Politecnico di Torino

Politecnico di Torino

Ingegneria Informatica (Computer Engineering)

A.a. 2024/2025

Graduation Session October 2025

Panoptic Segmentation for Fruit Harvesting

Supervisors:

Marcello Chiaberge

Alessandro Navone

Candidate:

Vincenzo Avantaggiato

Abstract

Agriculture is one of the oldest human activities, dating back over 11,000 years, and has always evolved alongside technological progress. Innovations in tools and machinery have historically aimed to reduce the physical and mental fatigue of farmers. Nevertheless, many operations, such as pruning and fruit harvesting, are still predominantly carried out manually, either because they require high precision or because existing solutions are far from being effective. Current research increasingly focuses on robotic and automated solutions, whose effectiveness, however, depends strongly on the ability of machines to perceive and interpret complex natural environments. In the case of fruit harvesting, the focus of this thesis, state-of-the-art perception technologies typically focus on detecting fruits while neglecting the surrounding structures that are equally crucial for robotic navigation and manipulation.

This work addresses this gap by investigating panoptic segmentation, a computer vision approach that unifies instance and semantic segmentation, to enhance perception in agricultural environments, with particular focus on apple orchards. To overcome the lack of suitable public datasets, two dedicated resources have been created: SPARTA (Synthetic Panoptic Apple oRchard Tree Annotations), a synthetic dataset designed for controlled variability and scalability, and ATHENS (Apple Tree Harvesting Environment with Natural Scenes), a real-world dataset capturing the complexity and heterogeneity of natural orchard conditions. The latter was acquired during two outdoor campaigns in Saluzzo (Cuneo, Italy) using multiple stero cameras.

On the methodological side, the thesis studies, compares, and extends state-of-the-art panoptic segmentation architectures, including Panoptic-DeepLab and ESANet, introducing modifications tailored to the task. The goal is to assess the effectiveness of synthetic datasets supporting in real-world scenarios, analyzing how resolution, dataset size, and variability affect final segmentation performance.

Experimental results show that a more variegated synthetic dataset tends to reduce performance on the synthetic benchmark itself but improves generalization to unseen real data. Moreover, incorporating depth information brings a slight gain in segmentation performance, although it comes at the cost of roughly doubling computational requirements during training and inference. Overall, this work provides contributions ranging from the creation of new datasets to methodological insights on segmentation architectures and synthetic-to-real transfer, thereby supporting future advances in robotic perception for agriculture.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Marcello Chiaberge and Alessandro Navone, for their constant guidance, patience, and insightful feedback throughout the realization of this work. I would also like to express my sincere appreciation to the entire PIC4SeR team for welcoming me into their environment. Their support, both technical and practical, as well as the availability of workspace and tools, were essential to carrying out the experiments and bringing this project to completion. Finally, I wish to thank all those who, in various ways, have contributed to this journey with their advice, discussions, and encouragement.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Agriculture, and in particular fruit production, relies on highly labor-intensive and physically demanding operations such as pruning and harvesting, which are still predominantly performed manually. Despite these challenges, fruit production remains a key pillar of the global agricultural economy: in 2022, fruit harvesting in the European Union was valued at €27.3 billion, with Spain, Italy, and Poland as the leading producers, while the United States recorded fruit sales worth $34.2 billion in the same year [1]. However, the sector is increasingly struggling to secure sufficient labor. In 2020, only 6.5% of EU farm managers were under the age of 35 [2], and labor costs have risen significantly due to various economic and demographic factors.

In response to these needs, and driven by technological advancements, the development of automatic and robotic machines has emerged as a promising solution. To facilitate the mobility and effectiveness of such systems, farmers have progressively reorganized canopies into wall or trellis structures. Robotic solutions are inherently multidisciplinary, combining expertise in mechanical design, control and automation. Central to their operation is the ability to perceive and interpret the environment through cameras and sensors. The raw data collected must be processed and analyzed to identify targets and navigate complex environments effectively.

This thesis is positioned within this context, focusing on the application of panoptic segmentation to enhance robotic perception of the external environment, with the aim of supporting more accurate and efficient automation in fruit harvesting. The research was conducted at **PIC4SeR** (PoliTO Interdepartmental Centre for Service Robotics), a research center active in a wide range of fields, including precision agriculture, smart cities, search and rescue, wellbeing-oriented service robotics, cultural heritage as well as underwater and space applications. In addition to research, PIC4SeR actively supports education at PoliTO by organizing seminars, lectures, student team projects and master's theses. Indeed, the present work has

been developed within the framework of the open thesis positions for the 2024/25 academic year.

The objective of this work is to study, compare, and implement different panoptic segmentation approaches for the identification of apple fruits and their surrounding objects, with the broader aim of supporting robotic perception in agricultural environments. Since no public dataset is specifically tailored to this task, a fundamental contribution of the thesis lies in the creation of two dedicated datasets. The first, named **SPARTA** (Synthetic Panoptic Apple oRchard Tree Annotations), is a synthetic dataset designed to provide controlled variability and scalability, while the second, **ATHENS** (Apple Tree Harvesting Environment w/ Natural Scenes), is a real-world dataset that captures the complexity and heterogeneity of natural orchard conditions.

On the methodological side, the work explores and evaluates several state-of-the-art architectures for panoptic segmentation, including **PanopticDeepLab** and **ESANet**, together with a set of modifications and extensions proposed and implemented as part of this thesis. These models are trained and validated on the newly developed datasets, allowing for a systematic comparison of their performance in both synthetic and real scenarios.

This thesis aligns with the United Nations Sustainable Development Goals (SDGs), a framework of 17 goals set in 2015 by all UN members that aim to tackle the most urgent global challenges. In particular, this thesis contributes to **SGD 2: Zero hunger** by providing solutions able to improve agricultural productivity and sustainability. Better robotic perception means increasing harvesting efficiency and reducing fruit damage, therefore reducing food loss. For the same reason, a big role is played in **SGD 12: Responsible consumption and production** by reducing waste along the supply chain. This work also relates to **SDG 8: Decent work and economic growth**, helping in reducing undesirable and injury-prone work.

Figure 1.1: Sustainable Development Goals addressed in this thesis.

The remainder of this thesis is organized as follows. Chapter 2 presents an introduction to computer vision and its evolution from simple algorithms to deep learning, with an in-depth focus on panoptic segmentation. Chapter 3 presents the state of the art solutions in fruit harvesting, including an analysis of pubblicly available apple tree datasets. Chapter 4 introduces the datasets developed in this work, namely SPARTA and ATHENS, describing their design, structure and annotation process. Chapter 5 outlines the methodology, detailing the architectures under study: Panoptic-DeepLab, ESANet and their proposed modifications. Chapter 6 reports the experimental setup and results obtained from training and evaluating the models on the proposed datasets. Finally, Chapter 7 summarizes the main findings and discusses possible directions for future research.

Chapter 2

Computer vision

2.1 A general overview

Computer vision (CV) is the branch of artificial intelligence concerned with processing and interpreting visual inputs such as images and videos. Its primary goal is to extract meaningful information that enables machines to perform specific tasks. The outputs of CV models have a wide range of applications, effectively substituting or augmenting human vision in fields where visual information is essential, such as autonomous driving, robotics or industrial automation. Over the last decades, the field has undergone a remarkable evolution, which will be outlined in the following sections, with particular attention to the most common tasks.

2.1.1 Pre-Deep Learning Era

Despite the recent success of deep learning techniques, computer vision lies its roots in algorithms and techniques that date back to the last century. The father of computer vision is considered to be Larry Roberts, who published in 1963 his Ph.D. thesis «Machine perception of three-dimensional solids» [3] on the topic of computer vision, specifically on the reconstruction of 3D models from 2D images. Another milestone in this field was represented by «Representation and recognition of the spatial organization of three-dimensional shapes» [4] of David Marr, who introduced a bottom-up paradigm for scene understanding that influenced the field for decades. In the following years, CV techniques focused on solving specific tasks such as line detection, performed through the Canny Edge Detector [5] or the Hough Transform [6] algorithms which are still widely used today.

Fast forward, the introduction of machine learning techniques opened the door to more complex models that could learn from data. Machine learning techniques can be grouped into three main categories:

- **Supervised learning**, which involves labeled training data that the model uses to learn the relationship between inputs and outputs. Popular algorithms are Support Vector Machines (SVM), KNN (K-Nearest Neighbors), decision trees, logistic regression and random forests.
- **Unsupervised learning** that, on the other hand, is based on unlabeled training data. Clustering algorithms such as K-Means and DBSCAN fall into this category.
- **Semi-supervised learning** is a combination of the two previous approaches, having a small amount of labeled data and a large amount of unlabeled data.

In image processing, unsupervised ML techniques were used to extract features from images while supervised ML techniques were used for detection and recognition tasks. [7] [8] [9]

2.1.2 Deep Learning Era

The main limitation of traditional machine learning techniques was the lack of ability to process raw data and therefore the need for human intervention to design suitable feature extractors. A clear example is represented by images, that are basically stored as arrays of pixels, with no immediate meaning or direct relationship with the final task. Deep learning models, instead, are composed of multiple layers of neurons that can learn hierarchical representations of the data, automatically extracting features from raw inputs. It is important to note that these layers are not human-engineered, but learned from data.

The key mechanism that enables this automatic learning process is the **back-propagation** algorithm. In practice, the layers of a deep learning model are initialized with random weights and then progressively refined by training on a dataset. The objective of training is to minimize a loss function, which quantifies the discrepancy between the model's predictions and the ground truth labels. To achieve this, the gradient of the loss function with respect to the model's weights is computed and used to adjust the same weights. Backpropagation can therefore be seen as the systematic application of the chain rule of derivatives in reverse order, starting from the output layer and propagating backwards through the network.

The breakthrough in deep learning for computer vision was represented by the introduction of **Convolutional Neural Networks** (CNNs), which, thanks to the nature of the convolution operation, are able to better understand structured data such as images. CNNs are mainly composed of two types of layers: convolutional layers and pooling layers. Convolutional layers apply a set of learnable filters (kernels) to the input, producing feature maps that capture local patterns in the

data. Pooling layers, instead, reduce the spatial dimensions of the feature maps, retaining only the most important information and reducing the computational complexity of the model. Figures 2.1 and 2.2 provide a visual representation of these operations. Despite being used since the early 2000s in different computer vision tasks, CNNs were marginal and not widely adopted until the introduction of **AlexNet** [10] in 2012, when it won the ILSVRC [11] (a yearly competition for object detection and image classification on the ImageNet dataset). The success of AlexNet is to be attributed to a better utilization of GPUs, the introduction of the ReLU activation function and the introduction of the dropout technique (explained in Figure 2.3).

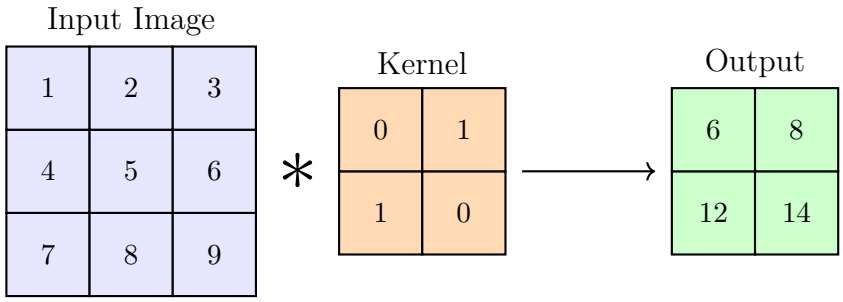


Figure 2.1: Illustration of the convolution operation: a 2×2 kernel slides over a 3×3 input image, producing a 2×2 output. Each output value is the sum of element-wise multiplications between the kernel and the corresponding input region.

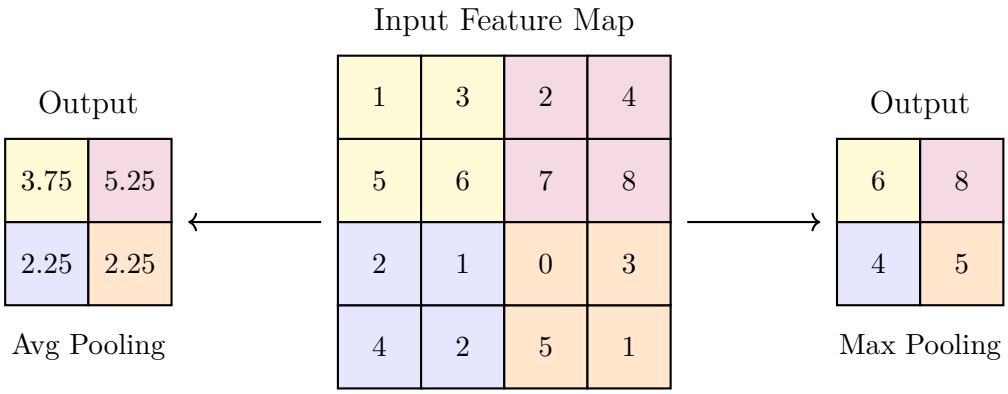
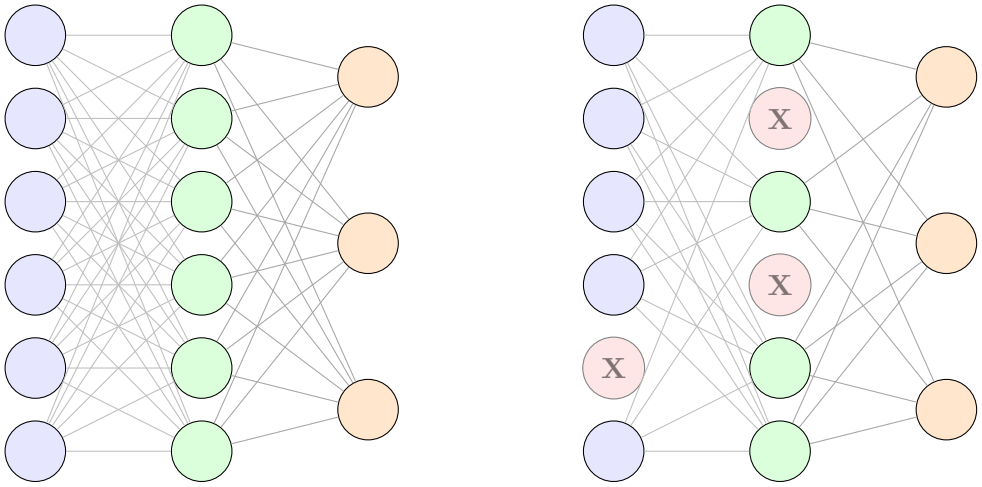


Figure 2.2: Illustration of the pooling operation: a 2×2 pooling window slides over a 4×4 input feature map, producing a 2×2 output feature map. Each output value is the maximum (right) or average (left) of the values in the corresponding input region.



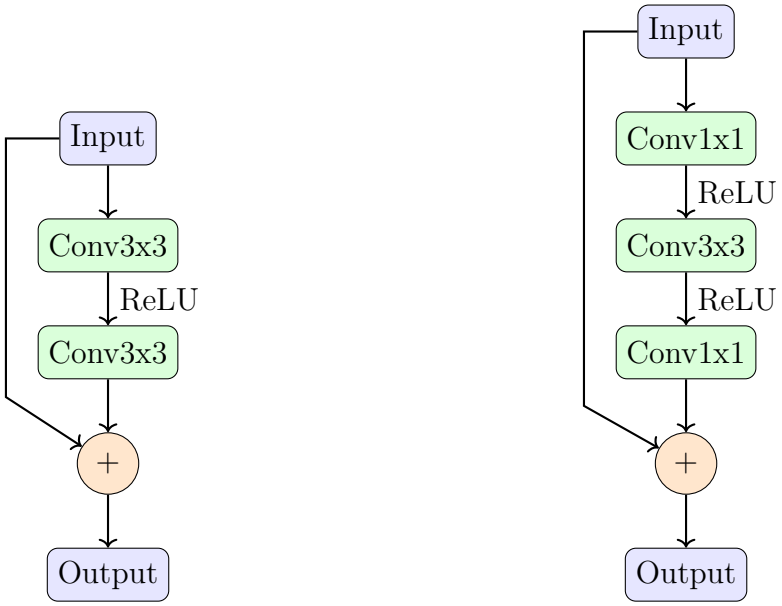
(a) Normal neural network. **(b)** Neural network with dropout.

Figure 2.3: Comparison between a standard neural network (a) and a network with dropout (b). During training, dropout randomly deactivates a subset of neurons at each iteration, reducing reliance on specific neurons and helping the model generalize better. At inference time, all neurons are active.

So, after 2012, CNNs became the standard for all computer vision tasks, inspiring many other architectures that gradually introduced new techniques and concepts.

One of the most influential architectures in deep learning is **ResNet** [12], which introduced the concept of **residual connections**. As He et al. demonstrated, training very deep networks is challenging because gradients tend to vanish or explode, preventing effective learning. Residual connections address this limitation by adding the input of a layer directly to its output, thereby facilitating gradient flow and enabling the network to also learn identity mappings. Figure 2.4 illustrates the two main types of residual blocks employed in ResNet. The introduction of this mechanism allowed to train and engineer much deeper architectures.

Last but not least, one of the most recent and impactful trends in computer vision is the introduction of **Vision Transformers** (ViTs) [13], which adapt the transformer architecture, originally developed with great success in Natural Language Processing, to the visual domain. This paradigm shift brings both strengths and limitations: on the one hand, transformers offer greater flexibility and scalability compared to convolutional models; on the other hand, they demand larger amounts of data for effective training and are computationally more expensive than traditional CNNs [14].



(a) Residual block without bottleneck. **(b)** Residual block with bottleneck.

Figure 2.4: Illustration of ResNet residual blocks: (a) basic block (two 3×3 convolutions), (b) bottleneck block (1x1, 3x3, 1x1 convolutions). The skip connection adds the input to the output of the block. Bottleneck blocks are used because they reduce the number of parameters and computational cost while maintaining performance.

2.1.3 Computer Vision Tasks

Computer vision encompasses several tasks, but some of them are more popular than others and are therefore grouped into three main categories, depending on the final goal and desiderata outputs: classification, detection and segmentation. These tasks can be solved using deep learning techniques, which have revolutionized the field in the last decade.

Image Classification, one of the simplest tasks, consists in assigning a category label to an image based on its content. It's important to note that this task provides a single label for the entire image, without any spatial information about the objects present in it. As an example, a model trained to classify images of animals like the one in Figure 2.5 would output a series of probabilities for each class, choosing the class with the highest probability as the final prediction.

Like any other classification task, Image Classification uses **accuracy** (Equation 2.1) as the main evaluation metric, alongside the strictly related metrics such as **precision** (Equation 2.3), **recall** (Equation 2.2) and **F1 score** (Equation 2.4).

Image credits: Pixabay

Cat: 90%
Dog: 8%
Rabbit: 2%

Figure 2.5: Image classification example on a picture of a cat.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total number of predictions}} \tag{2.1}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.2}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.3}$$

$$\text{F1 score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

Object Detection is instead a more complex task that provides a label for each identified object in the image coupled with a bounding box that delimits the area occupied by the object and a confidence score that indicates the model's certainty about the prediction, as shown in Figure 2.6.

One of the most common evaluation metrics for object detection is **Intersection over Union** (IoU, also known as Jaccard Index), which measures the overlap between the predicted bounding box and the ground truth bounding box. Equation 2.5 and Figure 2.7 illustrate the concept of IoU.

$$\text{Intersection over Union (IoU)} = \frac{\text{Area of Intersection}}{\text{Area of Union}} = \frac{|A \cap B|}{|A \cup B|} \tag{2.5}$$

Precision, **recall** and **F1 score** can also be used to evaluate object detection models and follow the previous definitions (Equations 2.3, 2.2 and 2.4). However, in this context, a way to identify true positives, false positives and false negatives

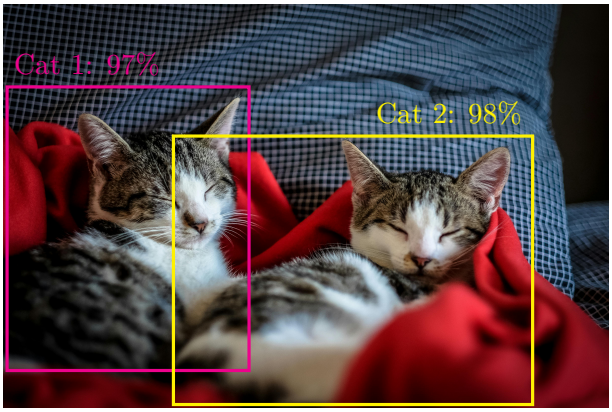


Image credits: Francesco Ungaro

Figure 2.6: Object detection example on an image with two cats.

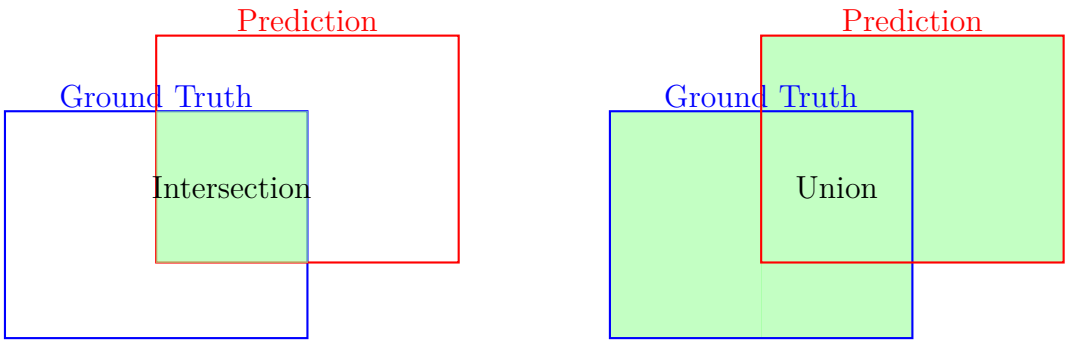


Figure 2.7: Illustration of Intersection and Union between two bounding boxes.

must be defined. That's where IoU comes into play: two bounding boxes are considered matching if their IoU is above a certain threshold (e.g. 0.5).

