases handle pruning regions with a minimum of two canes. As such, we performed 50 cuts; from those 50 cuts, one half is performed on canes, and the other half is performed on spurs or arm structures. The system could approach all the cane pruning points autonomously. A study on the accuracy of end position was not made, although the robot always approached the correct cane segment to prune, where the correct segment is comprehended between the second and third nodes of the cane to be kept. Due to issues with implemented collision avoidance, the system could not perform motion plans for the cuts placed in spur or arm structures.

Chapter 7

Conclusions

7.1 Summary

This thesis explored creating an automated robotic pruning system for grapevine winter pruning that uses a simplified structure of a robotic arm, an end effector equipped with pruning shears, and a depth camera, with the benefit of being platform agnostic. The results presented in this thesis represent the culmination of the work produced over the last three years. It encompasses various disciplines such as agriculture, computer vision, motion planning, and neural networks. Although an advanced stage was reached, the presented system needs improvements to several components.

The second chapter presented related work that inspired and guided the creation of the proposed system.

The third chapter presented an initial prototype pipeline that tested the feasibility and whether the proposed system does match the actual real-life needs.

The fourth chapter extends the prototype system by correcting some fundamental flaws of the system, such as the grapevine organs that need to be identified and starts to consider the pruning procedures from an agronomic perspective, with the culmination of a working prototype that can perform segmentation and prune a grapevine, although with human aid.

The fifth chapter considers several limitations of the vision system, such as occlusion, and introduces a method to create a 3D segmented point cloud based on multiple views created with 2D segmentation.

The sixth chapter addresses the execution pipeline and it introduces MoveIt Task Constructor, the framework used to simplify the process of planning, executing, and monitoring complex manipulation tasks for robotic arms.

7.2 Future Work

Future work would pass by multiple areas, one related to the neural network and synthetic dataset, a second referring to a depth camera accuracy issue, and a third one for improving the efficiency of the robotic arm planning.

The first significant issue concerns the neural network and the training dataset. Datasets for neural networks require extensive time to capture data and to create annotations. Coco Annotator, the tool used for the annotation process mentioned in Section 3.1.3, gives some statistics. One important statistic is the time taken per image, where each image takes an average of sixteen minutes, which, for 800 photos, corresponds to 213 hours, which is equal to 8 days and 21 hours. This is a relatively large amount of time for an expert in agriculture to spend annotating images, and the dataset still requires more data to generalize correctly. An idea to aid in creating annotated data is the generation of synthetic data. The synthetic data can be made from the already implemented neural network and generated point clouds.

As mentioned in Section 4.4, the neural network has generalization issues, and, taking inspiration from Chapter 2, one important area to work on is synthetic data generation.

Converting the existing point clouds into meshes makes it possible to generate more data automatically. Since it is possible to render the created meshes from different points of view, with the addition of generating the annotations automatically, we can move the burden from the agronomic expert towards the machine. This synthetic data generation could also allow research in the segmentation of 3D point clouds, instead of the currently used multiple 2D views to a 3D point cloud.

A second point should focus on the issue reported in Section 5.4, related to the depth camera sensor accuracy which affects the pruning calculations, by introducing errors in the vigor estimation of the grapevine organs.

A third point of future work is related to robotic arm planning, including its reliability and planning and execution time, which has to achieve human-like pruning time to reach a degree of real-life usability. A way to improve the system's time performance is to capture the entire grapevine instead of just targeting one pruning region. With this, it is possible to calculate the result of the plant pruning, along with the total number of nodes kept after pruning, with the addition of having a record of how the plant looked before pruning, which is information necessary to the agriculture experts exploring the vineyard. Another time-saving component to capturing the complete grapevine is planning a full motion that simultaneously targets all the pruning points, possibly optimizing the movement time.

References

Bargoti, S., & Underwood, J. P. (2017). Image segmentation for fruit detection and yield estimation in apple orchards. *Journal of Field Robotics*, *34*(6), 1039–1060. https://doi.org/10.1002/rob.21699

Betty. (2010). Mechanical pre-pruning in the vineyard [Accessed: 2023-12-10]. https://binghamfamilyvineyards.com/mechanical-pre-pruning-in-the-vineyard/

Billo. (2022). Vitis [Accessed: 2023-12-13]. https://www.billosrl.it/portfolio-items/vitis/

Borianne, P., Borne, F., Sarron, J., & Faye, E. (2019). Deep Mangoes: from fruit detection to cultivar identification in colour images of mango trees. *arXiv e-prints*, Article arXiv:1909.10939, arXiv:1909.10939. https://doi.org/10.48550/arXiv.1909.10939

Botterill, T., Paulin, S., Green, R., Williams, S., Lin, J., Saxton, V., Mills, S., Chen, X., & Corbett-Davies, S. (2017). A robot system for pruning grape vines. *Journal of Field Robotics*, *34*(6), 1100–1122. https://doi.org/10.1002/rob.21680

Camaro Nogues, F., Huie, A., & Dasgupta, S. (2018). Object Detection using Domain Randomization and Generative Adversarial Refinement of Synthetic Images. *arXiv e-prints*, Article arXiv:1805.11778, arXiv:1805.11778. https://doi.org/10.48550/arXiv.1805.11778

Casado García, Á., & Heras Vicente, J. (2018). Guiding the creation of deep learning-based object detectors. *XVIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2018): avances en Inteligencia Artificial. 23-26 de octubre de 2018 Granada, España*, 1207–1212. https://doi.org/10.48550/arXiv.1809.03322

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., … Lin, D. (2019). MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv e-prints*, Article arXiv:1906.07155, arXiv:1906.07155. https://doi.org/10.48550/arXiv.1906.07155

Chen, M., Hu, Q., Yu, Z., THOMAS, H., Feng, A., Hou, Y., McCullough, K., Ren, F., & Soibelman, L. (2022). Stpls3d: A large-scale synthetic and real aerial photogrammetry 3d point cloud dataset. *33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022*. https://bmvc2022.mpi-inf.mpg.de/429/

Chen, Y., & Medioni, G. (1992). Object modelling by registration of multiple range images. *Image and Vision Computing*, *10*(3), 145–155. https://doi.org/10.1016/0262-8856(92)90066-C

Coleman, D., Șucan, I. A., Chitta, S., & Correll, N. (2014). Reducing the barrier to entry of complex robotic software: A moveit! case study. *Journal of Software Engineering for Robotics*, *5*(1), 3–16. https://doi.org/10.6092/JOSER_2014_05_01_p3

Di Cicco, M., Potena, C., Grisetti, G., & Pretto, A. (2017). Automatic model based dataset generation for fast and accurate crop and weeds detection. *2017 IEEE/RSJ Inter-*

national Conference on Intelligent Robots and Systems (IROS), 5188–5195. https://doi.org/10.1109/IROS.2017.8206408

Esser, F., Rosu, R. A., Cornelißen, A., Klingbeil, L., Kuhlmann, H., & Behnke, S. (2023). Field robot for high-throughput and high-resolution 3d plant phenotyping: Towards efficient and sustainable crop production. *IEEE Robotics & Automation Magazine*, *30*(4), 20–29. https://doi.org/10.1109/mra.2023.3321402

Fernandes, M., Scaldaferri, A., Fiameni, G., Teng, T., Gatti, M., Poni, S., Semini, C., Caldwell, D., & Chen, F. (2021). Grapevine winter pruning automation: On potential pruning points detection through 2d plant modeling using grapevine segmentation. *2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, 13–18. https://doi.org/10.1109/CYBER53097.2021.9588303

Gentilhomme, T., Villamizar, M., Corre, J., & Odobez, J.-M. (2023). Towards smart pruning: Vinet, a deep-learning approach for grapevine structure estimation. *Computers and Electronics in Agriculture*, *207*, 107736. https://doi.org/10.1016/j.compag.2023.107736