Two other interesting metrics are **Average Precision** (AP) and **Mean Average Precision** (mAP). Average Precision evaluates the precision-recall trade-off by computing these two metrics at different confidence thresholds, plotting the precision-recall curve and calculating the area under the curve (AUC) (see Figure 2.8). The Mean Average Precision, instead, extends the concept of Average Precision by averaging AP across all classes. In some benchmark datasets, such as COCO, the mAP is computed at different IoU thresholds (e.g. 0.25, 0.5, 0.75, and then averaged across all thresholds) to provide a more comprehensive evaluation of the model's performance. [15]

Finally, **segmentation** is a family of pixel-level tasks that provide a prediction for each pixel in the image. The main differences between the segmentation tasks

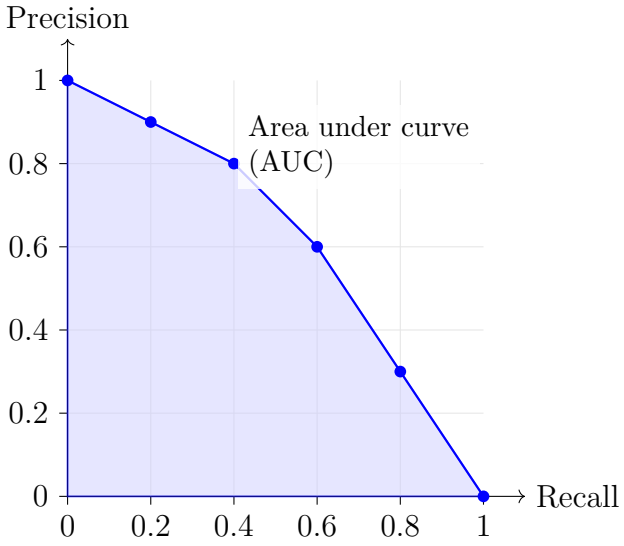


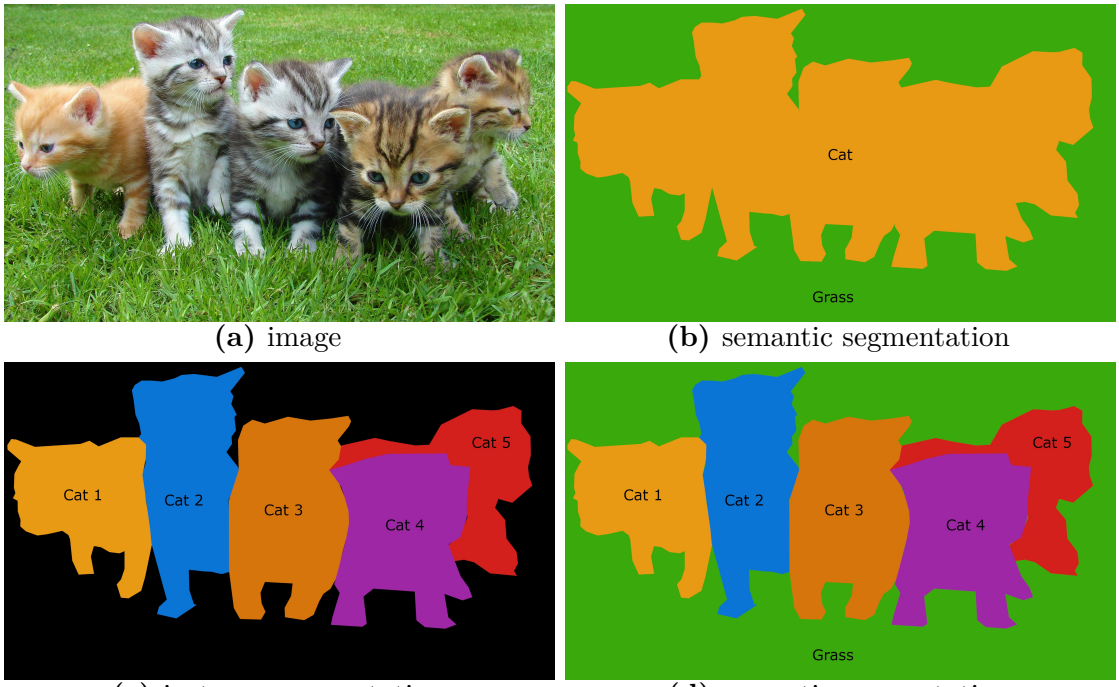
Figure 2.8: Example of a precision-recall curve and the area under the curve (AUC) representing the Average Precision (AP).

lie in how they treat stuff and things (better defined in Section 2.2), whether they take into account the instance of the objects and how they encode the output.

- **Semantic segmentation** assigns, for each pixel in the image, just a single label that indicates the class of the pixel, without distinguishing between different instances of the same class.
- **Instance segmentation** predicts the boundaries and shape of each object (so each thing instance) in the image, assigning a unique label to each instance. It does not provide any information about stuff regions.
- **Panoptic segmentation** combines the two previous tasks, providing a label for each pixel in the image, distinguishing between stuff and things, and assigning a unique label to each instance of the things. This type of segmentation is the core of this thesis and will be discussed in detail in the following sections.

To better understand the differences between these tasks, Figure 2.9 shows an example of each type of segmentation applied to the same image, highlighting the different outputs.

In segmentation tasks, several metrics [16] can be used to evaluate the performance of the model. These metrics, just like in object detection, must encompass both the concepts of classification and localization. Unlike object detection, however, segmentation needs a pixel-level definition of these metrics. For semantic segmentation, the most common metrics are:



(a) image **(b)** semantic segmentation

(c) instance segmentation **(d)** panoptic segmentation

Image credits: Pixabay

Figure 2.9: Comparison of different segmentation tasks on the same image.

- **Pixel Accuracy** (Equation 2.6): the ratio of correctly classified pixels to the total number of pixels in the image.

$$\text{PA} = \frac{\sum_{j=1}^{k} n_{jj}}{\sum_{j=1}^{k} t_j} \tag{2.6}$$

  where n_{jj} is the number of pixels correctly classified as class j (TP) and t_j is the total number of pixels in class j.

- **Mean Pixel Accuracy** (Equation 2.7): the average pixel accuracy across all classes.

$$\text{mPA} = \frac{1}{k} \sum_{j=1}^{k} \frac{n_{jj}}{t_j} \tag{2.7}$$

- **Intersection over Union** (Equation 2.8): the ratio of the intersection of the pixel-wise classification results with the ground truth, to their union.

$$IoU = \frac{\sum_{j=1}^{k} n_{jj}}{\sum_{j=1}^{k} (n_{ij} + n_{ji} + n_{jj})}, \qquad i \neq j \tag{2.8}$$

where n_{ij} is the number of pixels classified as class j but belonging to class i (FP) and n_{ji} is the number of pixels belonging to class j but classified as class i (FN).

- **Mean Intersection over Union** (Equation 2.9): the average IoU across all classes.

$$mIoU = \frac{1}{k}\sum_{j=1}^{k}\frac{n_{jj}}{n_{ij}+n_{ji}+n_{jj}}, \qquad i \neq j \tag{2.9}$$

For instance segmentation, instead, the metrics are based on the Precision-Recall curve. At pixel level, **Precision** (Equation 2.10) and **Recall** (Equation 2.11) for a given class j are defined as follows:

$$\text{Precision}_j = \frac{n_{jj}}{n_{jj}+n_{ij}}, \qquad i \neq j \tag{2.10}$$

$$\text{Recall}_j = \frac{n_{jj}}{n_{jj}+n_{ji}}, \qquad i \neq j \tag{2.11}$$

while **F1 score** follows the same definition as in object detection (Equation 2.4). The main standard metric for instance segmentation is **Average Precision** (AP), which is computed by averaging the precision at different recall levels, as in object detection. The **Mean Average Precision** (mAP) is then computed by averaging the AP across all classes.

2.2 Panoptic segmentation

Given its central role in this thesis, it is worth dedicating a section to a more detailed discussion of the **panoptic segmentation** task. The section begins by clarifying the distinction between *things* and *stuff*, which forms the basis of the task formulation. It then presents the formal definition of panoptic segmentation, introduces the most commonly used evaluation metrics, reviews benchmark datasets, and concludes with an overview of the main architectural approaches proposed in the literature.

The first step to understand panoptic segmentation is to introduce the concepts of things and stuff. **Stuff** refers to amorphous regions without clear boundaries, such as sky, grass, or water, while **things** denote objects that can be easily separated from the background, such as cars, people, or animals (see Figure 2.10).

2.2.1 Task definition

Panoptic segmentation task was firstly formalized by Kirillov et al. in 2019 [17] as follows:

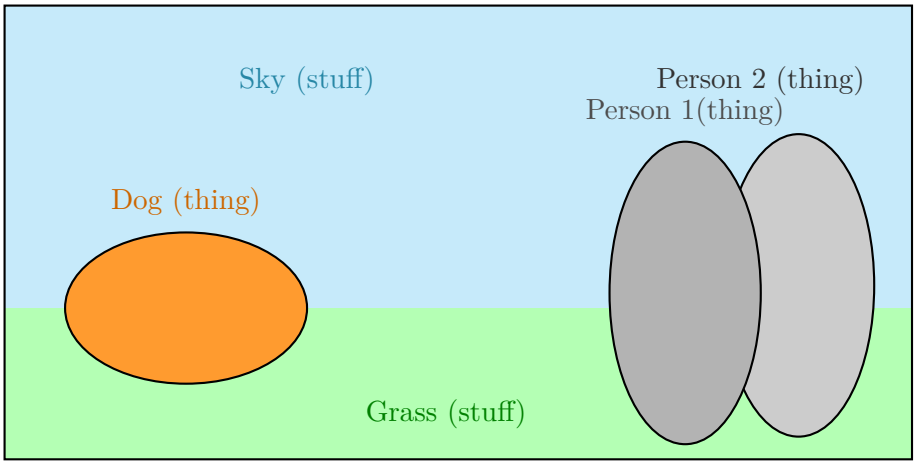


Figure 2.10: Illustration of "stuff" (amorphous regions like sky and grass) and "things" (countable objects like dog and person) in an image.

Definition 1 *Given a predetermined set of L semantic classes encoded by $\mathcal{L} := \{0, ..., L-1\}$ the task requires a panoptic segmentation algorithm to map each pixel i of an image to a pair $(l_i, z_i) \in \mathcal{L} \times \mathbb{N}$ where l_i represents the semantic class of pixel i and z_i represents its instance id. The z_i's group pixels of the same class into distinct segments. Ground truth annotations are encoded identically. Ambiguous or out-of-class pixels can be assigned a special void label.*

The semantic label set $\mathcal{L}$ consist of subsets $\mathcal{L}^{St}$ (stuff labels) and $\mathcal{L}^{Th}$ (thing labels) with the following properties:

- $\mathcal{L} = \mathcal{L}^{St} \cup \mathcal{L}^{Th}$: all labels belong to either one of the two subsets
- $\mathcal{L}^{St} \cap \mathcal{L}^{Th} = \emptyset$: no overlaps are allowed

Like semantic segmentation, but unlike instance segmentation, panoptic segmentation does not require confidence scores associated with each segment.

2.2.2 Evaluation metrics

Along with the definition of the task, Kirillov et al. [17] also defined an official evaluation metric for panoptic segmentation, called **Panoptic Quality** (PQ).

Panoptic Quality was designed in order to be *complete* (i.e. treat stuff and thing segments uniformly), *interpretable* and *simple* (so that it can be easily computed and reimplemented).

$$\mathrm{PQ} = \frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p, g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \tag{2.12}$$

Let's analyze the components of the PQ formula in Equation 2.12: it simply computes the average IoU between the predicted segments p and the ground truth segments g that are considered matching (i.e. true positives, TP) and divides it by a normalization factor that accounts for false positives (FP) and false negatives (FN) in order to penalize segments that are not correctly predicted or missed. It is important to note that Panoptic Quality is calculated for each class separately and then averaged over all classes, making it insensitive to class imbalance.

However, equation 2.12 does not explain how to compute the sets TP, FP and FN. By leveraging Theorem 1, two segments p and g are considered matching if their intersection over union (IoU) is strictly greater than 0.5, allowing to define the three sets.

Theorem 1 *Given a predicted and ground truth panoptic segmentation of an image, each ground truth segment can have at most one corresponding predicted segment with IoU strictly greater than 0.5 and vice verse.*

Interestingly, by multiplying and dividing the numerator and denominator of Equation 2.12 by $|TP|$, the formula can be rewritten as follows:

$$\mathrm{PQ} = \underbrace{\frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition quality (RQ)}} \tag{2.13}$$

where the first term is called **segmentation quality** (SQ) and the second term is called **recognition quality** (RQ) and corresponds to the well-known *F1 score*. This decomposition is useful to better interpret the results of a panoptic segmentation model, but it should not be considered as a combination of semantic and instance segmentation metrics, as it is not possible to compute the two terms separately. In fact, semantic metrics (e.g. pixel accuracy, mean IoU) are not suitable for panoptic segmentation because they do not take into account the instance id, while instance metrics (e.g. AP, mAP) require the model to output confidence scores for each segment, which is not the case for panoptic segmentation.

2.2.3 Datasets

There are several datasets designed for panoptic segmentation, with a few becoming particularly popular and widely adopted as benchmarks. Since panoptic annotations combine semantic and instance information, these datasets can also be employed for the related tasks of semantic segmentation and instance segmentation, making them highly versatile resources for research.

COCO Panoptic [18] contains 80 thing categories and 91 stuff categories. COCO, short for *Common Objects in Context*, was specifically designed to capture everyday scenes, depicting common objects situated in realistic environments rather than isolated. Its diversity and scale have made it one of the most influential datasets across a wide range of computer vision tasks, including detection, segmentation and now panoptic segmentation.

Image credits: https://cocodataset.org/

Figure 2.11: Example of COCO Panoptic annotations.

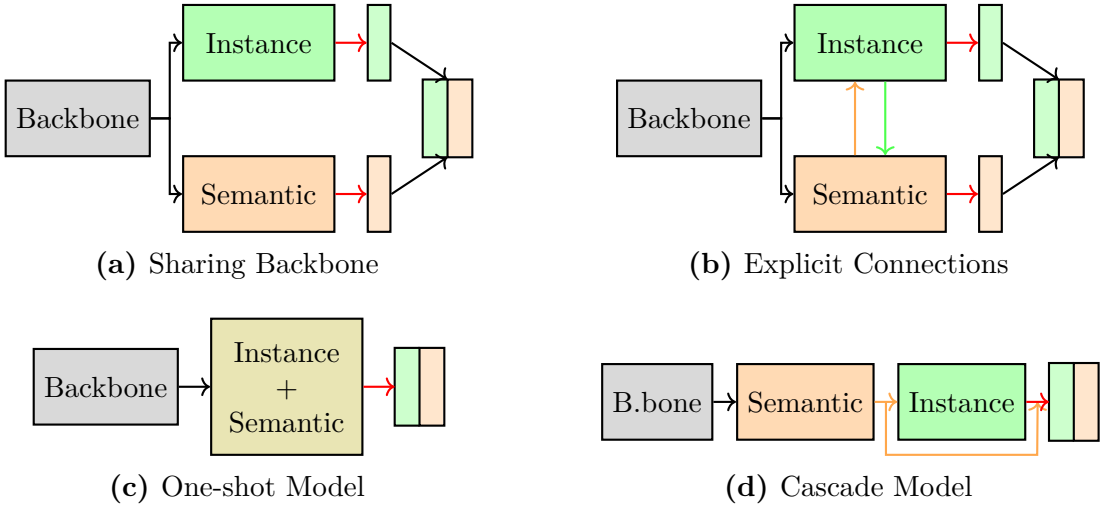
Cityscapes Panoptic Parts [19] focuses on high-resolution urban street scenes, collected from 50 German cities under varying weather and lighting conditions. For the panoptic task, it provides annotations for 8 thing categories and 11 stuff categories. Its emphasis on road scenes makes it especially valuable for research in autonomous driving, where accurate pixel-level understanding of both dynamic objects (cars, pedestrians) and static elements (road, buildings) is essential.

Image credits: https://www.cityscapes-dataset.com/

Figure 2.12: Example of Cityscapes Panoptic annotations.

Mapillary Vistas [20] is a large-scale street-level dataset with global coverage. It provides annotations for 70 thing categories and 46 stuff categories, capturing a wide range of environments, from dense urban areas to rural roads, across diverse continents, climates, and weather conditions. This level of variability makes it particularly challenging and useful for evaluating the generalization capabilities of panoptic segmentation models.

2.2.4 Common architectures



(a) Sharing Backbone

(b) Explicit Connections

(c) One-shot Model

(d) Cascade Model

Figure 2.13: Different model designs for combining semantic and instance segmentation.

Panoptic segmentation can be seen as a combination of semantic and instance segmentation, and this is reflected in the design of the models that can be used to solve the task. Figure 2.13 shows some of the most common designs for panoptic segmentation models. Some of them start with a shared backbone, which extracts features from the input image, that are then fed to two separate branches, one for instance segmentation and one for semantic segmentation. These two branches can be either completely independent (2.13a), or they can share explicit connections used to exchange information between them (2.13b). There are also models that combine the two branches into a single one, which directly predict the final panoptic segmentation (2.13c), or models that use a cascade approach, where the instance segmentation branch is built on top of the semantic segmentation one (2.13d).

RGB images represent the primary data source in which most of the panoptic segmentation algorithms have been applied, but not the only one. Medical images, such as X-rays, and LiDAR data have also been used to tackle the task. [21]

Chapter 3

State of the art

This chapter reviews the state of the art in the application of computer vision techniques to the field of fruit harvesting, with a particular focus on apple production. The discussion begins with an overview of available datasets, followed by an analysis of existing solutions proposed in the literature. Since apples represent the specific case study of this thesis, both the datasets and the methods considered are mostly related to apple harvesting.

3.1 Apple tree datasets

Several datasets containing images of generic fruit trees and more specifically apple trees are publicly available. However, none of them are perfectly tailored for the task of this thesis.

PFuji-Size dataset [22] is composed of 3D point clouds of Fuji apple trees, the raw data to generate them, segmented point clouds for all apples, their centroids and their ground truth diameters. Data was generated from 6 trees containing a total of 615 apples, captured at separate temporal moments, resulting in different maturity stages. In addition, 25 apples were captured under laboratory conditions. The primary goal of this dataset was helping research in apple size estimation.

PApple-RGB-D-Size dataset [23] is generated by extracting a subset of RGB and depth images from [22] and annotating them with instance segmentation masks, apple diameter ground truth and 2D projection of each 3D spherical mask. The dataset constains a total of 3925 images.

Fuji-SfM dataset [24] contains a set of 288 RGB images and their corresponding apple segmentation masks, as well as a set of 582 images defining a motion sequence

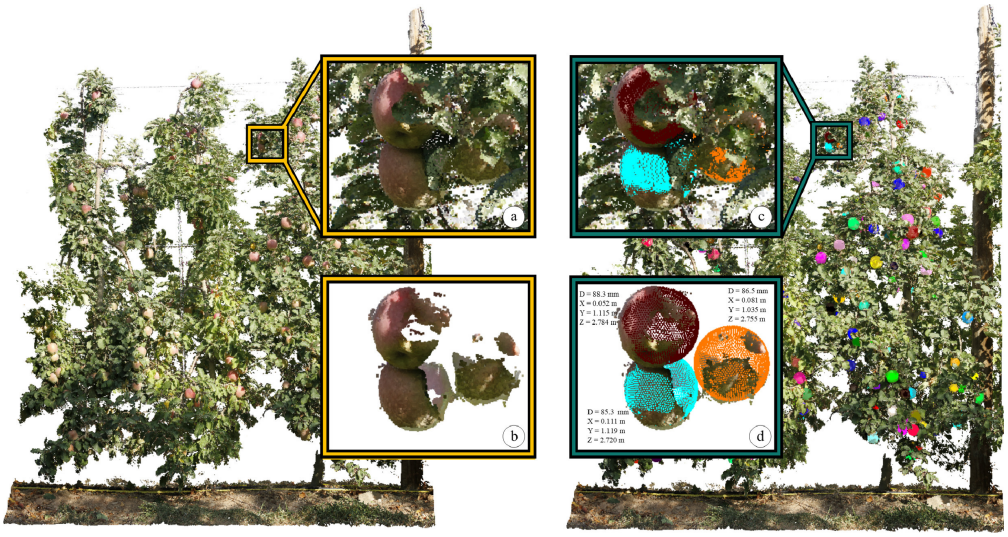


Figure 3.1: Point clouds and annotations from the PFuji-Size dataset

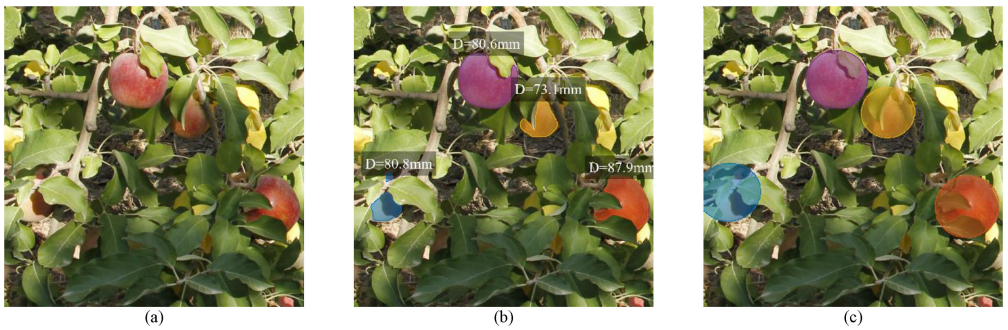


Figure 3.2: Sample from PApple-RGB-D-Size dataset with (a) original image, (b) instance segmentation annotation and (c) projected spherical mask.

fo SfM (structure-from-motion) and the 3D point cloud of the scanned scene. All images are taken from a orchard of 11 Fuji apple trees.

KFuji RGB-DS database [25] is composed of 967 images of Fuji apples in 3 modalities: color (RGB), depth (D) and range corrected IR intensity (S). 12839 apple bounding boxes are annotated.

MinneApple dataset [26] is a dataset designed for fruit detection, segmentation and counting in orchard environments composed of 1000 images and over 410000 annotated objects.