Gilby, C. (2018). Man vs machine: The challenges of harvesting [Accessed: 2023-12-10]. https://www.thewinesociety.com/discover/explore/the-road-less-travelled/man-vs-machine-the-challenges-of-harvesting

Görner, M., Haschke, R., Ritter, H., & Zhang, J. (2019). Moveit! task constructor for task-level motion planning. *2019 International Conference on Robotics and Automation (ICRA)*, 190–196. https://doi.org/10.1109/ICRA.2019.8793898

Grimm, J., Herzog, K., Rist, F., Kicherer, A., Töpfer, R., & Steinhage, V. (2018). An adaptive approach for automated grapevine phenotyping using vgg-based convolutional neural networks. *CoRR*, *abs/1811.09561*. https://doi.org/10.48550/arXiv.1811.09561

Guadagna, P., Fernandes, M., Chen, F., Santamaria, A., Teng, T., Frioni, T., Caldwell, D. G., Poni, S., Semini, C., & Gatti, M. (2023). Using deep learning for pruning region detection and plant organ segmentation in dormant spur-pruned grapevines. *Precision Agriculture*, *24*(4), 1547–1569. https://doi.org/10.1007/s11119-023-10006-y

Guadagna, P., Frioni, T., Chen, F., Delmonte, A. I., Teng, T., Fernandes, M., Scaldaferri, A., Semini, C., Poni, S., & Gatti, M. (2021). Fine-tuning and testing of a deep learning algorithm for pruning regions detection in spur-pruned grapevines. *Precision agriculture 21*, 147–153. https://doi.org/10.3920/978-90-8686-916-9_16

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2980–2988. https://doi.org/10.1109/ICCV.2017.322

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Hou, J., Dai, A., & Nießner, M. (2019). 3d-sis: 3d semantic instance segmentation of rgb-d scans. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4416–4425. https://doi.org/10.1109/CVPR.2019.00455

Intrieri, C., & Poni, S. (1995). Integrated evolution of trellis training systems and machines to improve grape quality and vintage quality of mechanized italian vineyards. *American Journal of Enology and Viticulture*, *46*(1), 116–127. https://doi.org/10.5344/ajev.1995.46.1.116

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., & Girshick, R. (2023). Segment Anything. *arXiv*

e-prints, Article arXiv:2304.02643, arXiv:2304.02643. https://doi.org/10.48550/arXiv.2304.02643

Kohlbrecher, S., von Stryk, O., Meyer, J., & Klingauf, U. (2011). A flexible and scalable slam system with full 3d motion estimation. *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 155–160. https://doi.org/10.1109/SSRR.2011.6106777

Kuznichov, D., Zvirin, A., Honen, Y., & Kimmel, R. (2019). Data augmentation for leaf segmentation and counting tasks in rosette plants. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2580–2589. https://doi.org/10.1109/CVPRW.2019.00314

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *arXiv e-prints*, Article arXiv:2003.08934, arXiv:2003.08934. https://doi.org/10.48550/arXiv.2003.08934

Milioto, A., Lottes, P., & Stachniss, C. (2018). Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in cnns. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2229–2235. https://doi.org/10.1109/ICRA.2018.8460962

Ochs, A. L. (2021). The grapevine magazine - pruning phases, tools & techniques for vineyards [Accessed: 2023-12-10]. https://thegrapevinemagazine.net/2021/11/pruning-phases-tools-techniques-for-vineyards/

Olenskyj, A. G., Sams, B. S., Fei, Z., Singh, V., Raja, P. V., Bornhorst, G. M., & Earles, J. M. (2022). End-to-end deep learning for directly estimating grape yield from ground-based imagery. *Computers and Electronics in Agriculture*, *198*, 107081. https://doi.org/10.1016/j.compag.2022.107081

Palliotti, A., Gatti, M., & Poni, S. (2011). Early leaf removal to improve vineyard efficiency: Gas exchange, source-to-sink balance, and reserve storage responses. *American Journal of Enology and Viticulture*, *62*(2), 219–228. https://doi.org/10.5344/ajev.2011.10094