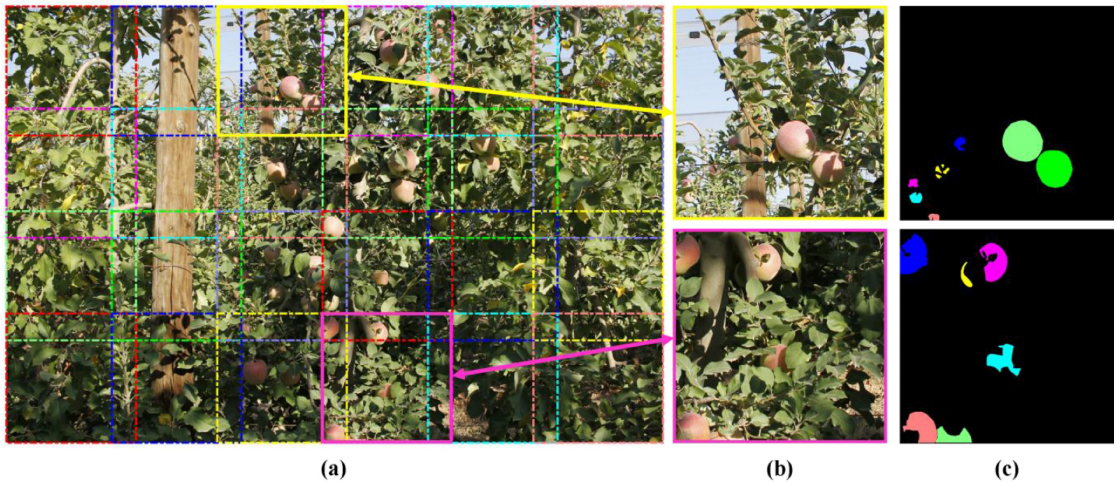


Figure 3.3: Illustration from Fuji-SfM dataset. (a) Image cropping borders, (b) Example of two sub-images, (c) Ground truth segmentation masks

3.2 Existing solutions

Among the numerous solutions proposed for the detection and localization of fruits on trees, several stand out for their originality, performance, or relevance to real-world harvesting scenarios. This subsection reviews the most representative approaches, highlighting the strategies they adopt to handle common challenges such as occlusions, varying illumination and complex orchard environments.

«Panoptic mapping with fruit completion and pose estimation for horticultural robots» [27] proposes a multi-resolution panoptic mapping system in order to estimate 3D shapes of fruit and and their pose. Figure 3.6 shows the whole process followed in this study. A stream of RGB-D images is given as input to a Mask R-CNN [28] network to perform instance segmentation, resulting in per-fruit instance masks that are then used to assign temporal consistent submap IDs to panoptic entities. Once a fruit is not observed for G frames consecutively, it is marked as frozen and used for joint shape completion and pose estimation using a pretrained DeepSDF [29] model and an occlusion-aware differentiable rendering technique. Experiments are carried out on sweet peppers and strawberry crops.

«Instance Segmentation and Localization of Strawberries in Farm Conditions for Automatic Fruit Harvesting» [30] implements a standard instance segmentation to localize strawberries and identify their ripeness level, with a particular attention to occluded fruits. These corner-case fruits have a smaller bounding box, that can can cause problems to the gripper that has to pick them up. The

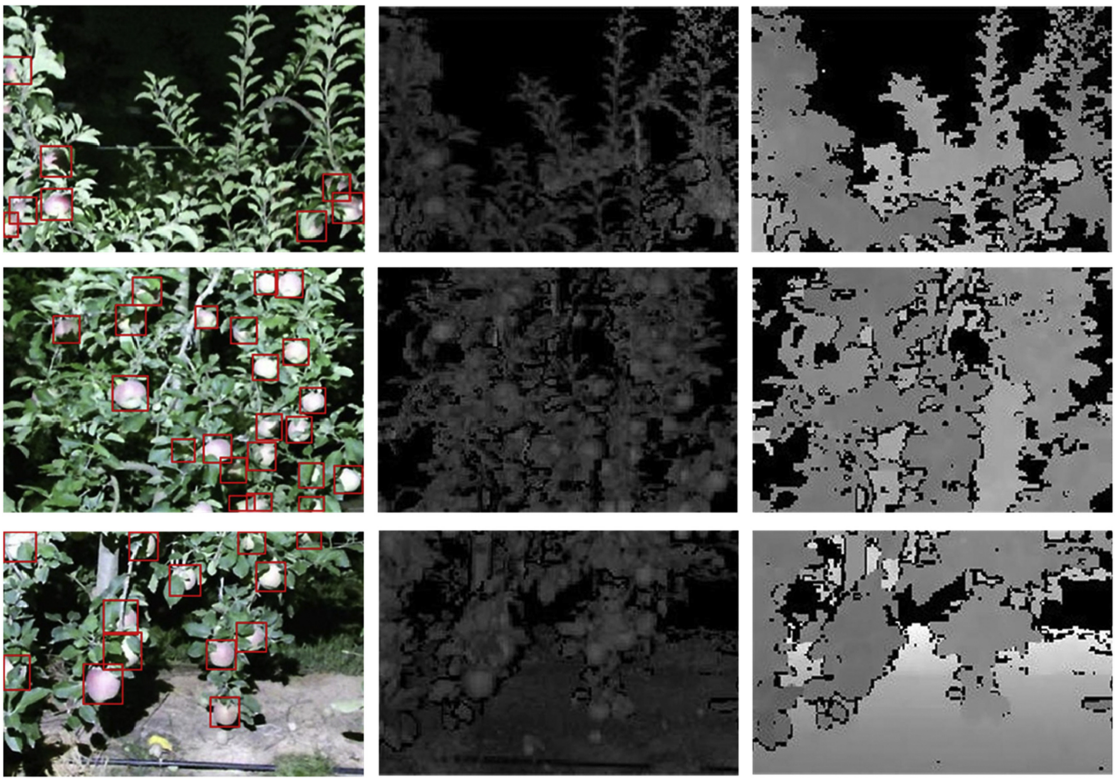


Figure 3.4: Sample of 3 multi-modal images from KFuji RGB-DS database. Each column corresponds respectively to RGB, S and D. Ground truth bounding box are also shown in the first column.

WHR (width-height ratio) is used to detect occlusion, followed by a box-refinement algorithm that recovers real bounding boxes by dividing the instance mask into two parts along the x direction and identifying the non-occluded one (usually convex).

«3D Hierarchical Panoptic Segmentation in Real Orchard Environments Across Different Sensors» [31] simultaneously provide semantic segmentation, instance segmentation of trunks and fruits, and instance segmentation of trees. The architecture (Figure 3.8) is based on a MinkUNet [32] that takes a coloured point cloud as input in which the decoder is replicated three times, one for each task. The outputs of instance segmentation decoders are offset vectors, that need to be clustered with HDBSCAN [33] to obtain the final result.

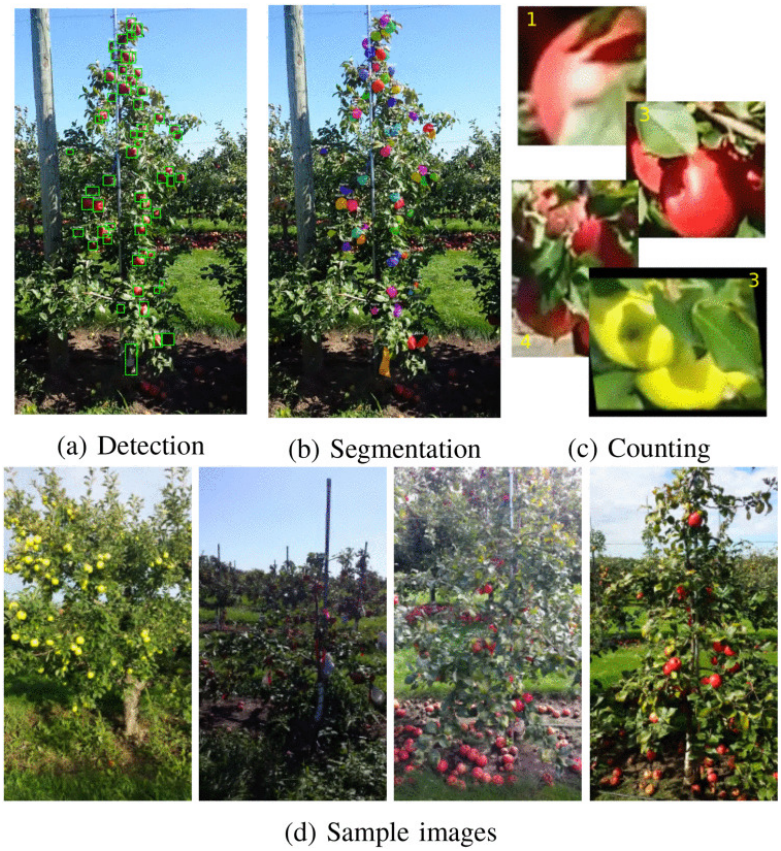


Figure 3.5: Samples from MinneApple dataset

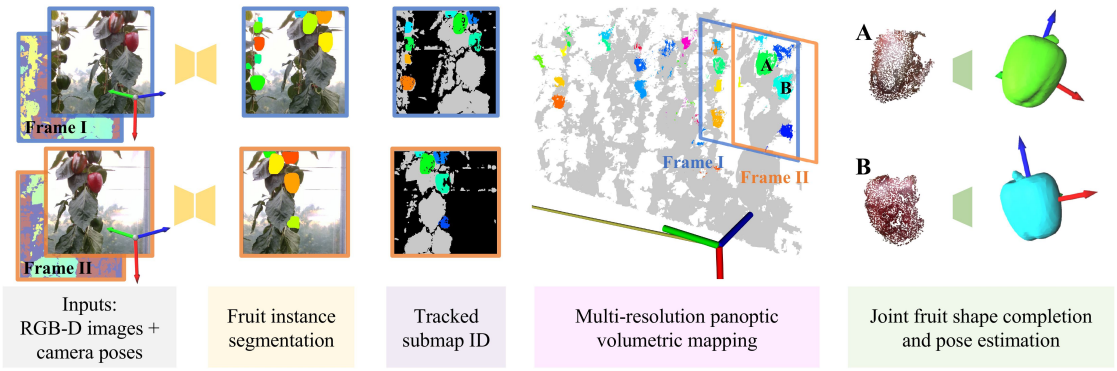


Figure 3.6: Overview pipeline of the mapping system proposed by Pan et al.

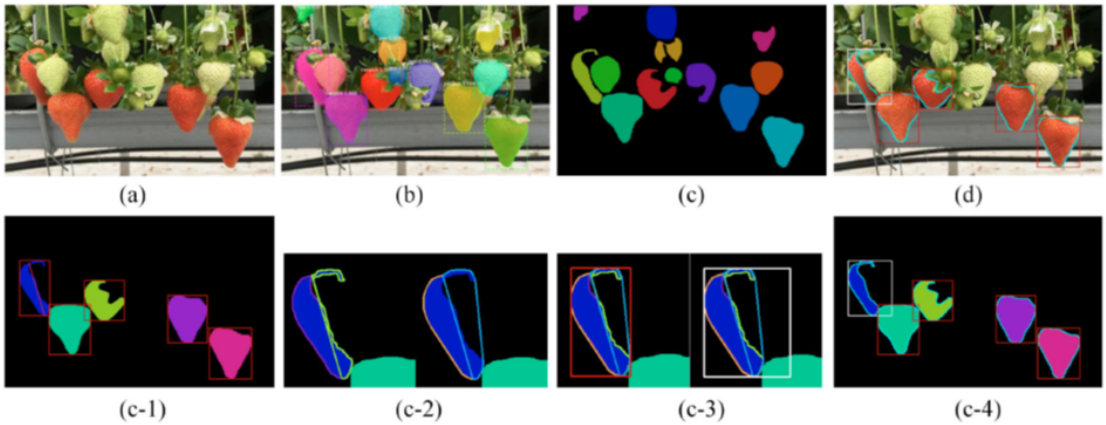


Figure 3.7: Refinement method workflow of Ge et al.: (a) to (d) are the input images, visualized detection results, detected masks and visualized refined results, respectively; (c-1)-(c-4) are the original bounding boxes on the masks, visualized two sides splitting, refinement process and refined bounding boxes on the masks, respectively.

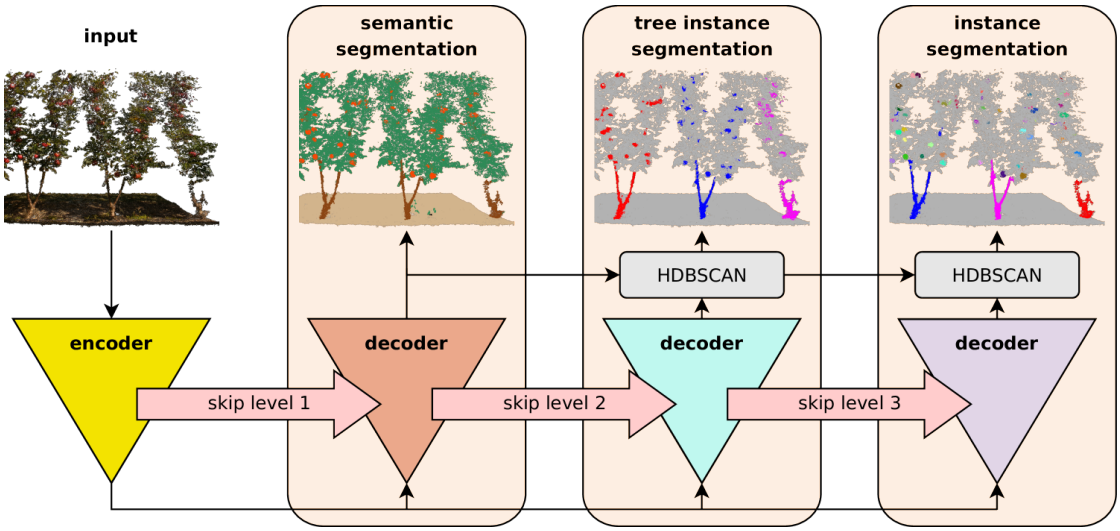


Figure 3.8: Overview of the architecture proposed by Sodano et al.

Chapter 4

Datasets

As described in Section 3.1, the publicly available datasets related to fruit harvesting are not specifically tailored to the panoptic segmentation task addressed in this thesis. For this reason, two new dataset families have been developed, namely **SPARTA** and **ATHENS**, both designed with the explicit goal of supporting panoptic segmentation in apple orchards. SPARTA is a synthetic dataset, generated through controlled simulation to provide large amounts of annotated data under diverse and customizable conditions. This allows the creation of balanced scenarios with precise ground truth annotations, which are particularly valuable for training deep learning models. In contrast, ATHENS consists exclusively of real-world images collected in apple orchards, providing a more realistic but also more challenging benchmark due to natural variability in lighting, occlusions, and background complexity. Together, these two datasets form a complementary pair: one offering scalability and precision through synthetic data, the other providing realism and ecological validity through field imagery.

4.1 Synthetic Datasets for Computer Vision

Datasets in computer vision are typically created by collecting real-world data and enriching it with annotations that serve as ground truth (GT). These annotations are essential, as they provide the supervisory signal that enables models to learn. However, the annotation process is often expensive and time-consuming, particularly in tasks such as segmentation, where pixel-level precision is required. The effort needed to produce high-quality GT can quickly become a bottleneck, limiting the scale and diversity of available datasets. [34]

A common strategy to alleviate this issue is the use of **synthetic datasets**. The idea of employing synthetic data in computer vision is not new: early examples date back to 1988 with the work of Pomerleau [35]. Nonetheless, it is only in the

past decade that the approach has gained widespread attention, largely due to advances in computer graphics, simulation engines, and generative techniques.

The primary advantage of synthetic datasets lies in the ability to generate vast amounts of data along with perfectly accurate annotations at virtually no additional cost. However, other benefits make them even more compelling. Synthetic pipelines can produce balanced and diverse data distributions, including rare or edge-case scenarios that may be underrepresented in real datasets. They are also valuable when data collection is impractical, dangerous, or constrained by ethical and legal considerations such as privacy regulations. Furthermore, the reduced time and cost associated with synthetic data generation make it an efficient alternative to manual annotation.

Evidence of the potential of synthetic data has been reported across different domains. For instance, Patki et al. [36], in their study on the effectiveness of synthetic data, observed that in 70% of cases, results obtained with real data could be replicated using synthetic data. Although this study was conducted outside the computer vision domain, it highlights the strong potential of synthetic approaches, which has only recently begun to be realized in vision-related tasks.

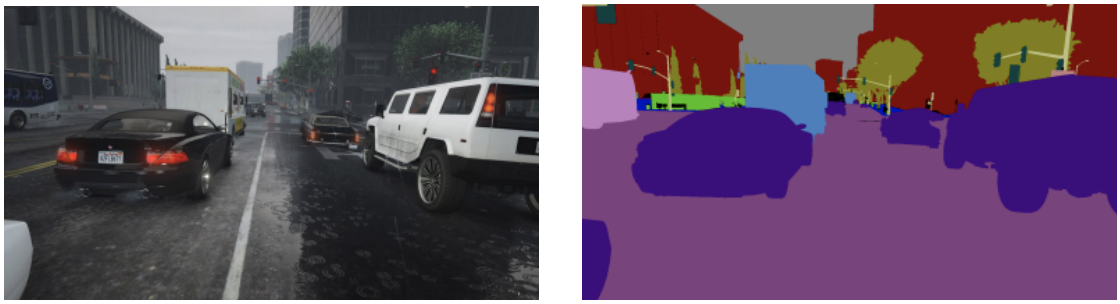
According to Tripathi et al. [37], the effectiveness of synthetic data in computer vision depends on three key properties: it must be *efficient*, in the sense of being cost-effective and scalable; *task-aware*, meaning generated samples should be tailored to the specific requirements of the target task; and *realistic*, ensuring that synthetic images remain visually close to real-world data to minimize the domain gap. These principles guide the design of synthetic datasets and determine their success in practical applications.

Among the most popular synthetic datasets for computer vision, the following are the most popular:

- **GTA5** [38] is a dataset of 24966 images collected from the popular video game *Grand Theft Auto V* (GTA5) annotated for semantic segmentation with 19 classes that are compatible with those of real-world datasets like Cityscapes.

- **SYNTHIA** [39] consists of photo-realistic synthetic images from a virtual city, with 13 classes for semantic segmentation. This virtual city is built using the Unity game engine and allows simulating different weather conditions, times of day, and seasons, but also adding new objects and classes effortlessly.

- **Virtual KITTI** [40], similarly to SYNTHIA, contains 50 high-resolution monocular videos (21260 frames) from five different virtual scenes, in urban context and varying weather conditions.



(a) synthetic image **(b)** ground truth

Figure 4.1: Example of synthetic image and ground truth from the GTA5 dataset.

4.2 SPARTA Dataset

Following the principles outlined in Section 4.1, a new synthetic dataset specifically designed for the task of panoptic segmentation in apple orchards is introduced in this thesis. The dataset is named **SPARTA** (Synthetic Panoptic Apple oRchard Tree Annotations) and represents a large-scale resource tailored to the unique challenges of orchard environments. By leveraging synthetic generation, SPARTA provides pixel-accurate annotations at no additional cost, enabling the training and evaluation of deep learning models under diverse and customizable conditions.

SPARTA images are annotated with 6 panoptic classes, chosen to represent the most relevant elements in an apple orchard scenario:

- **Apple** (thing): the apple fruit, which constitutes the primary object of interest in this dataset.
- **Branches** (stuff): the structural components of the apple tree, excluding leaves and fruit.
- **Leaves** (stuff): the foliage of the apple tree, which often causes significant occlusions and thus represents a major challenge in fruit detection.
- **Supports & Wires** (stuff): the artificial structures used in orchards to sustain the branches and guide tree growth, which may interfere with detection and segmentation.
- **Soil** (stuff): the ground surface on which the tree is planted, included to provide contextual information and complete scene understanding.
- **Background** (stuff): any image region not belonging to the aforementioned classes, capturing surrounding scenery or distant objects, mainly corresponding to sky.

4.2.1 Virtual Apple Orchard

The images in SPARTA are generated from a virtual apple orchard. The orchard was created by PIC4SeR team following a process similar to the one described in their previous work «Enhancing visual autonomous navigation in row-based crops with effective synthetic data generation» [41].

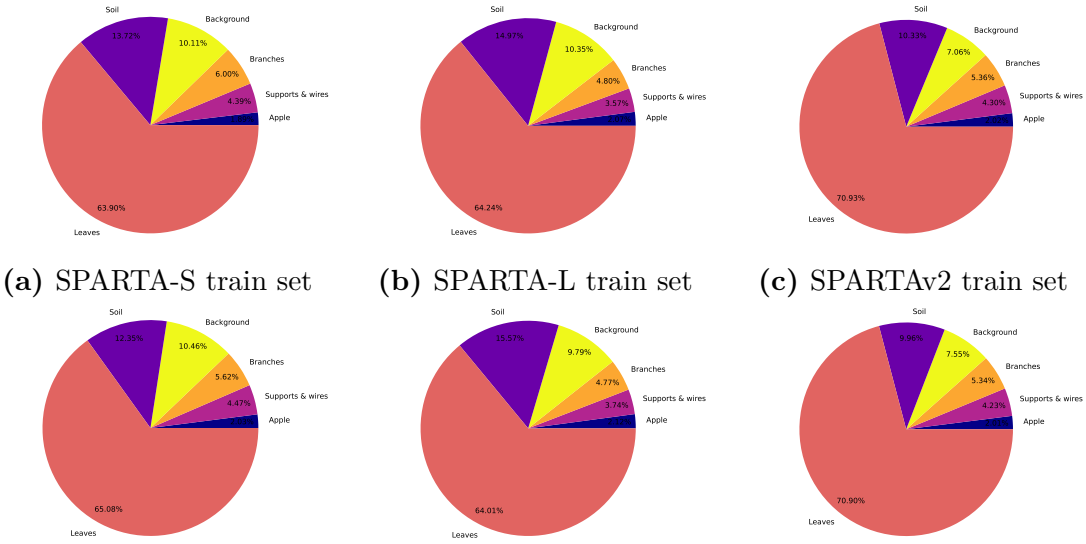
First step consists in modeling 3D representation of the scene elements, including branches, leaves, trunks, apples and supporting structures. This was achieved using *Blender*, a free and open-source 3D computer graphics software. The models were developed using real plant textures and standard dimensions as reference, ensuring that the resulting orchard is both visually realistic and structurally consistent with actual apple trees. Additional components, such as a realistic sky model and an irregular terrain, were incorporated to enhance the authenticity of the virtual environment. Once modeled, the individual objects were assembled into rows of trees, replicating the layout of real orchards.

Certain design choices made during this modeling phase directly influenced the structure of the resulting panoptic dataset. For instance, foliage was modeled at the leaf level but integrated into the orchard as a single element. This imposed the need to treat leaves as a *stuff* class, since no instance-level annotation could be provided. Similarly, supports and wires were fused together in a single 3D model, which consequently resulted in their inclusion as a single dataset class.

Blender's Python scripting interface was then employed to automatically generate the data, enabling the extraction of RGB images, depth maps and segmentation masks with minimal manual intervention. For each rendered image, the pipeline outputs a dedicated folder containing the following four files:

- `camera_pos.txt`, a text file storing the spatial and rotational parameters of the virtual camera (X, Y, Z, roll, pitch, yaw).
- `Depth0001.exr`, a depth map providing per-pixel distance information expressed in meters.
- `Image0001.png`, the rendered RGB image of the scene.
- `SegmentationBW0001.exr`, a segmentation map where each pixel is assigned a class identifier. These identifiers are not directly compatible with the final dataset format but follow a rule-of-thumb convention (Soil: 1, Supports & Wires: 2, Leaves: 3, Branches: 4, Apples: >10).

Finally, a custom post-processing script was developed to convert the raw outputs into the standardized annotation format described in Section 4.4. This script ensures that class identifiers are correctly assigned and that the resulting dataset is ready for use in training and evaluating panoptic segmentation models.



(a) SPARTA-S train set **(b)** SPARTA-L train set **(c)** SPARTAv2 train set

(d) SPARTA-S val set **(e)** SPARTA-L val set **(f)** SPARTAv2 val set

Figure 4.2: Pixel distribution of the classes in the SPARTA dataset (training sets in the first row, validation sets in the second row).

4.2.2 Dataset Splits

The SPARTA dataset has been released in three versions, each tailored to different goals. All releases include depth information directly extracted from the 3D model.

The first release, **SPARTA-S**, is a small collection of 1000 low-resolution images (640×480 px), split into 800 for training and 200 for validation. The second release, **SPARTA-L**, expands the dataset to 4000 high-resolution images (1280×720 px), with 3500 for training and 500 for validation. While these versions were useful for initial experimentation, it soon became evident that the images were often too similar to one another, resulting in limited diversity and reduced potential for generalization when training vision models.

To overcome these issues, a third release was created with a focus on diversity and robustness. **SPARTAv2** contains 10000 images at 640×480 px resolution, each paired with detailed annotations. Unlike the first versions, these images were explicitly generated to cover a broader range of conditions: they are organized into five sets that differ in texture and lighting, thereby introducing significant variability in the visual characteristics of the scenes. For training and evaluation, each of the five sets is split 80%/20% into SPARTAv2-train and SPARTAv2-val, ensuring a balanced distribution of conditions. This design makes SPARTAv2 more suitable for developing and benchmarking models intended to perform well under heterogeneous scenarios.

(a) SPARTA-S **(b)** SPARTA-L

(c) SPARTAv2: Set 1 **(d)** SPARTAv2: Set 2

(e) SPARTAv2: Set 3 **(f)** SPARTAv2: Set 4 **(g)** SPARTAv2: Set 5

Figure 4.3: Samples form SPARTA-S, SPARTA-L, and each of the five sets composing SPARTAv2.

4.3 ATHENS Dataset

ATHENS (Apple Tree Harvesting Environment w/ Natural Scenes) is the real-world counterpart of the synthetic SPARTA dataset. It provides panoptic annotations of apple orchards, following the class definitions introduced in Section 4.2 and the annotation structure described in Section 4.4.

The dataset was created from images acquired in Saluzzo (Cuneo, Italy) during October 2023 in orchards belonging to the *Agrion* foundation. Data was collected from 9 distinct ROS [42] bags, each corresponding to a row of apple trees characterized by differences in cultivar, lighting conditions, background clutter, and plant

structure (see Figure 4.4). Acquisition was performed using two depth cameras: an `Intel® RealSense Depth Camera D435` (bags 1–5, 7) and an `Intel® RealSense Depth Camera D455` (bags 6, 8, 9).

Since each image is manually annotated, labeling every single pixel would have been nearly impossible and prohibitively time-consuming. To address this, a greedy annotation strategy was adopted. The process began by annotating all apples, followed by the most visible branches, supports, and wires. When feasible, large regions of background and soil were also annotated. The most challenging class, leaves, was left for the final stage: only those in the foreground or in close proximity to other annotated classes were labeled. Ensuring that adjacent pixels were annotated was crucial, as this increases the penalty for mispredictions during training. Figure 4.5 illustrates the number of annotated pixels per bag and per class.

Due to some errors in the archiving of the ROS bags, depth was not available for the ATHENS dataset. To make up for this lack, a second, small, version of the dataset was collected in September 2025. This new version, **ATHENSv2**, is created from two bags collected with an `Intel® RealSense Depth Camera D435`.

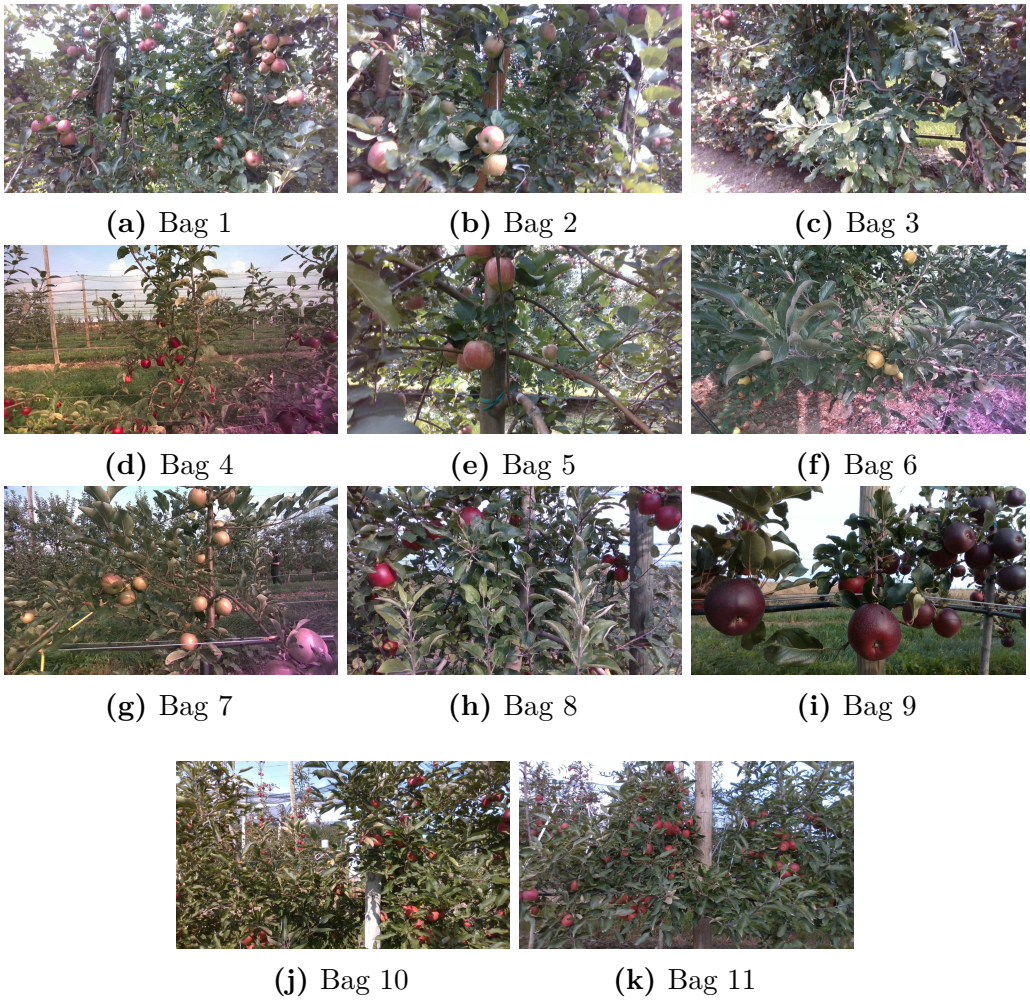
4.3.1 Dataset splits

ATHENS dataset consists of 164 high-resolution images (1280 × 720px), divided into training and validation subsets designed to preserve balance across the bags. Specifically, 80% of the images from each bag were allocated to the training set, while the remaining 20% were reserved for validation. This strategy ensures that the validation set provides a representative sample of each bag, avoiding bias toward a specific vineyard section.

- **ATHENS-train**: 134 images, divided as follows

- **ATHENS-val**: 32 images, divided as follows

This split not only maximizes the amount of data available for training but also guarantees that the validation set contains examples from all vineyard sections. In this way, the model is evaluated on conditions comparable to those seen during training, while still testing its ability to generalize across different bags.



(a) Bag 1 **(b)** Bag 2 **(c)** Bag 3

(d) Bag 4 **(e)** Bag 5 **(f)** Bag 6

(g) Bag 7 **(h)** Bag 8 **(i)** Bag 9

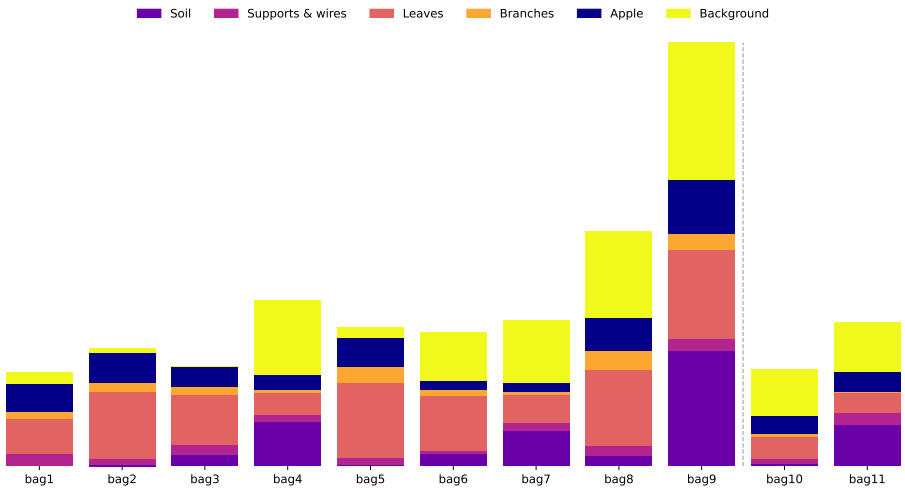
(j) Bag 10 **(k)** Bag 11

Figure 4.4: Samples from ATHENS and ATHENSv2. Each sample belongs to a different bag.

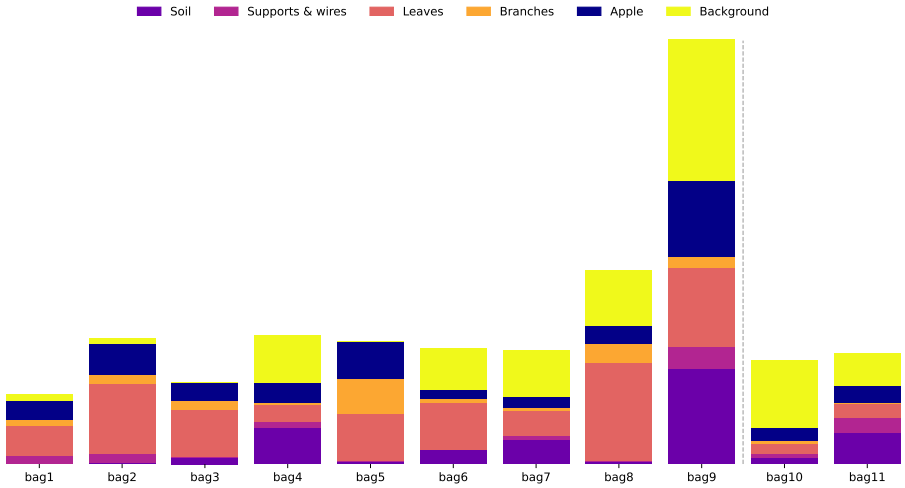
ATHENSv2 is composed of 37 high-resolution images (1280 × 720px), split according to the same policy used for the main dataset. Specifically, **ATHENSv2-train** contains 31 images (15 from bag 10 and 16 from bag 11) and **ATHENSv2-val** contains 6 images (3 from each bag).

4.3.2 Manual annotation with SALT

The images were manually annotated using SALT (Segment Anything Labeling Tool), an open-source annotation tool built on top of the Segment Anything Model (SAM) by Meta AI, available at `https://github.com/anuragxel/salt`.



(a) train set



(b) validation set

Figure 4.5: Number of annotated pixels per bag. Notice that not all pixels are annotated and that bag8 and bag9 contain an higher number of images.

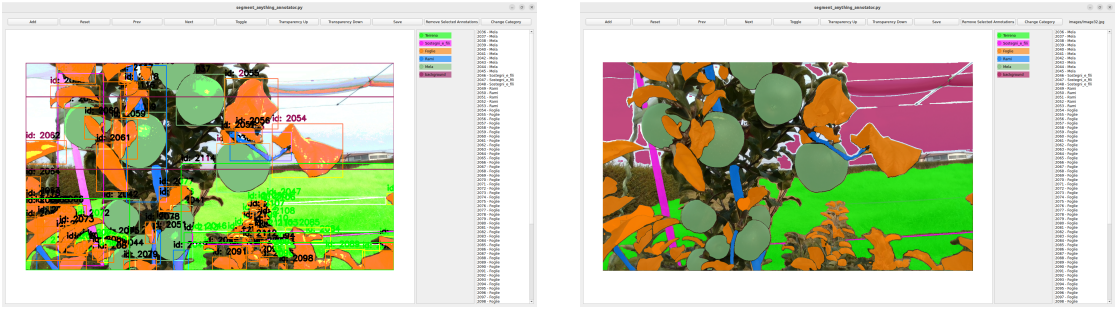
Although providing high-quality segmentation, the default SALT interface slowed down the annotation process for panoptic segmentation. To improve efficiency, several modifications were implemented.

Bounding box toggle Bounding boxes are useful for instance detection, but in dense scenes with overlapping objects they clutter the interface. To address this, a new key binding (B) was introduced to toggle bounding boxes on and off.

Solid coloring of selected segments Instead of blending selected segments with the background (which often made them barely visible), they are now displayed using a solid color to improve visibility.

File name display The file name of the image currently being annotated is now shown in the interface, simplifying dataset organization.

SALT outputs annotations in COCO-style `JSON` format, where each segment is assigned a unique incremental ID. After annotation, a custom script was used to merge all segments belonging to *stuff* classes and assign them the correct IDs, while ensuring that *thing* instances were consistently mapped.



(a) original **(b)** modified

Figure 4.6: Segment Anything Labeling Tool GUI. Notice the solid colors: the segments on the sky are barely visible in (a). No bounding boxes and file name on top-right corner in (b).

4.4 Annotations Format

This section provides a detailed description of the annotation format used in SPARTA and ATHENS, by outlining the different files and directories that compose the datasets.

Each version of the dataset is organized into a hierarchical directory structure, which includes separate folders for the training and validation sets. Within each of these, dedicated subdirectories are provided for storing the images and their corresponding annotations. Figure 4.7 illustrates the standard organization of the dataset, which is consistent across all datasets.

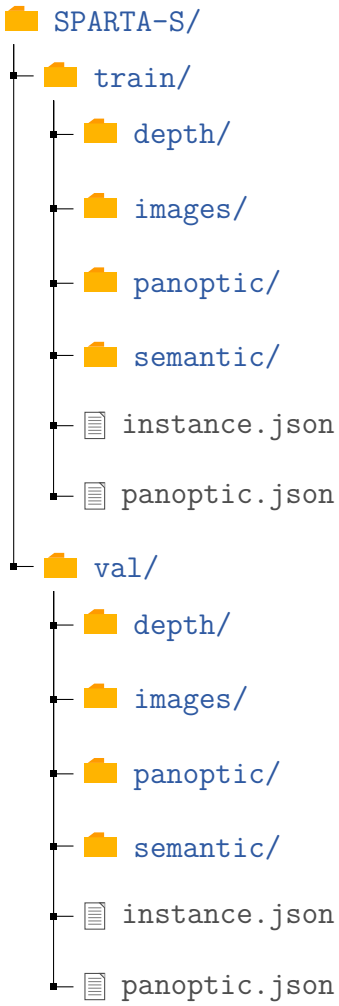


Figure 4.7: Folder structure of SPARTA-S dataset

panoptic.json

The `panoptic.json` file follows the COCO panoptic format and contains the following fields:

- `info`: metadata about the dataset, such as year, version, description, contributor, URL and date of creation.
- `images`: a list of the images in the dataset, each entry containing an `id`, `file_name`, `height`, and `width`.
- `licenses`: a list of dataset licenses, which in SPARTA and ATHENSremains empty.

- `categories`: a list of the categories, each defined by its `id`, `supercategory`, `name` and an `isthing` flag.

- `annotations`: a list of all the annotations for every image in the dataset. Each annotation contains three fields: the `id` of the image it belongs to, the `file_name` corresponding to the segmentation image name and the `segments_info` field, which contains a list of segments for that image.

  Each segment in `segments_info` has its own `category_id`, `id` (unique within the image and calculated as the `category_id` multiplied by 1000 to which an incremental instance ID is added), `area`, `bbox`(bounding box) and `iscrowd` flag.

  The `iscrowd` flag indicates whether the segment is a crowd of objects or not: it is used when there are multiple instances of objects that are too close to each other and manually annotating them would be too difficult or time-consuming, like a crowd of people at a concert. In the two proposed datasets, this flag is always set to 0, as there are no crowds of objects in the dataset.

instance.json

The `instance.json` file follows the same overall structure as `panoptic.json`, with the main differences lying in the `annotations` field.

Each annotation in `annotations` has an unique `id` (in this work, it was generated by multiplying the corresponding image ID by 1,000,000 and adding the segment ID), the corresponding `image_id`, `category_id`, `bbox`, `area`, and `iscrowd` fields, but it does not contain the `segments_info` field, which is replaced by the `segmentation` field, containing the spacial information of the location of the segment in the image. The other important difference is that this file only contains annotations for the *thing* classes, i.e. the Apple class for the SPARTA amd ATHENS datasets.

This `segmentation` field can contain information in two different formats:

- `RLE`: run-length encoding, which is a compact representation of the segmentation mask that allow to store the mask as a list of counts of consecutive pixels, alternating between background and foreground pixels. To better understand how it works, see Figure 4.8.

- `polygon`: a list of polygons that represent the segmentation mask. The polygons are represented as a list of points, each with its own pair of coordinates as shown in Figure 4.9.

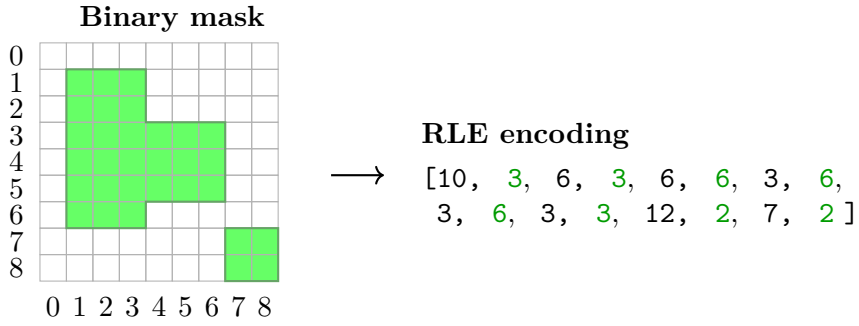


Figure 4.8: Illustration of Run-Length Encoding (RLE) for a binary segmentation mask. The mask is flattened into a 1D array and encoded as alternating runs of background (0) and foreground pixels (1). `[10, 3, 6, 3, 6, 6, 3, 6, ...]` means (left to right, top to bottom): 10 background pixels, 3 foreground pixels, 6 background pixels, etc.

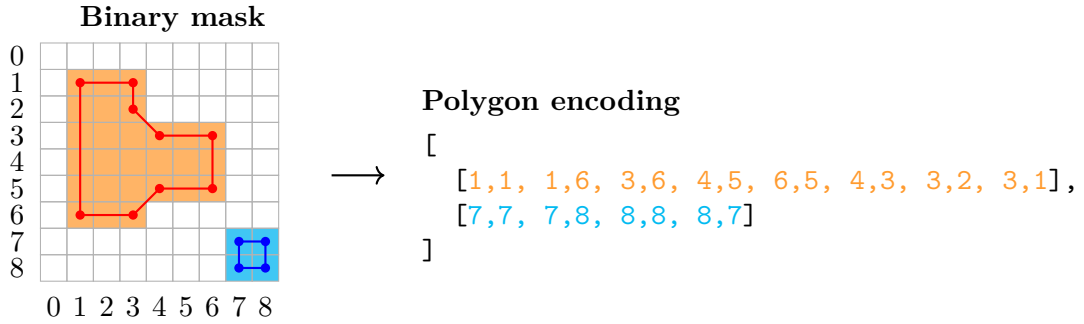


Figure 4.9: Illustration of polygon encoding for a segmentation mask. The mask is represented as a list of polygon vertices, where each polygon is defined by a list of coordinates flattened into a 1D array.

panoptic/

The `panoptic/` directory contains `.png` files representing the panoptic segmentation masks for each image in the dataset. These masks are RGB images, where each pixel color encodes the `segment_id` of the region it belongs to. The encoding is defined as follows:

- Red channel: `segment_id` mod 256
- Green channel: (`segment_id` ÷ 256) mod 256
- Blue channel: (`segment_id` ÷ 65536) mod 256

Symmetrically, the `segment_id` can be retrieved from the pixel color calculating

`segment_id = red + green * 256 + blue * 256 * 256`. These identifiers are unique within the image, and correspond to the ones in the `segments_info` field of the `panoptic.json` file.

semantic/

Similarly to the `panoptic/` directory, the `semantic/` directory contains `.png` files with the semantic segmentation masks for each image in the dataset. The only difference is that different instances of the same class are not distinguished, and all pixels belonging to the same class have the same color.

images/

The `images/` directory contains the original images of the dataset, in `.jpg` format.

depth/

Finally, the `depth/` directory contains a `.npy` file for each image representing for each pixel its depth expressed in meters.