Poni, S., Gatti, M., Palliotti, A., Dai, Z., Duchêne, E., Truong, T.-T., Ferrara, G., Matarrese, A. M. S., Gallotta, A., Bellincontro, A., Mencarelli, F., & Tombesi, S. (2018). Grapevine quality: A multiple choice issue. *Scientia Horticulturae*, *234*, 445–462. https://doi.org/https://doi.org/10.1016/j.scienta.2017.12.035

Poni, S., Tombesi, S., Palliotti, A., Ughini, V., & Gatti, M. (2016). Mechanical winter pruning of grapevine: Physiological bases and applications. *Scientia Horticulturae*, *204*, 88–98. https://doi.org/10.1016/j.scienta.2016.03.046

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv e-prints*, Article arXiv:1804.02767, arXiv:1804.02767. https://doi.org/10.48550/arXiv.1804.02767

Santos, T. T., de Souza, L. L., dos Santos, A. A., & Avila, S. (2020). Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. *Computers and Electronics in Agriculture*, *170*, 105247. https://doi.org/10.1016/j.compag.2020.105247

Schneider, E., Jayanth, S., Silwal, A., & Kantor, G. (2023). 3d skeletonization of complex grapevines for robotic pruning. *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3278–3283. https://doi.org/10.1109/IROS55552.2023.10341828

Semini, C., Barasuol, V., Focchi, M., Boelens, C., Emara, M., Casella, S., Villarreal, O., Orsolino, R., Fink, G., Fahmi, S., Medrano-Cerda, G. A., Caldwell, D., Sangiah, D.,

Lesniewski, J., Fulton, K., Donadon, M., & Baker, M. (2019). Brief introduction to the quadruped robot hyqreal. *2019 I-RIM Conference*, 86–87. https://doi.org/10.5281/zenodo.4782613

Silwal, A., Yandún, F., Nellithimaru, A. K., Bates, T., & Kantor, G. (2022). Bumblebee: A path towards fully autonomous robotic vine pruning. *Field Robotics*, *2*(1), 1661–1696. https://doi.org/10.55417/FR.2022051

VitiBot. (2022). Bakus [Accessed: 2023-12-15]. https://vitibot.fr/produits-et-services-popup/vineyard-robot-bakus-l/?lang=en

Vu, T., Kim, K., Luu, T. M., Nguyen, T., Kim, J., & Yoo, C. D. (2022). Scalable SoftGroup for 3D Instance Segmentation on Point Clouds. *arXiv e-prints*, Article arXiv:2209.08263, arXiv:2209.08263. https://doi.org/10.48550/arXiv.2209.08263

Wang, Y., Xu, Z., Shen, H., Cheng, B., & Yang, L. (2020). Centermask: Single shot instance segmentation with point representation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9310–9318. https://doi.org/10.1109/CVPR42600.2020.00933

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., & Girshick, R. (2021). Detectron 2 model zoo [Accessed: 2023-12-11]. https://github.com/facebookresearch/detectron2/blob/a0e22dbfa791e6235e4f196d5ce25e754d02be31/MODEL_ZOO.md

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., & Girshick, R. (2019). Detectron2. https://github.com/facebookresearch/detectron2

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5987–5995. https://doi.org/10.1109/CVPR.2017.634

Yang, Y., & Soatto, S. (2020). Fda: Fourier domain adaptation for semantic segmentation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4084–4094. https://doi.org/10.1109/CVPR42600.2020.00414

Yanmar. (2021). Yanmar yv01 [Accessed: 2023-12-13]. https://www.yanmar.com/eu/campaign/2021/10/vineyard/

You, A., Hemming, J., Grimm, C., & Davidson, J. R. (2023). A real-time, hardware agnostic framework for close-up branch reconstruction using RGB data. https://doi.org/10.48550/arXiv.2309.11580

Zeng, L., & Sabah, M. (2020). EOLO: Embedded Object Segmentation only Look Once. *arXiv e-prints*, Article arXiv:2004.00123, arXiv:2004.00123. https://doi.org/10.48550/arXiv.2004.00123