Chapter 5

Methodology

To address the task of panoptic segmentation, this work explores and implements a range of deep learning techniques and architectures. The study begins with PanopticDeepLab, which operates on RGB images, and subsequently investigates the integration of depth information to enhance performance. In this context, architectures such as ESANet are employed to effectively leverage both RGB and depth modalities.

5.1 PanopticDeepLab

PanopticDeepLab [43] is the first bottom-up single-shot panoptic segmentation model able to achieve state-of-the-art results on common benchmarks. The design, shown in Figure 5.3, is really simple and requires only three losses to be trained.

This model leverages the Atrous Spatial Pyramid Pooling (ASPP) module, that was firstly introduced in the original DeepLab paper [44], to extract multi-scale features from the input image. It makes use of the so-called *atrous convolution*: a convolution that uses filters like the ones shown in Figure 5.1. These filters, despite having different sizes share the same number of parameters and operations per position, since only the non-zero filled values must be taken into account. They allow enlarging the *field-of-view* at any layer by just increasing the *rate* value, with zero consequences on the model performance.

ASPP is inspired by R-CNN spatial pyramid pooling [45] idea of extracting features at different scales. Multiple atrous convolutions with different sampling rates are used to extract features that are then fused to generate the final result.

The network is structured into two branches, one dedicated to semantic segmentation and the other to instance segmentation. These two branches share the same backbone, which is adapted from an ImageNet pre-trained neural network. Empirical evidence shows that the two branches require different contextual and

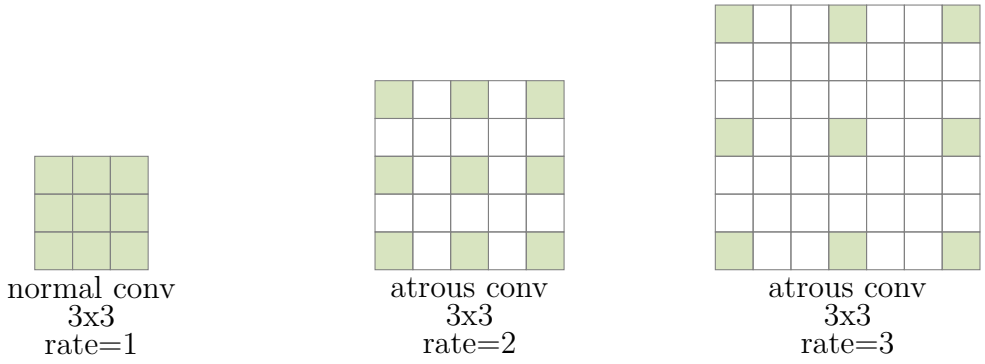


Figure 5.1: Example of atrous convolution filters. The *rate* determines the factor by which the filter is dilated. Empty values are filled with zeros.

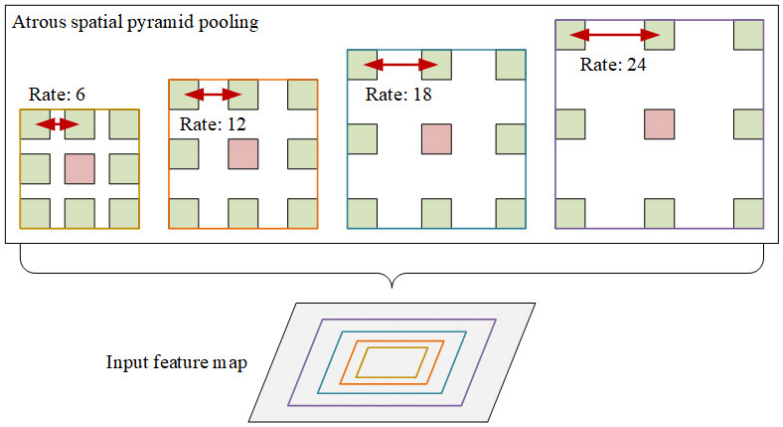


Figure 5.2: Atrous Spatial Pyramid Pooling (ASPP). To classify the center pixel, ASPP exploits multi-scale features by employing multiple parallel filters with different rates. The effective Field-Of-Views are shown in different colors.

decoding information; therefore, the authors propose to use two different ASPP modules, one for each branch. The outputs of these two modules are then fed separately to two different decoders (based on the DeepLabv3+ decoder [46] with a few modifications). Finally, two heads are attached to the decoders, concurring to the final panoptic prediction.

- **Semantic head**. The semantic head is attached to the semantic decoder and predicts both things and stuff, using a softmax activation function to output the class probabilities for each pixel. This head is trained using a weighted cross-entropy loss.
- **Class-agnostic instance head**. Each object instance is represented by a center of mass. So, this head predicts a center heatmap, which is a probability

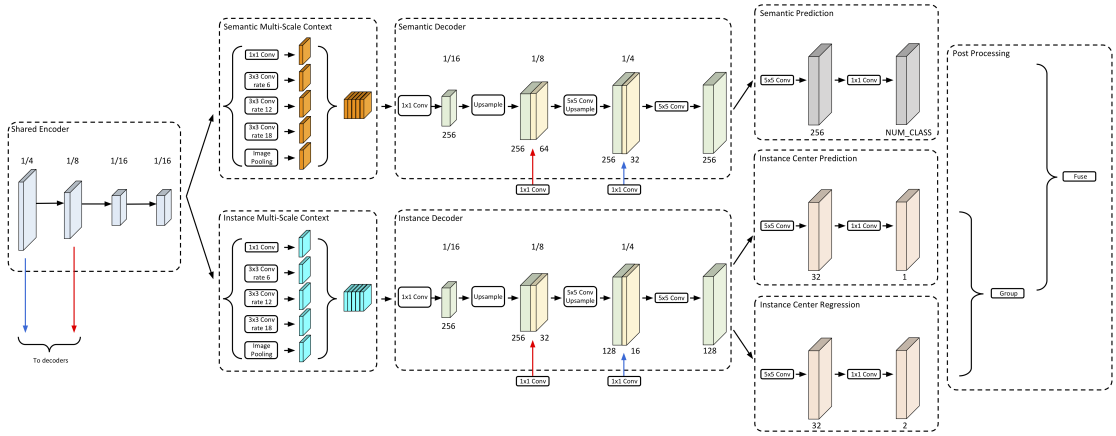


Image credits: PanopticDeepLab

Figure 5.3: Network architecture of PanopticDeepLab

map that indicates the likelihood of each pixel being the center of an object instance, and an offset map, which indicates for each pixel the offset to the center of the object instance. The center heatmap is trained using a MSE loss, while the offset map is trained using a L1 loss.

Now, to obtain the final panoptic segmentation, these three outputs (semantic, center heatmap and offset map) must be combined. Recalling Definition 1, the final panoptic segmentation is a pair $(l_{i,j}, z_{i,j})$ for each pixel of coordinates (i, j), where $l_{i,j}$ is the semantic class and $z_{i,j}$ is the instance id. The semantic class $l_{i,j}$ is simply obtained from the semantic head output, while a post-processing step is required to obtain the instance id $z_{i,j}$ from the center heatmap and offset map. In particular, a pixel of coordinates i, j is assigned to the closest center of mass C_z after applying the corresponding offset $\mathcal{O}(i, j)$, following the equation:

$$\hat{z}_{i,j} = \underset{z}{\operatorname{argmin}} \, ||C_z - ((i, j) + \mathcal{O}(i, j))||^2 \tag{5.1}$$

5.2 ESANet

ESANet (Efficient Scene Analysis Network) is an architecture proposed by Seichter et al. [47] in order to leverage depth information in addition to standard RGB inputs to perform semantic segmentation. ESANet is designed to grant faster inference time compared to other RGB-D segmentation methods.

The architecture of this network, shown in Figure 5.4, is inspired by SwiftNet [48] approach of using a shallow encoder with pretrained ResNet18 backbone and large downsampling, followed by a context module and a shallow decoder. SwiftNet

is designed for RGB images, therefore it has a single encoder, while ESANet introduces a second one.

Encoders. ESANet uses two encoders based on ResNet-34. However, unlike DeepLabv3 [49], strided convolutions are not replaced by atrous convolutions. Instead of the basic ResNet block, ESANet employs the *Non-Bottleneck-1D Block* (NBt1D), in which the 3×3 convolution is replaced by a 3×1 and a 1×3 convolution with a ReLU in between, as shown in Figure 5.4 (violet).

RGB-D Fusion. Unlike many other RGB-D segmentation approaches that fuse the feature representations later in the network, ESANet fuses depth features into the RGB encoder at each of the five resolution stages of the encoders. Features are reweighted using a Squeeze and Excitation (SE) module [50] and summed element-wisely, as depicted in 5.4 (light green).

Context Module. To give more context information, a Pyramid Pooling Module is used to aggregate features at different scales, Figure 5.4 (orange).

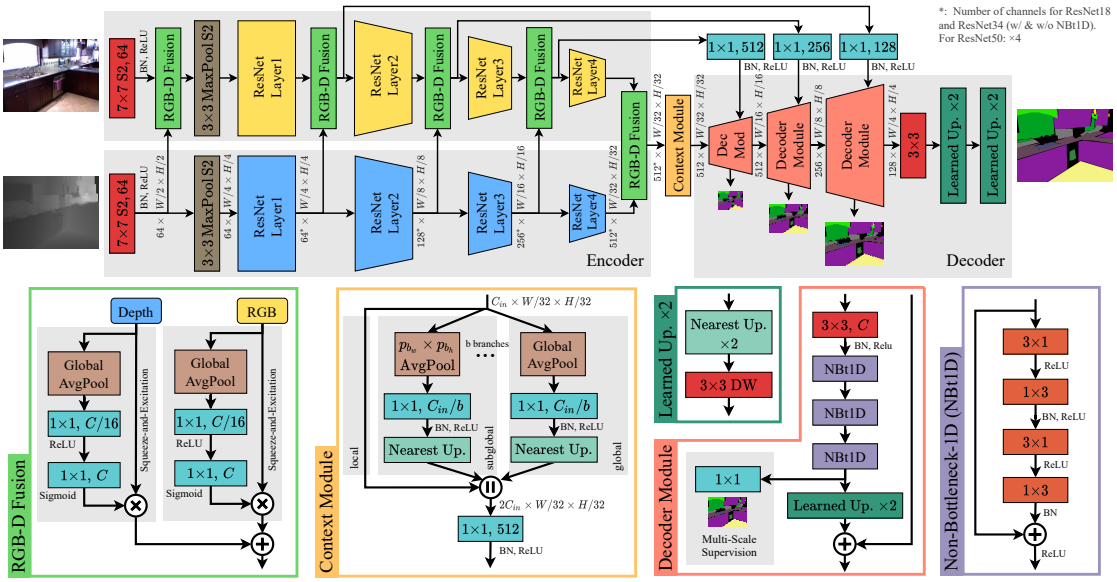
Decoder. The decoder is comprised of three modules, each composed of a 3×3 convolution, three NBt1D blocks and an upscaling module. Neither transposed convolution nor bilinear interpolation are used for upsampling, but a new learned module is used instead: a nearest neighbor upsampling followed by a 3×3 depthwise convolution. The resulting upscaled feature maps are then enriched with information coming directly from the encoders through skip connection. Loss is calculated not only at the end but also at the exit of each decoder module.

5.3 Improving ESANet

As described in Section 5.2, ESANet just performs semantic segmentation but can be easily adapted to provide panoptic outputs. That's what Sodano et al. [51] does, alongside providing a new fusion method called *ResidualExcite*.

Panoptic segmentation adaption. To adapt ESANet to panoptic segmentation two more decoders are added, one that predicts the location of object centers and the other that predicts an embedding vector for each pixel of the image. More in detail:

- **Semantic decoder** produces $I_{\text{sem}} \in \mathbb{R}^{C\times H\times W}$ outputs, where C is the number of semantic classes. It is trained on a cross-entropy loss $\mathcal{L}_{\text{sem}}$.



Legend: $\boxed{k_w \times k_h, C}$: Convolution with kernel size $k_w \times k_h$ and C output channels, S2: Stride 2, BN: Batch Normalization, Up.: Upsampling, DW: Depthwise, ⓘ: Concatenation

Figure 5.4: Overview of ESANet (top) and specific network parts (bottom).

- **Center prediction decoder** produces $I_{\text{cen}} \in \mathbb{R}^{1 \times H \times W}$ outputs. A sigmoid activation function predicts pixelwise probabilities of being a center. It is trained with a binary focal loss:

$$\mathcal{L}_{\text{cen}} = \begin{cases} -\alpha \, (1 - \hat{y})^{\tau} \, \log(\hat{y}) & \text{, if } y = 1, \\ -(1 - \alpha) \, \hat{y}^{\tau} \, \log(1 - \hat{y}) & \text{, otherwise,} \end{cases} \tag{5.2}$$

- **Embedding prediction decoder** produces a D_{emb}-dimensional embedding vector $I_{\text{emb}} \in \mathbb{R}^{D_{\text{emb}} \times H \times W}$. It is trained with a composed Hinge loss:

$$\mathcal{L}_{\text{emb}} = \beta_1 \, \mathcal{L}_{\text{att}} + \beta_2 \, \mathcal{L}_{\text{rep}} + \beta_3 \, \mathcal{L}_{\text{reg}} \tag{5.3}$$

  where $\mathcal{L}_{\text{att}}$ attracts embeddings of the same instance, $\mathcal{L}_{\text{rep}}$ repels embeddings of different instances and $\mathcal{L}_{\text{reg}}$ acts as a regularization term.

ResidualExcite. The second and main contribution of Sodano et al. [51] is the introduction of a novel feature fusion strategy. Unlike the Squeeze-and-Excitation module [50], which first generates a descriptor for each channel (squeeze) and then applies channel-specific weights to the feature map (excitation), ResidualExcite computes a single global modulation weight. This is achieved by removing the squeezing operation and incorporating a residual connection, as illustrated in Equation 5.4 and Figure 5.5.

$$\mathrm{X}_{\mathrm{rgb}} = \mathrm{X}_{\mathrm{rgb}} + \lambda \left(E(\mathrm{X}_{\mathrm{rgb}})\, \mathrm{X}_{\mathrm{rgb}} + E(\mathrm{X}_{\mathrm{depth}})\, \mathrm{X}_{\mathrm{depth}} \right) \tag{5.4}$$

Where λ is a weight hyperparameter and $E(\mathrm{X}_i)$ is the excitation module, composed of a sequence of 1×1 convolutions followed by a sigmoid activation.

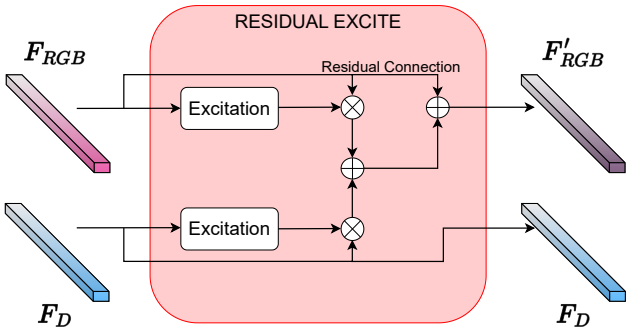


Figure 5.5: ResidualExcite module used to fuse RGB and depth features

5.4 PanopticDeepLab with Double Encoder

This variant of PanopticDeepLab integrates the double-encoder approach of ESANet into the original PanopticDeepLab architecture, which is based on a single ResNet encoder.

The first major difference lies in the stem. ESANet adopts the standard ResNet stem, consisting of a single 7×7 convolution with stride 2, followed by batch normalization, activation, and max pooling. This configuration rapidly reduces the spatial resolution of the input while expanding the feature channels to the desired output dimension. In contrast, PanopticDeepLab employs a customized stem composed of three consecutive 3×3 convolutions. The first convolution downscales the input with stride 2 while producing half of the final output channels. The second convolution preserves both the spatial dimensions and the number of channels, refining the intermediate features. The third convolution then restores the channel dimension to the full output size, preparing the representation for the subsequent stages of the network. This sequential design enables PanopticDeepLab to capture more detailed and progressively refined features compared to the single-step reduction in ESANet.

As discussed in Section 5.2, ESANet builds upon the Non-Bottleneck-1D block, whereas this PanopticDeepLab adaptation relies on the standard Bottleneck block from ResNet-50. The remainder of the backbone follows the same alternation of fusion operations and ResNet stages introduced by ESANet, as summarized in Figure 5.6.

Importantly, only the backbone of PanopticDeepLab is modified; all other components remain unchanged. Thus, the overall structure is identical to that described in Section 5.1, including the two separate ASPP modules, decoders, and heads, as well as the three independent loss functions.

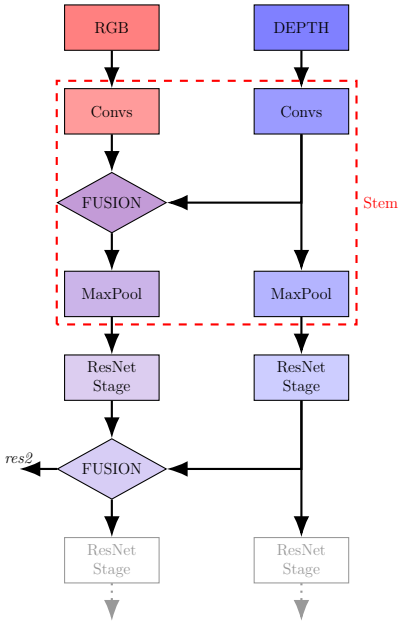


Figure 5.6: Backbone feature fusion in the double encoder version of PanopticDeepLab

Chapter 6

Experiments

This chapter presents the experiments conducted and their corresponding results. After a brief introduction to the experimental environment and the tools employed, each experiment is described and analyzed in detail, with comparisons among related groups and both qualitative and quantitative results reported. Additional details and results of the experiments can be found in Appendix B.

6.1 Environment and Framework

Environment All experiments were conducted on a workstation named *Bonnie*, located at the PIC4SeR laboratories in Turin. It runs `Ubuntu 22.04.1` and features two NVIDIA® GeForce™ RTX 2080 GPUs (8 GB each), an Intel® Core™ i7-9700K CPU at 3.60 GHz, and 65 GiB of RAM. The system is connected to a Network Attached Storage (NAS) unit with a total capacity of 17.3 TB.

As the workstation is shared among multiple users, the experiments were executed on a single GPU rather than utilizing the system's full capacity. Access to *Bonnie* was established via SSH, using Visual Studio Code's integrated SSH functionality for coding tasks and the Microsoft Remote Desktop (RDP) client for GUI-based operations.

Framework All experiments presented in this thesis were conducted using the **Detectron2** framework [52], developed by Facebook AI Research (FAIR) as the successor to Detectron, the original Caffe2-based implementation. The framework is widely adopted in both academia and industry due to its ease of use, high performance, and scalability to large datasets and multi-GPU environments. Detectron2 provides an extensive model zoo containing more than 70 pretrained models, implemented in a memory- and compute-efficient manner, allowing users to rapidly experiment with high-performing architectures.

Integrating a custom dataset into Detectron2 is straightforward, particularly when the dataset follows the COCO-style annotation format. For the SPARTA and ATHENS datasets used in this work, dataset registration was performed using the `register_coco_panoptic` function, after which the datasets were immediately available for training and evaluation. Each dataset can be associated with customizable metadata, such as `thing_classes`, `stuff_classes`, and `ignore_label`, enabling precise control over label semantics and evaluation protocols.

A standardized trainer abstraction simplifies and accelerates the training process by providing default configurations for key components such as the optimizer, learning rate schedule, logging, evaluation, and checkpointing. Detectron2 also streamlines the computation of performance metrics through a set of predefined evaluators that implement standard dataset-specific APIs, such as the `COCOPanopticEvaluator` for panoptic segmentation. Compared to running evaluations manually, these evaluators can be combined using the `DatasetEvaluators` wrapper, allowing multiple metrics to be computed in a single forward pass over the dataset.

The framework employs a flexible key–value configuration system based on YAML and `yacs`[1], which enables centralized and reproducible management of experimental settings. In addition, it provides standardized visualization utilities for qualitative inspection of model predictions, facilitating debugging and result interpretation.

6.2 Experiments with RGB-only inputs

The architecture used for the following experiments is **PanopticDeepLab**, described in Section 5.1. Unless otherwise stated, the input images are resized to 640×640 pixels before training and inference. The model is initialized from COCO pre-trained weights and fine-tuned on each dataset without freezing any layers. Three data augmentations are applied: random horizontal flipping, random cropping with resizing of the shortest edge, and color jitter.

For evaluation, Panoptic Quality (PQ), Segmentation Quality (SQ), and Recognition Quality (RQ) are used. Qualitative results are also reported to complement the quantitative analysis. A warmup polynomial learning rate schedule is used, with a learning rate that is increased up to 0.0025 over the first 50 epochs and then decayed by a factor of 0.9 until reaching 0.00001 at the end of training. On the hardware side, inference on GPU achieves an average of 41.3 ms per image ($\approx$ 24.23 FPS). On CPU, performance drops to 660.7 ms per image ($\approx$ 1.51 FPS).

[1] `https://github.com/rbgirshick/yacs`

6.2.1 Experiments on SPARTA

This set of experiments serves as a benchmark to assess the potential of PanopticDeepLab when trained exclusively on the synthetic SPARTA datasets. Having three versions of the dataset will allow to compare their relative performance and the effects of dataset size, resolution and variability on the final results.

SPARTA-S

In the first experiment PanopticDeepLab is trained and evaluated on the SPARTA-S dataset. Results are shown in Table 6.1, while Figure 6.1 provides qualitative samples. Performance is limited by the relatively small dataset size and low resolution. Errors often appear at object boundaries or in small objects that are too far. Nevertheless, the model is able to capture the main structural components of the images.

	PQ	SQ	RQ
All	61.035	81.595	74.395
Things	42.940	81.740	52.532
Stuff	64.654	81.566	78.767

Table 6.1: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained and evaluated on the SPARTA-S dataset.

(a) ground truth **(b)** prediction

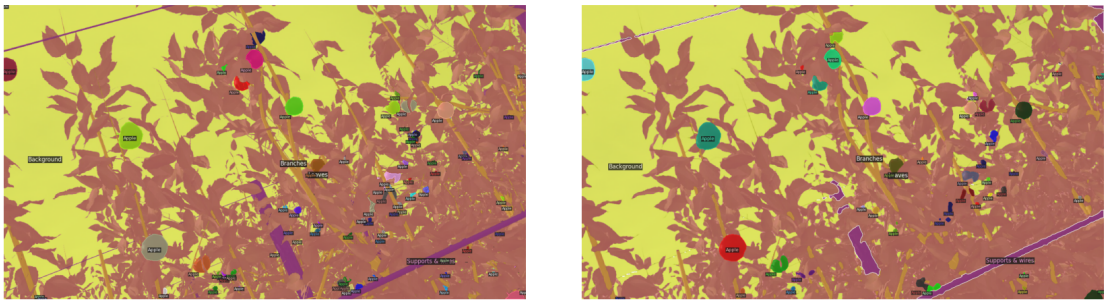
Figure 6.1: Qualitative panoptic results of PanopticDeepLab trained and evaluated on the SPARTA-S dataset.

SPARTA-L

This experiment follows the same training protocol but using the larger and higher-resolution SPARTA-L dataset. As reported in Table 6.2, performance significantly improves compared to SPARTA-S. This gain can be attributed both to the greater number of training images and to the increased spatial detail. Qualitative results (Figure 6.2) show more precise segmentation of fine structures and better class separation.

	PQ	SQ	RQ
All	72.250	84.367	85.624
Things	50.222	82.975	60.526
Stuff	76.656	84.646	90.643

Table 6.2: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained and evaluated on the SPARTA-L dataset.



(a) ground truth **(b)** prediction

Figure 6.2: Qualitative panoptic results of PanopticDeepLab trained and evaluated on the SPARTA-L dataset.

SPARTAv2

The third experiment employs the new SPARTAv2 dataset, designed with higher variability in textures, lighting, and object arrangements. Unsurprisingly, as shown in Table 6.3, performance metrics drop compared to SPARTA-L and SPARTA-S. However, qualitative results (Figure 6.3) reveal that the model still learns robust features, despite the more challenging dataset.

	PQ	SQ	RQ
All	55.001	77.214	70.811
Things	29.522	79.489	37.139
Stuff	60.097	76.759	77.545

Table 6.3: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained and evaluated on the SPARTAv2 dataset.

(a) ground truth **(b)** prediction

Figure 6.3: Qualitative panoptic results of PanopticDeepLab trained and evaluated on the SPARTAv2 dataset.

Summary and discussion

The following table (6.4) sums up the performance of PanopticDeepLab on the different versions of the SPARTA dataset. SPARTA-L completely outscores the other two datasets in all metrics, even though those numbers are not fully comparable since they are evaluated on different splits. It is worth noting that, although all three experiments show a drop in the *things* performance, the decrease is particularly pronounced for SPARTAv2. This can be explained by the more challenging apple representation in this dataset, which includes a wider variety of textures and lighting conditions. While this variability makes the task harder, it also provides a more realistic benchmark, highlighting the importance of robustness to visual diversity in real-world applications.

Train dataset	Val dataset	All			Things			Stuff		
		PQ	SQ	RQ	PQ	SQ	RQ	PQ	SQ	RQ
SPARTA-S	SPARTA-S	61.035	81.595	74.395	42.940	81.740	52.532	64.654	81.566	78.767
SPARTA-L	SPARTA-L	72.250	84.367	85.624	50.222	82.975	60.526	76.656	84.646	90.643
SPARTAv2	SPARTAv2	55.001	77.214	70.811	29.522	79.489	37.139	60.097	76.759	77.545

Table 6.4: Comparison of PQ, SQ, RQ metrics at best PQ step for each experiment on PanopticDeepLab trained on SPARTA dataset.

6.2.2 Cross-Dataset Generalization (SPARTA → ATHENS)

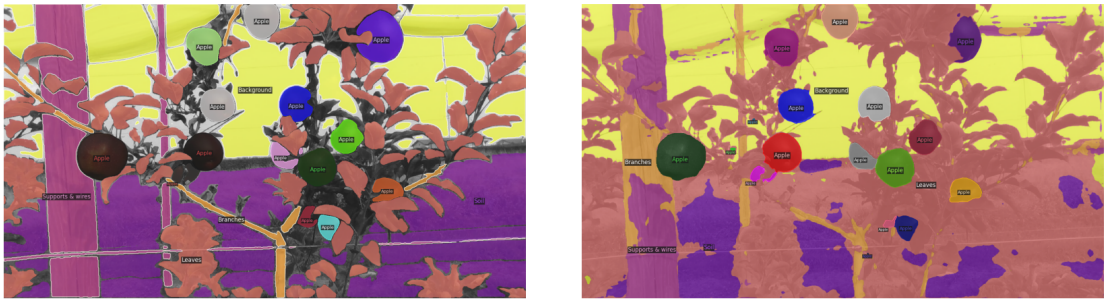
The main objective of the following experiments is to evaluate the degree to which models trained on synthetic datasets (SPARTA-S, SPARTA-L, and SPARTAv2) are able to generalize to real-world imagery, represented by the ATHENS dataset. This setup provides insights into the robustness of the learned representations and highlights the challenges of domain shift between synthetic and real data. Differences across SPARTA variants will allow to assess the impact of dataset size, resolution, and variability on generalization performance.

SPARTA-S to ATHENS

As a first test, the model trained on SPARTA-S is directly evaluated on the ATHENS dataset. The results in Table 6.5 indicate limited generalization, with the model struggling to adapt to the novel textures and scene layouts. Qualitative examples in Figure 6.4 reveal frequent misclassifications and fragmented segmentations, particularly among classes with similar appearance. Nevertheless, some categories, such as apples, are detected with reasonable accuracy.

	PQ	SQ	RQ
All	25.373	56.920	35.871
Things	20.270	65.541	30.928
Stuff	26.394	55.196	36.859

Table 6.5: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained on the SPARTA-S dataset and evaluated on the ATHENS dataset.



(a) ground truth **(b)** prediction

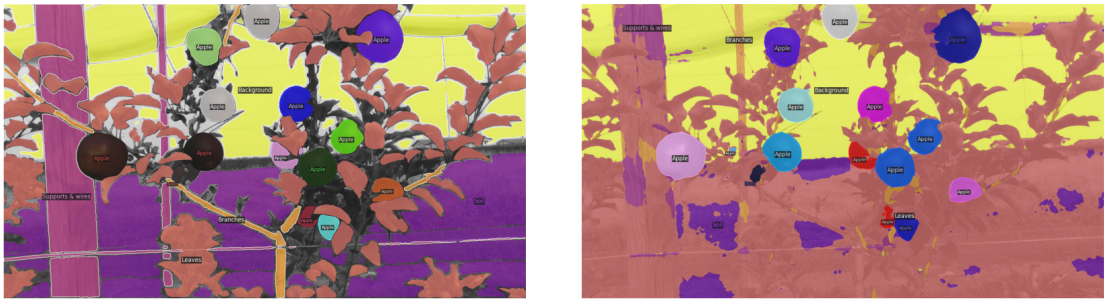
Figure 6.4: Qualitative panoptic results of PanopticDeepLab trained on the SPARTA-S dataset and evaluated on the ATHENS dataset.

SPARTA-L to ATHENS

Evaluating the SPARTA-L trained model on ATHENS yields higher quantitative scores, as reported in Table 6.6. Despite the qualitative examples in Figure 6.5 appear less convincing than those from the previous experiment, it is worth to note that the chosen image does not represent the whole validation set.

	PQ	SQ	RQ
All	27.799	69.374	39.454
Things	28.170	66.051	42.648
Stuff	27.725	70.039	38.815

Table 6.6: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained on the SPARTA-L dataset and evaluated on the ATHENS dataset.



(a) ground truth **(b)** prediction

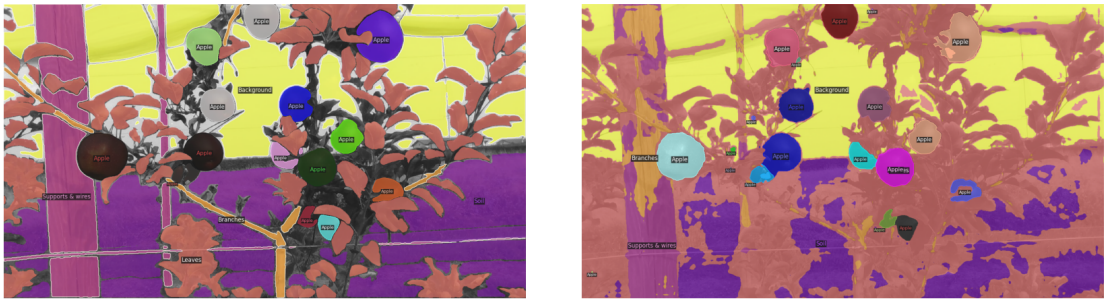
Figure 6.5: Qualitative panoptic results of PanopticDeepLab trained on the SPARTA-L dataset and evaluated on the ATHENS dataset.

SPARTAv2 to ATHENS

Finally, the model trained on SPARTAv2 is evaluated on ATHENS. Thanks to the greater diversity of SPARTAv2, the model generalizes better than the other two datasets, as shown in Table 6.7. Figure 6.6 illustrates qualitatively improved predictions, with more stable segmentations, fewer class confusions and more precise class borders. This demonstrates the effectiveness of dataset variability in improving robustness across domains.

	PQ	SQ	RQ
All	28.154	56.819	40.603
Things	22.995	64.252	35.789
Stuff	29.186	55.332	41.565

Table 6.7: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained on the SPARTAv2 dataset and evaluated on the ATHENS dataset.



(a) ground truth **(b)** prediction

Figure 6.6: Qualitative panoptic results of PanopticDeepLab trained on the SPARTAv2 dataset and evaluated on the ATHENS dataset.

Summary and discussion

Looking more in detail at the obtained results, summed up in Table 6.8, it is evident that SPARTA-S lags behind in all metrics, confirming its limited capability to generalize. SPARTA-L achieves the highest segmentation quality (SQ), which can be attributed to the larger image resolution available in this dataset, leading to more precise mask boundaries. On the other hand, SPARTAv2 reaches the best overall panoptic quality (PQ) and recognition quality (RQ), particularly in the *stuff* classes. This outcome is consistent with its more diverse and variegated image set, which favors better generalization across heterogeneous scenes.

Train dataset	Val dataset	All PQ	All SQ	All RQ	Things PQ	Things SQ	Things RQ	Stuff PQ	Stuff SQ	Stuff RQ
SPARTA-S	ATHENS	25.373	56.920	35.871	20.270	65.541	30.928	26.394	55.196	36.859
SPARTA-L	ATHENS	27.799	**69.374**	39.454	**28.170**	**66.051**	**42.648**	27.725	**70.039**	38.815
SPARTAv2	ATHENS	**28.154**	56.819	**40.603**	22.995	64.252	35.789	**29.186**	55.332	**41.565**

Table 6.8: Comparison of PQ, SQ, RQ metrics at best PQ step for each experiment about cross-dataset generalization.

6.2.3 Experiments on ATHENS

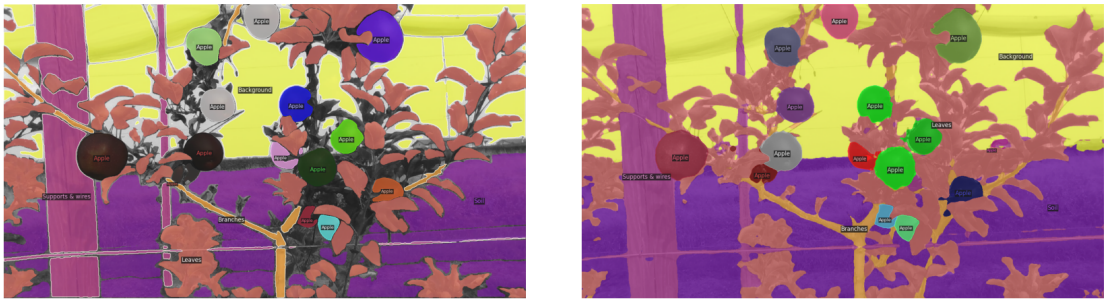
The following set of experiments aims to determine the most effective way to integrate ATHENS into the training process, whether by using only the real dataset, fine-tuning a model trained on SPARTA, or combining both datasets.

Direct Training on ATHENS

As a baseline, the model is trained directly on the ATHENS dataset, initialized from COCO-pretrained weights. This experiment establishes a reference point for subsequent fine-tuning approaches. The quantitative and qualitative results in Table 6.9 and Figure 6.7 already demonstrate strong performance, leaving limited room for further improvements beyond fine-grained refinements.

	PQ	SQ	RQ
All	66.684	83.649	79.901
Things	46.780	67.365	69.442
Stuff	70.665	86.906	81.993

Table 6.9: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained and evaluated on the ATHENS dataset.



(a) ground truth **(b)** prediction

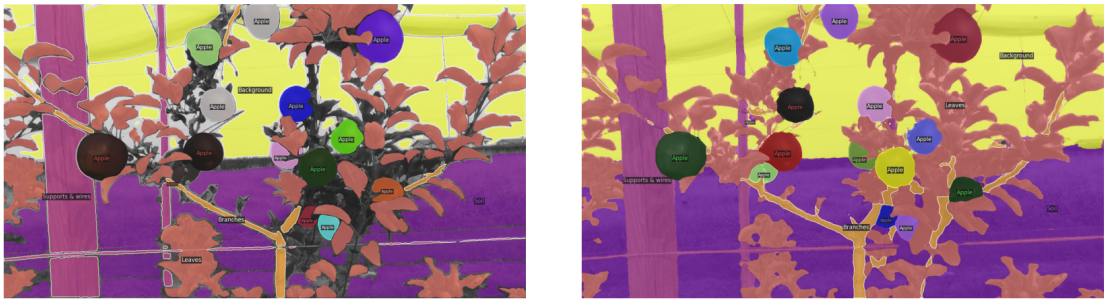
Figure 6.7: Qualitative panoptic results of PanopticDeepLab trained and evaluated on the ATHENS dataset.

Fine-tuning from SPARTA-S

As a first test, the model initialized with SPARTA-S pretraining is fine-tuned on the ATHENS dataset. As shown in Table 6.10 and Figure 6.8, the metrics significantly improve over the SPARTA-S → ATHENS direct evaluation, confirming the benefits of adaptation. However, when compared to the baseline trained directly on ATHENS, the gains remain limited.

	PQ	SQ	RQ
All	68.818	84.066	82.048
Things	48.429	67.021	72.259
Stuff	72.895	87.475	84.006

Table 6.10: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained on the SPARTA-S dataset, fine tuned on the ATHENS and then evaluated on the ATHENS dataset.



(a) ground truth **(b)** prediction

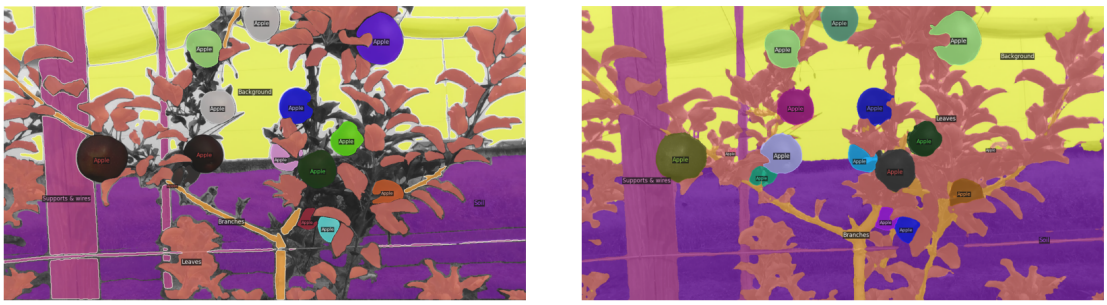
Figure 6.8: Qualitative panoptic results of PanopticDeepLab trained on the SPARTA-S dataset, fine tuned on the ATHENS and then evaluated on the ATHENS dataset.

Fine-tuning from SPARTA-L

The second experiment, fine-tuning the SPARTA-L pretrained model on ATHENS, follows the same trend observed with SPARTA-S. As shown in Table 6.11 and Figure 6.9, performance improves considerably compared to the direct SPARTA-L → ATHENS evaluation, yet only slightly surpasses the baseline trained directly on ATHENS. Interestingly, the final scores are even marginally lower than those obtained with SPARTA-S pretraining, suggesting that a larger synthetic dataset does not necessarily guarantee better transferability to the target domain.

	PQ	SQ	RQ
All	66.826	83.523	80.157
Things	47.456	66.886	70.952
Stuff	70.700	86.851	81.998

Table 6.11: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained on the SPARTA-L dataset, fine tuned on the ATHENS and then evaluated on the ATHENS dataset.



(a) ground truth **(b)** prediction

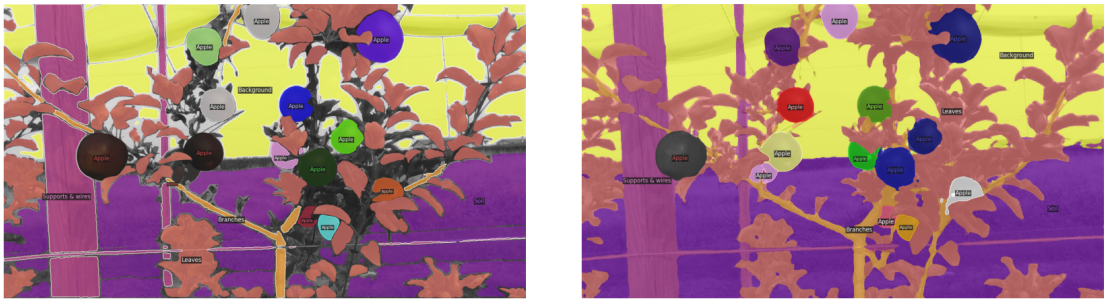
Figure 6.9: Qualitative panoptic results of PanopticDeepLab trained on the SPARTA-L dataset, fine tuned on the ATHENS and then evaluated on the ATHENS dataset.

Fine-tuning from SPARTAv2

Finally, this experiment leverages SPARTAv2 pretrained weights for fine-tuning on ATHENS. As in the previous cases, the general trend remains the same: fine-tuning yields a clear improvement over direct cross-dataset evaluation, while only slightly surpassing the baseline trained from scratch on ATHENS. However, this setup achieves the best overall results (Table 6.12 and Figure 6.10), indicating that the increased variability of SPARTAv2 provides the most effective pretraining prior among the three versions.

	PQ	SQ	RQ
All	69.084	85.071	81.288
Things	48.402	67.133	72.099
Stuff	73.220	88.658	83.125

Table 6.12: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained on the SPARTAv2 dataset, fine tuned on the ATHENS and then evaluated on the ATHENS dataset.



(a) ground truth **(b)** prediction

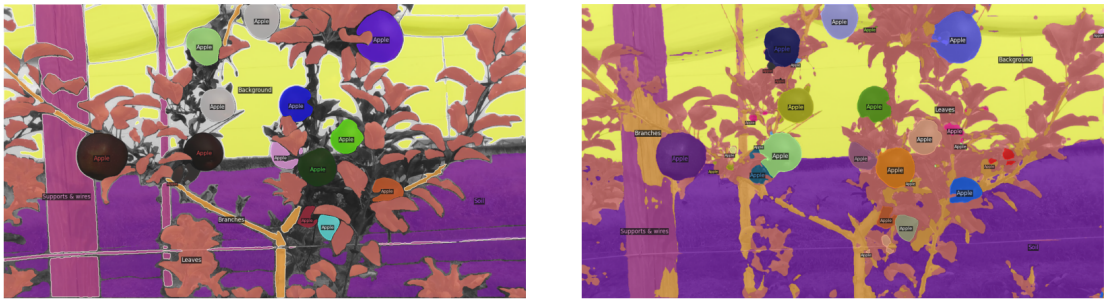
Figure 6.10: Qualitative panoptic results of PanopticDeepLab trained on the SPARTAv2 dataset, fine tuned on the ATHENS and then evaluated on the ATHENS dataset.

Mixing ATHENS & SPARTAv2

As an alternative to sequential training, first on the synthetic dataset and then fine-tuning on the real one, another approach is to jointly train on both datasets. However, the experimental results (Table 6.13 and Figure 6.11) indicate that this strategy does not achieve the same level of accuracy as fine-tuning.

	PQ	SQ	RQ
All	39.453	72.784	52.557
Things	37.375	66.168	56.485
Stuff	39.869	74.107	51.772

Table 6.13: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained on the SPARTAv2 and ATHENS datasets jointly and evaluated on the ATHENS dataset.



(a) ground truth **(b)** prediction

Figure 6.11: Qualitative panoptic results of PanopticDeepLab trained on the SPARTAv2 and ATHENS datasets jointly and evaluated on the ATHENS dataset.

Summary and discussion

To sum up, Table 6.14 reports all the results of this set of experiments in which ATHENS is integrated in the training. Overall, fine-tuning SPARTA-trained models on ATHENS generally improves performance compared to using ATHENS alone. The SPARTAv2 training followed by ATHENS fine-tuning achieves the highest PQ and SQ values overall, benefiting from the diverse representations of SPARTAv2 while leveraging the real-world images of ATHENS. In contrast, direct training on both SPARTAv2 and ATHENS without careful alignment leads to a dramatic performance drop, highlighting that naive mixing of datasets with different distributions can harm generalization.

		All			Things			Stuff		
Train dataset	Val dataset	PQ	SQ	RQ	PQ	SQ	RQ	PQ	SQ	RQ
ATHENS	ATHENS	66.684	83.649	79.901	46.780	**67.365**	69.442	70.665	86.906	81.993
SPARTA-S + ATHENS	ATHENS	68.818	84.066	**82.048**	**48.429**	67.021	**72.259**	72.895	87.475	**84.006**
SPARTA-L + ATHENS	ATHENS	66.826	83.523	80.157	47.456	66.886	70.952	70.700	86.851	81.998
SPARTAv2 + ATHENS	ATHENS	**69.084**	**85.071**	81.288	48.402	67.133	72.099	**73.220**	**88.658**	83.125
SPARTAv2 & ATHENS	ATHENS	39.453	72.784	52.557	37.375	66.168	56.485	39.869	74.107	51.772

Table 6.14: Comparison of PQ, SQ, RQ metrics at best PQ step for each experiment on PanopticDeepLab trained or partially trained on ATHENS.

6.3 Experiments with RBG+Depth inputs

In this set of experiments, the dataloader is configured to load both RGB images and corresponding depth maps. Since the depth data are stored as images with the same spatial resolution as the RGB inputs, the overall memory consumption is approximately doubled compared to the PanopticDeepLab experiments. As a result, the batch size must be reduced by half to maintain feasible training conditions. The experiments are conducted using two architectures: ESANet and a modified version of PanopticDeepLab designed to process both modalities.

6.3.1 Baselines

SPARTAv2 baseline

For a fair comparison, the baseline is defined as PanopticDeepLab trained on SPARTAv2 with a batch size of 2. The results, reported in Table 6.12 and Figure 6.12, show a decrease in performance. However, this reduction is not critical, as the purpose of this baseline is solely to provide a consistent reference for evaluating the impact of incorporating depth information with ESANet.

	PQ	SQ	RQ
All	46.920	73.323	63.496
Things	26.896	79.017	34.039
Stuff	50.925	72.184	69.388

Table 6.15: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained and evaluated on the SPARTAv2 dataset with a batch size of 2.

(a) ground truth **(b)** prediction

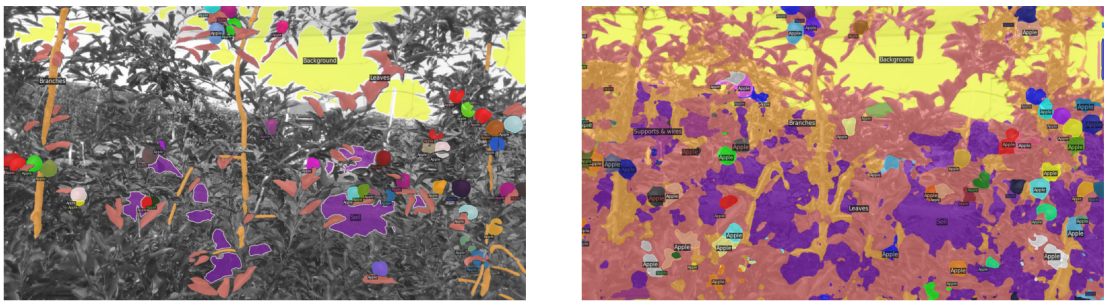
Figure 6.12: Qualitative panoptic results of PanopticDeepLab trained and evaluated on the SPARTAv2 dataset with a batch size of 2.

ATHENSv2 baseline

As already discussed in Section 4.3, the first version of ATHENS did not provide any depth information due to some errors in the archiving of the ROS bags. So, the real-life baseline is represented by the following results (Table 6.16, Figure 6.13) obtained on the ATHENSv2 dataset using a PanopticDeepLab architecture, still with a batch size of 2.

	PQ	SQ	RQ
All	58.218	77.565	76.740
Things	54.455	65.715	82.866
Stuff	58.971	79.935	75.515

Table 6.16: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab trained and evaluated on the ATHENSv2 dataset with a batch size of 2.



(a) ground truth **(b)** prediction

Figure 6.13: Qualitative panoptic results of PanopticDeepLab trained and evaluated on the ATHENSv2 dataset with a batch size of 2.

6.3.2 ESANet

The following experiments employ the **ESANet** architecture described in Section 5.2, with the panoptic adaptation and the *ResidualExcite* module proposed by Sodano et al. [51] described in Section 5.3. The main objective is to investigate whether incorporating depth information, readily available from the virtual environment, can improve the final results.

The original implementation by Sodano et al. [51], available at `https://github.com/PRBonn/PS-res-excite`, relies on a standard Python training loop. As a result, training requires several minutes per epoch, making hyperparameter optimization prohibitively time-consuming. To address this limitation, the entire architecture was re-implemented within the Detectron2 framework, achieving a speedup of several orders of magnitude, with each epoch taking only about 0.317 seconds. Moreover, Detectron2 provides a unified evaluation pipeline, ensuring consistent metrics across all experiments.

Unless otherwise stated, all experiments resize input images to 640×640 pixels before training and inference. Image-Net pretrained weights are used and training lasts 1000 epochs. The same three augmentations are applied: random horizontal flipping, random cropping with resizing of the shortest edge, and color jitter. For evaluation, Panoptic Quality (PQ), Segmentation Quality (SQ), and Recognition Quality (RQ) are used. Qualitative results are also reported to complement the quantitative analysis.

A one-cycle learning rate schedule is used, where the learning rate is first increased from an initial value of 1e−4 up to a maximum of 0.0025 during the first 10% of the training steps, and then gradually decreased following a cosine annealing strategy to a minimum of 1e−8 by the end of training. Depth inputs are preprocessed by applying a threshold, defined as a hyperparameter, to remove extreme values. Specifically, all depth values larger than the threshold are replaced with

the maximum valid value below it. After thresholding, the inputs are normalized using the mean and standard deviation of the depth values.

```
depth_inputs = [
    torch.where(
        x < self.depth_threshold,
        x,
        torch.max(x[x < self.depth_threshold])
    )
    for x in depth_inputs
]
```

Inference on GPU achieves an average of 76.4 ms per image (≈ 13.09 FPS), while on CPU performance decreases to 760.6 ms per image (≈ 1.31 FPS). In both cases, the model is slower than PanopticDeepLab, as it needs to process both depth and RGB information.

Experiment on SPARTAv2

In this experiment, the modified ESANet model was trained on the SPARTAv2 dataset with a batch size of 2. After some tuning, a threshold of 15 meters is chosen. The results, reported in Table 6.17 and Figure 6.14, indicate an overall improvement compared to the baseline. However, a closer analysis reveals a substantial drop in Panoptic Quality for the *thing* classes (in this case, apples), which is clearly visible in the qualitative results. While the model successfully segments all objects, it struggles to assign correct instance labels, leading to a significant reduction in instance-level performance.

	PQ	SQ	RQ
All	47.794	74.957	61.905
Things	9.650	74.913	12.881
Stuff	55.423	74.966	71.709

Table 6.17: Quantitative results (PQ, SQ, RQ) of ESANet trained and evaluated on the SPARTAv2 dataset.

(a) ground truth **(b)** prediction

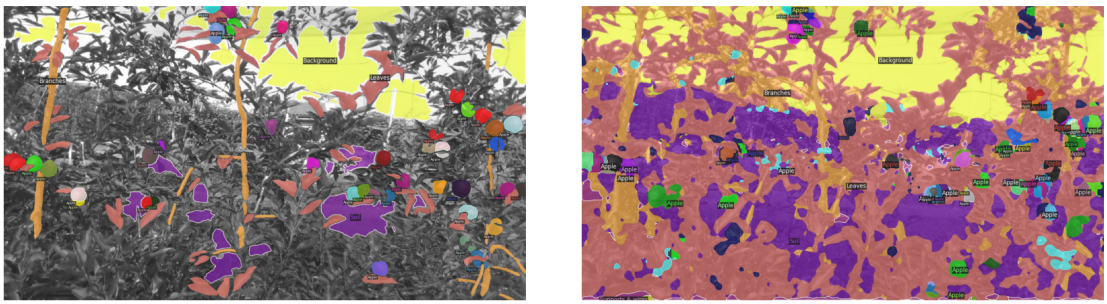
Figure 6.14: Qualitative panoptic results of ESANet trained and evaluated on the SPARTAv2 dataset.

Experiment on ATHENSv2

This second experiment performed using ATHENSv2 dataset and a threshold of 10 meters, shows a similar behavior to the previous ESANet experiment. The model is quite consistent in segmenting classes but struggles in identifying instances, leading to a lower Panoptic Quality for the *thing* class. However, unlike the previous experiment, the performance can not keep up with the baseline. The reason must be identified in the quality of depth data and the difference between the synthetic and real ones (see Figure 6.18).

	PQ	SQ	RQ
All	48.196	76.061	63.167
Things	11.651	61.320	19.000
Stuff	55.505	79.009	72.000

Table 6.18: Quantitative results (PQ, SQ, RQ) of ESANet trained and evaluated on the ATHENSv2 dataset.



(a) ground truth **(b)** prediction

Figure 6.15: Qualitative panoptic results of ESANet trained and evaluated on the ATHENSv2 dataset.

6.3.3 PanopticDeepLab with Double Encoder

The previous experiments revealed complementary strengths and weaknesses of PanopticDeepLab and ESANet. To exploit the advantages of both, a double-encoder variant of PanopticDeepLab is introduced in Section 5.4. The experiments presented here evaluate the effectiveness of this combined architecture.

For consistency, the baselines defined in Section 6.3.1 are retained, since halving the batch size remains necessary. Depth inputs are pre-processed using the same thresholding strategy described in Section 6.3.2, while the other training parameters are the same described for the simple PanopticDeepLab implementation described in Section 6.2.

Inference speed is almost on par with ESANet, with an average of 76.3 ms per image ($\approx$ 13.11 FPS) on GPU and 995.4 ms per image ($\approx$ 1.00 FPS) on CPU.

Experiment on SPARTAv2

In this experiment, the modified PanopticDeepLab is trained on SPARTAv2 dataset, using a batch size of 2 and a threshold of 20 meters. The improvement with respect to the baseline and the ESANet experiment is clear both from the quantitative (Table 6.19) and qualitative results (Figure 6.16). However, the overall performance gain remains limited, suggesting that while the inclusion of depth features provides marginal benefits, it does not lead to a substantial enhancement in panoptic quality.

	PQ	SQ	RQ
All	48.614	74.425	63.220
Things	23.102	76.657	30.137
Stuff	53.717	73.979	69.837

Table 6.19: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab with double encoder trained and evaluated on the SPARTAv2 dataset.

(a) ground truth **(b)** prediction

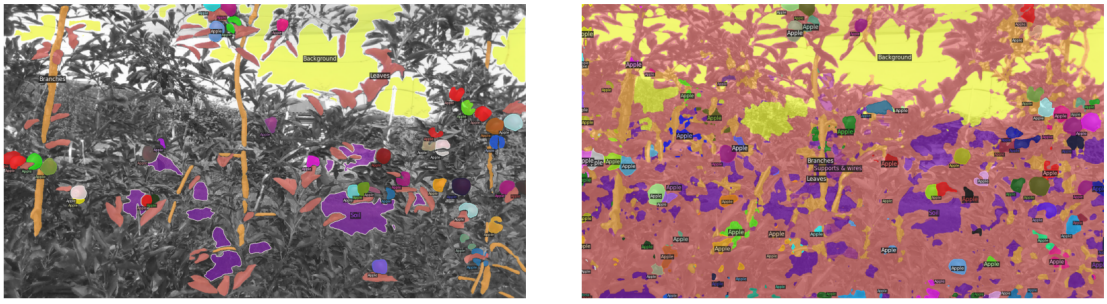
Figure 6.16: Qualitative panoptic results of PanopticDeepLab with double encoder trained and evaluated on the SPARTAv2 dataset.

Experiment on ATHENSv2

In this final experiment, the modified PanopticDeepLab was trained on the ATHENSv2 dataset, with a threshold of 20 meters identified as the best-performing setting. This experiment shows solid performance across all types of classes, as reported in Table 6.20 and illustrated in Figure 6.17. Although the model achieves higher Panoptic Quality values compared to the ESANet experiment, the baseline still outperforms it, largely due to the persistent issue of low-quality depth data.

	PQ	SQ	RQ
All	51.462	72.361	73.064
Things	51.095	65.186	78.383
Stuff	51.535	73.795	72.000

Table 6.20: Quantitative results (PQ, SQ, RQ) of PanopticDeepLab with double encoder trained and evaluated on the ATHENSv2 dataset.



(a) ground truth **(b)** prediction

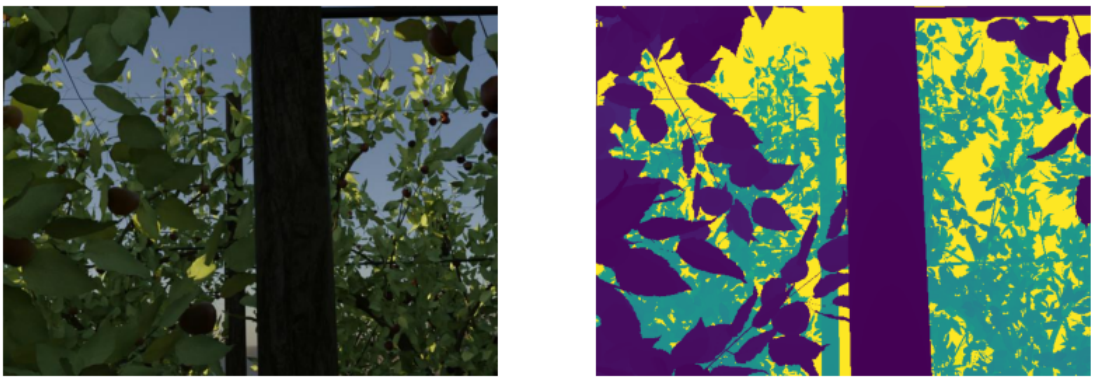
Figure 6.17: Qualitative panoptic results of PanopticDeepLab with double encoder trained and evaluated on the ATHENSv2 dataset.

6.3.4 Summary and discussion

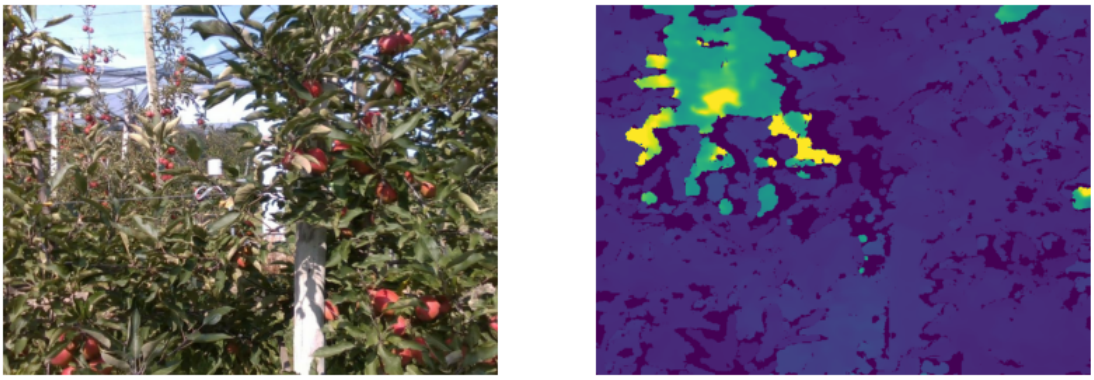
Two main findings emerge from this set of experiments. First, collecting reliable depth information in agricultural scenarios is particularly challenging. Depth cameras such as the `Intel® RealSense D435` suffer from limited precision and sensitivity to outdoor conditions, leading to noisy or incomplete measurements. Figure 6.18 shows the discrepancy between real and synthetic depth data.

Second, incorporating depth information nearly doubles the computational and memory requirements, while providing only marginal performance improvements. This trade-off also reduces inference speed, limiting the model's applicability in real-time settings.

A closer examination of Table 6.21, focusing on the synthetic dataset, shows that the modified PanopticDeepLab does not consistently outperform the other architectures in specific categories, but achieves slightly better results overall. ESANet excels on the *stuff* classes, while the baseline PanopticDeepLab achieves higher performance on the *thing* classes.



(a) SPARTAv2



(b) ATHENSv2

Figure 6.18: Comparison of depth data from the synthetic and real datasets.

Architecture	Train & Val dataset	All PQ	All SQ	All RQ	Things PQ	Things SQ	Things RQ	Stuff PQ	Stuff SQ	Stuff RQ
PanopticDeepLab (Baseline)	SPARTAv2	46.920	73.323	**63.496**	**26.896**	**79.017**	**34.039**	50.925	72.184	69.388
ESANet		47.794	**74.957**	61.905	9.650	74.913	12.881	**55.423**	**74.966**	**71.709**
PanopticDeepLab (Double)		**48.614**	74.425	63.220	23.102	76.657	30.137	53.717	73.979	69.837
PanopticDeepLab (Baseline)	ATHENSv2	**58.218**	**77.565**	**76.740**	**54.455**	**65.715**	**82.866**	**58.971**	**79.935**	**75.515**
ESANet		48.196	76.061	63.167	11.651	61.320	19.000	55.505	79.009	72.000
PanopticDeepLab (Double)		51.462	72.361	73.064	51.095	65.186	78.383	51.535	73.795	72.000

Table 6.21: Comparison of PQ, SQ, RQ metrics at best PQ step for each experiment with RGB+Depth inputs (Except for the baseline). All results refer to a training with batch size = 2.

Chapter 7

Conclusions

The main objective of this thesis was to address the lack of perception technologies in agricultural robotics that are able to understand the full scene context rather than focusing only on isolated elements. A review of the state-of-the-art approaches and publicly available datasets, in fact, confirmed that existing solutions tend to just identify limited portions of the images, typically where the objects of interest, such as the fruits, are located. Consequently, most computer vision methods for agricultural robotics revolve around the tasks of object detection or instance segmentation.

Given these limitations, the focus of this work was directed towards performing panoptic segmentation on fruit trees, more specifically on apple trees. To achieve this, two main contributions were introduced: the creation of two families of datasets, SPARTA and ATHENS, and the review and usage of two neural architectures, PanopticDeepLab and ESANet, on the aforementioned datasets.

SPARTA (Synthetic Panoptic Apple oRchard Tree Annotations) proved to be a highly valuable resource. Although it requires the initial effort of 3D modeling the virtual environment, it offers virtually unlimited potential: by simply adjusting simulation parameters, it is possible to generate a large variety of training data at minimal additional cost. ATHENS (Apple Tree Harvesting Environment with Natural Scenes) also demonstrated its usefulness. Despite its relatively small size, it provides high-quality panoptic ground truths for real-world images, making it a crucial complement to the synthetic dataset.

The experiments conducted on the two architectures, PanopticDeepLab and ESANet, have produced interesting results. First, the usage of RGB data alone for training and inference proved to achieve a solid performance. Employing pre-training on the synthetic SPARTA dataset as a starting point to further fine-tune on the real-word ATHENS dataset stood out as the most effective strategy. On the other hand, the integration of depth information brings a slight performance gain which also introduces significant drawbacks. Specifically, it comes at the cost

of almost doubling the computational power required, not to mention the effort to collect reliable depth data which can be difficult to obtain in real-world agricultural scenarios.

Overall, the obtained results can be useful for a wide range of applications that can diverge from the original goal of supporting fruit-harvesting robots. Understanding the whole context of a rural scene means having the opportunity to extract features like fruit size, leaf density or branches structure, which can be used to estimate the state of health or the productivity of the plant. For instance, monitoring fruit size over time can be helpful for regulating nutrients supply or guiding fertilization strategies. Similarly, the ratio between the quantity of fruits and leaves can be an indicator of plant productivity, allowing comparison between different cultivation practices or management techniques.

Several future works can build upon this thesis. The analyzed techniques can be easily extended to different crops, particularly those organized in rows such as vines or tomatoes. The collected datasets can be both expanded to increase variability and therefore, the ability of the models to generalize to unseen conditions. In addition to RGB and depth, other sensing modalities (thermal, multi-spectral cameras) can be tested to see if they bring performance improvements. The natural follow-up of this research is the design and implementation of a working robotic system, which can be tested on-field to evaluate both its effectiveness and efficiency.

Appendix A

Datasets examples and statistics

This appendix complements Chapter 4 by presenting examples of the dataset files and statistics.

A.1 Datasets statistics

The goal of this section is to provide a quick and compact overview of the statistics and differences between the introduced datasets through the following summary tables.

Dataset	**Split**	**Resolution**	**# images**	# ann. pixels per image	# apples	# apples per image
SPARTA-S	Train	640 × 480	800	307182.83	26575	33.22
	Val		200	307183.16	6654	33.27
SPARTA-L	Train	1280 × 720	3500	921587.16	139563	39.88
	Val		500	921587.28	19973	39.95
SPARTAv2	Train	640 × 480	8000	307163.41	475493	59.44
	Val		2000	307163.64	117821	58.91
ATHENS	Train	1280 × 720	132	379370.14	2740	20.76
	Val		32	389265.97	625	19.53
ATHENSv2	Train	1280 × 720	31	157418.13	1313	42.35
	Val		6	185917.17	330	55.00

Table A.1: Overview of dataset statistics and annotations for each split.

Dataset	Split	RGB		Depth	
		Mean	Std	Mean	Std
SPARTA-S	Train	`[56.93, 75.88, 39.96]`	`[43.18, 48.69 ,45.85]`	2.30m	2.05m
	Val	`[57.50, 76.96, 40.28]`	`[43.90, 48.86, 46.49]`	2.30m	2.11m
SPARTA-L	Train	`[41.14, 55.24, 32.05]`	`[50.11, 47.65, 53.38]`	2.68m	2.11m
	Val	`[40.59, 54.42, 31.31]`	`[49.55, 47.15, 52.53]`	2.72m	2.11m
SPARTA-V2	Train	`[62.63, 68.90, 46.38]`	`[42.75, 45.20, 46.14]`	3.07m	2.40m
	Val	`[63.08, 69.51, 47.12]`	`[43.34, 45.91, 47.18]`	3.05m	2.40m
ATHENS	Train	`[111.03, 106.09, 99.28]`	`[58.30, 58.70, 64.48]`	-	-
	Val	`[109.81, 104.80, 97.91]`	`[57.64, 58.13, 62.98]`	-	-
ATHENSv2	Train	`[105.80, 102.87, 98.45]`	`[54.04, 55.14, 61.84]`	1.84m	2.48m
	Val	`[103.22, 99.47, 89.66]`	`[58.00, 59.82, 68.95]`	2.70m	5.47m

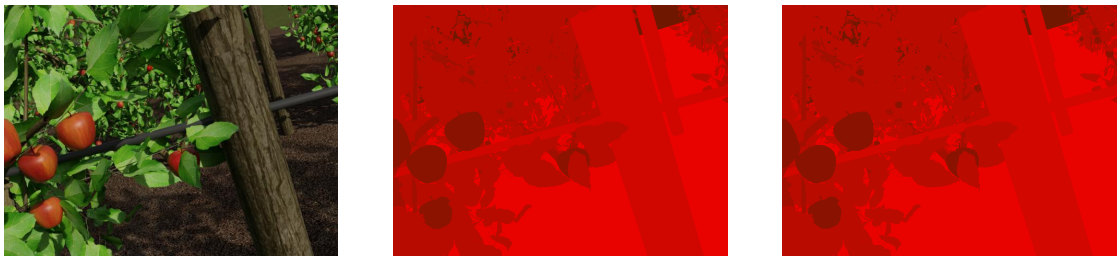
Table A.2: RGB and depth statistics over each dataset split. RGB values are in the range $0 - 255$, depth values are expressed in meters.

Dataset	Split	Apple	Soil	Supports & Wires	Leaves	Branches	Background
SPARTA-S	Train	1.89%	13.72%	4.39%	63.90%	6.00%	10.11%
	Val	2.03%	12.35%	4.47%	65.08%	5.62%	10.46%
SPARTA-L	Train	2.07%	14.97%	3.57%	64.24%	4.80%	10.35%
	Val	2.12%	15.57%	3.74%	64.01%	4.77%	9.79%
SPARTAv2	Train	2.02%	10.33%	4.30%	70.93%	5.36%	7.06%
	Val	2.01%	9.96%	4.23%	70.90%	5.34%	7.55%
ATHENS	Train	14.69%	14.92%	4.82%	31.84%	5.66%	28.07%
	Val	17.30%	13.38%	3.66%	33.46%	7.19%	25.01%
ATHENSv2	Train	15.62%	18.56%	6.86%	16.97%	2.29%	39.70%
	Val	13.52%	17.34%	9.16%	10.94%	2.40%	46.64%

Table A.3: Class distribution of annotated pixels for each dataset split.

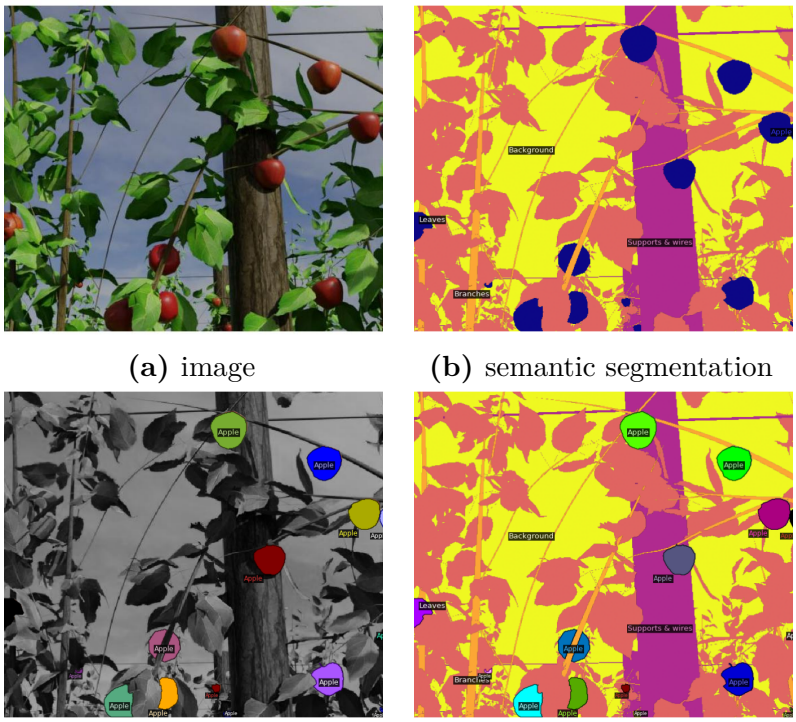
A.2 Dataset Samples

To provide a clearer overview of the datasets, this section presents sample files and illustrations from SPARTA and ATHENS.



(a) Original image **(b)** Panoptic image **(c)** Semantic image

Figure A.1: Example from the SPARTA-S dataset showing the original image and its corresponding raw panoptic and semantic segmentation files.



(a) image **(b)** semantic segmentation

(c) instance segmentation **(d)** panoptic segmentation

Figure A.2: Comparison of different segmentation tasks on an image of the SPARTA-S dataset (`Image58.jpg`).

```
{
    "info": {
        "year": 2025,
        "version": "1.0",
        "description": "SPARTA-S: Synthetic Panoptic
                        Apple oRchard Tree Annotations",
        "contributor": "PIC4SeR & VincenzoAvantaggiato",
        "url": "",
        "date_created": "2025/06/13 10:08:19"
    },
    "images": [
        {
            "id": 152,
            "file_name": "Image152.jpg",
            "height": 480,
            "width": 640
        },
        ...
    ],
    "licenses": [],
    "annotations": [
        {
            "image_id": 152,
            "file_name": "Image152.png",
            "segments_info": [
                {
                    "id": 6000,
                    "category_id": 6,
                    "area": 11534.0,
                    "bbox": [0, 0, 629, 439],
                    "iscrowd": 0
                },
                ...
            ],
        },
        ...
    ]
    "categories": [
        {
            "id": 1,
            "name": "Soil",
            "supercategory": "None",
            "isthing": 0
        },
        ...
    ]
}
```

Listing A.1: Example structure of `panoptic.json`

```
{
    "info": {
        "year": 2025,
        "version": "1.0",
        "description": "SPARTA-S: Synthetic Panoptic
                        Apple oRchard Tree Annotations",
        "contributor": "PIC4SeR & VincenzoAvantaggiato",
        "url": "",
        "date_created": "2025/06/13 10:08:19"
    },
    "images": [
        {
            "id": 152,
            "file_name": "Image152.jpg",
            "height": 480,
            "width": 640
        },
        ...
    ],
    "licenses": [],
    "annotations": [
        {
            "id": 152005002,
            "image_id": 152,
            "category_id": 5,
            "segmentation": [[17,266, 16,267, 15,267,
                              14,267, 13,267, 12,268,
                              13,268, 14,268, 15,269,
                              16,269, 17,268, 17,267]]
            "bbox": [12, 266, 6, 4],
            "area": 7.0,
            "iscrowd": 0,
            ],
        },
        ...
    ]
    "categories": [
        {
            "id": 5,
            "name": "Apple",
            "supercategory": "None",
            "isthing": 1
        }
    ]
}
```

Listing A.2: Example structure of `instance.json`

(a) SPARTA-S

(b) SPARTA-L

(c) SPARTAv2: Set 1

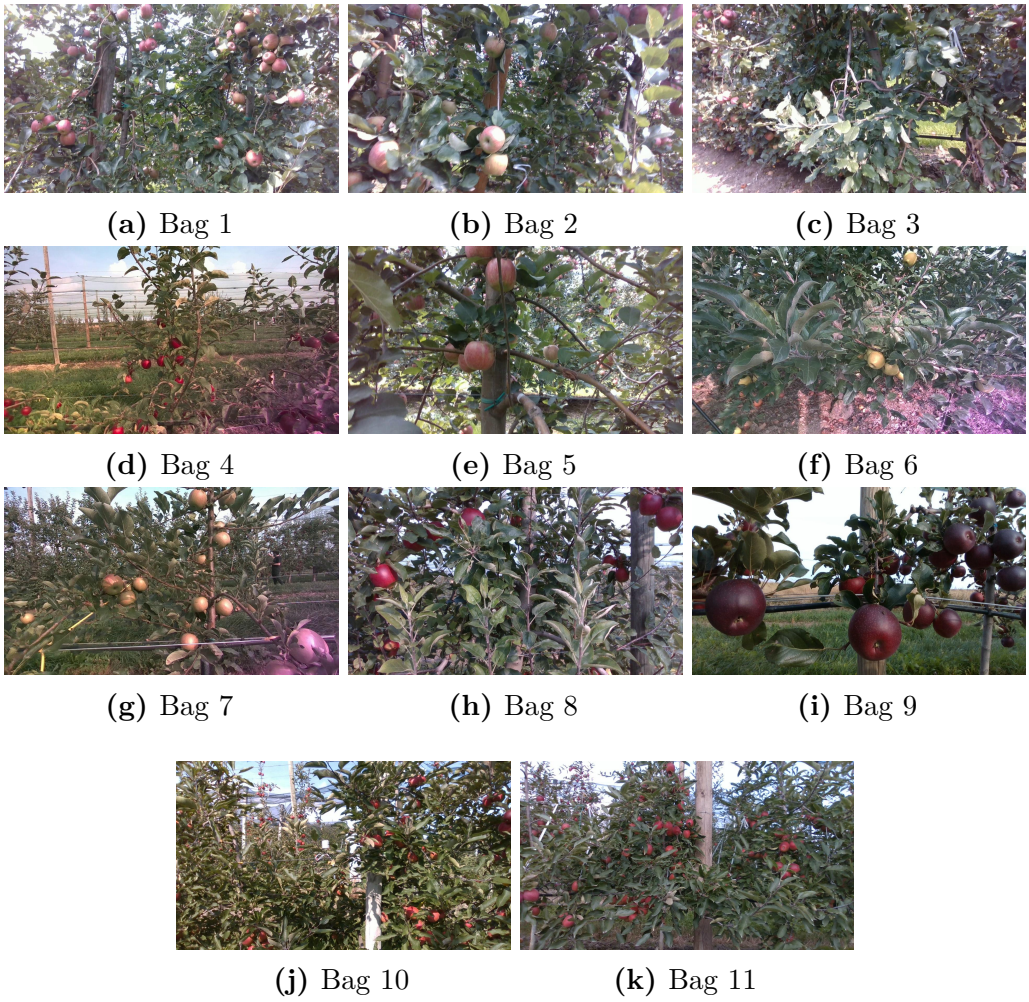
(d) SPARTAv2: Set 2

(e) SPARTAv2: Set 3

(f) SPARTAv2: Set 4

(g) SPARTAv2: Set 5

Figure A.3: Samples form SPARTA-S, SPARTA-L, and each of the five sets composing SPARTAv2.



(a) Bag 1 **(b)** Bag 2 **(c)** Bag 3

(d) Bag 4 **(e)** Bag 5 **(f)** Bag 6

(g) Bag 7 **(h)** Bag 8 **(i)** Bag 9

(j) Bag 10 **(k)** Bag 11

Figure A.4: Samples from ATHENS and ATHENSv2. Each sample belongs to a different bag.

Appendix B

Experimental results

This appendix complements Chapter 6 by summarizing the experimental results in tables and figures. While the main chapter provides a detailed discussion of individual experiments, the material presented here is intended to offer a concise overview, making it easier to compare models, evaluate performance across settings, and identify general trends.

Architecture	GPU		CPU	
	Time [ms]	FPS	Time [ms]	FPS
PanopticDeepLab	41.3	24.23	660.7	1.51
ESANet	76.4	13.09	760.6	1.31
PanopticDeepLab + Double Encoder	76.3	13.11	995.4	1.00

Table B.1: Average inference time and throughput on GPU and CPU.

Train dataset	Val dataset	All			Things			Stuff		
		PQ	SQ	RQ	PQ	SQ	RQ	PQ	SQ	RQ
SPARTA-S	SPARTA-S	61.035	81.595	74.395	42.940	81.740	52.532	64.654	81.566	78.767
SPARTA-L	SPARTA-L	72.250	84.367	85.624	50.222	82.975	60.526	76.656	84.646	90.643
SPARTAv2	SPARTAv2	55.001	77.214	70.811	29.522	79.489	37.139	60.097	76.759	77.545
SPARTA-S	ATHENS	25.373	56.920	35.871	20.270	65.541	30.928	26.394	55.196	36.859
SPARTA-L	ATHENS	27.799	**69.374**	39.454	**28.170**	**66.051**	**42.648**	27.725	**70.039**	38.815
SPARTAv2	ATHENS	**28.154**	56.819	**40.603**	22.995	64.252	35.789	**29.186**	55.332	**41.565**
ATHENS	ATHENS	66.684	83.649	79.901	46.780	**67.365**	69.442	70.665	86.906	81.993
SPARTA-S + ATHENS	ATHENS	68.818	84.066	**82.048**	**48.429**	67.021	**72.259**	72.895	87.475	**84.006**
SPARTA-L + ATHENS	ATHENS	66.826	83.523	80.157	47.456	66.886	70.952	70.700	86.851	81.998
SPARTAv2 + ATHENS	ATHENS	**69.084**	**85.071**	81.288	48.402	67.133	72.099	**73.220**	**88.658**	83.125
SPARTAv2 & ATHENS	ATHENS	39.453	72.784	52.557	37.375	66.168	56.485	39.869	74.107	51.772

Table B.2: Comparison of PQ, SQ, RQ metrics at best PQ step for each experiment on PanopticDeepLab with RGB input.

Architecture	Train & Val dataset	All			Things			Stuff		
		PQ	SQ	RQ	PQ	SQ	RQ	PQ	SQ	RQ
PanopticDeepLab (Baseline)	SPARTAv2	46.920	73.323	**63.496**	**26.896**	**79.017**	**34.039**	50.925	72.184	69.388
ESANet		47.794	**74.957**	61.905	9.650	74.913	12.881	**55.423**	**74.966**	**71.709**
PanopticDeepLab (Double)		**48.614**	74.425	63.220	23.102	76.657	30.137	53.717	73.979	69.837
PanopticDeepLab (Baseline)	ATHENSv2	**58.218**	**77.565**	**76.740**	**54.455**	**65.715**	**82.866**	**58.971**	**79.935**	**75.515**
ESANet		48.196	76.061	63.167	11.651	61.320	19.000	55.505	79.009	72.000
PanopticDeepLab (Double)		51.462	72.361	73.064	51.095	65.186	78.383	51.535	73.795	72.000

Table B.3: Comparison of PQ, SQ, RQ metrics at best PQ step for each experiment with RGB+Depth inputs (Except for the baseline). All results refer to a training with batch size = 2.

Bibliography

[1] European Commission. «The fruit and vegetable sector in the EU - a statistical overview». In: *Eurostat* (2024) (cit. on p. 1).

[2] European Commission. «Farmers and the agricultural labour force - statistics». In: *Eurostat* (2022) (cit. on p. 1).

[3] Lawrence G Roberts. «Machine perception of three-dimensional solids». PhD thesis. Massachusetts Institute of Technology, 1963 (cit. on p. 5).

[4] David Marr and Herbert Keith Nishihara. «Representation and recognition of the spatial organization of three-dimensional shapes». In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 200.1140 (1978), pp. 269–294 (cit. on p. 5).

[5] John Canny. «A computational approach to edge detection». In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698 (cit. on p. 5).

[6] Paul V. C. Hough. «Method and Means for Recognizing Complex Patterns». Patent US3069654A. Nov. 1962. URL: https://patents.google.com/patent/US3069654A (cit. on p. 5).

[7] Abdullah Ayub Khan, Asif Ali Laghari, and Shafique Ahmed Awan. «Machine learning in computer vision: A review.» In: *EAI Endorsed Transactions on Scalable Information Systems* 8.32 (2021) (cit. on p. 6).

[8] T Huang. «Computer Vision: Evolution And Promise». In: (1996). DOI: 10.5170/CERN-1996-008.21. URL: https://cds.cern.ch/record/400313 (cit. on p. 6).

[9] *What is Computer Vision? (History, Applications, Challenges)*. https://medium.com/@ambika199820/what-is-computer-vision-history-applications-challenges-13f5759b48a5. Accessed: 06-06-2025 (cit. on p. 6).

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «Imagenet classification with deep convolutional neural networks». In: *Advances in neural information processing systems* 25 (2012) (cit. on p. 7).

[11] *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)*. https://www.image-net.org/challenges/LSVRC/. Accessed: 08-06-2025 (cit. on p. 7).

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep residual learning for image recognition». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on p. 8).

[13] Alexey Dosovitskiy et al. «An image is worth 16x16 words: Transformers for image recognition at scale». In: *arXiv preprint arXiv:2010.11929* (2020) (cit. on p. 8).

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. «Deep learning». In: *nature* 521.7553 (2015), pp. 436–444 (cit. on p. 8).

[15] *Metrics Matter: A Deep Dive into Object Detection Evaluation*. https://medium.com/@henriquevedoveli/metrics-matter-a-deep-dive-into-object-detection-evaluation-ef01385ec62. Accessed: 04-06-2025 (cit. on p. 11).

[16] Irem Ulku and Erdem Akagündüz. «A survey on deep learning-based architectures for semantic segmentation on 2d images». In: *Applied Artificial Intelligence* 36.1 (2022), p. 2032924 (cit. on p. 12).

[17] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. «Panoptic segmentation». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 9404–9413 (cit. on pp. 14, 15).

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. «Microsoft coco: Common objects in context». In: *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*. Springer. 2014, pp. 740–755 (cit. on p. 17).

[19] Panagiotis Meletis, Xiaoxiao Wen, Chenyang Lu, Daan de Geus, and Gijs Dubbelman. «Cityscapes-panoptic-parts and pascal-panoptic-parts datasets for scene understanding». In: *arXiv preprint arXiv:2004.07944* (2020) (cit. on p. 17).

[20] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. «The mapillary vistas dataset for semantic understanding of street scenes». In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 4990–4999 (cit. on p. 18).

[21] Omar Elharrouss, Somaya Al-Maadeed, Nandhini Subramanian, Najmath Ottakath, Noor Almaadeed, and Yassine Himeur. «Panoptic segmentation: A review». In: *arXiv preprint arXiv:2111.10250* (2021) (cit. on p. 18).

[22] Jordi Gené-Mola, Ricardo Sanz-Cortiella, Joan R Rosell-Polo, Alexandre Escolà, and Eduard Gregorio. «PFuji-Size dataset: A collection of images and photogrammetry-derived 3D point clouds with ground truth annotations for Fuji apple detection and size estimation in field conditions». In: *Data in Brief* 39 (2021), p. 107629 (cit. on p. 19).

[23] Mar Ferrer-Ferrer, Javier Ruiz-Hidalgo, Eduard Gregorio, Verónica Vilaplana, Josep-Ramon Morros, and Jordi Gené-Mola. «Simultaneous fruit detection and size estimation using multitask deep neural networks». In: *Biosystems Engineering* 233 (2023), pp. 63–75 (cit. on p. 19).

[24] Jordi Gené-Mola, Ricardo Sanz-Cortiella, Joan R Rosell-Polo, Josep-Ramon Morros, Javier Ruiz-Hidalgo, Verónica Vilaplana, and Eduard Gregorio. «Fuji-SfM dataset: A collection of annotated images and point clouds for Fuji apple detection and location using structure-from-motion photogrammetry». In: *Data in brief* 30 (2020), p. 105591 (cit. on p. 19).

[25] Jordi Gené-Mola, Verónica Vilaplana, Joan R Rosell-Polo, Josep-Ramon Morros, Javier Ruiz-Hidalgo, and Eduard Gregorio. «KFuji RGB-DS database: Fuji apple multi-modal images for fruit detection with color, depth and range-corrected IR data». In: *Data in brief* 25 (2019), p. 104289 (cit. on p. 20).

[26] Nicolai Häni, Pravakar Roy, and Volkan Isler. «MinneApple: a benchmark dataset for apple detection and segmentation». In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 852–858 (cit. on p. 20).

[27] Yue Pan, Federico Magistri, Thomas Läbe, Elias Marks, Claus Smitt, Chris McCool, Jens Behley, and Cyrill Stachniss. «Panoptic mapping with fruit completion and pose estimation for horticultural robots». In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 4226–4233 (cit. on pp. 21, 23).

[28] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. «Mask R-CNN». In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322 (cit. on p. 21).

[29] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. «Deepsdf: Learning continuous signed distance functions for shape representation». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 165–174 (cit. on p. 21).

[30] Yuanyue Ge, Ya Xiong, and Pål J. From. «Instance Segmentation and Localization of Strawberries in Farm Conditions for Automatic Fruit Harvesting». In: *IFAC-PapersOnLine* 52.30 (2019). 6th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture AGRICONTROL 2019, pp. 294–299. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.

2019.12.537. URL: https://www.sciencedirect.com/science/article/pii/S2405896319324565 (cit. on pp. 21, 24).

[31] Matteo Sodano, Federico Magistri, Elias Marks, Fares Hosn, Aibek Zurbayev, Rodrigo Marcuzzi, Meher VR Malladi, Jens Behley, and Cyrill Stachniss. «3D Hierarchical Panoptic Segmentation in Real Orchard Environments Across Different Sensors». In: *arXiv preprint arXiv:2503.13188* (2025) (cit. on pp. 22, 24).

[32] Christopher Choy, JunYoung Gwak, and Silvio Savarese. «4d spatio-temporal convnets: Minkowski convolutional neural networks». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3075–3084 (cit. on p. 22).

[33] Leland McInnes, John Healy, Steve Astels, et al. «hdbscan: Hierarchical density based clustering.» In: *J. Open Source Softw.* 2.11 (2017), p. 205 (cit. on p. 22).

[34] Goran Paulin and Marina Ivasic-Kos. «Review and analysis of synthetic dataset generation methods and techniques for application in computer vision». In: *Artificial intelligence review* 56.9 (2023), pp. 9221–9265 (cit. on p. 25).

[35] Dean A Pomerleau. «Alvinn: An autonomous land vehicle in a neural network». In: *Advances in neural information processing systems* 1 (1988) (cit. on p. 25).

[36] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. «The synthetic data vault». In: *2016 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE. 2016, pp. 399–410 (cit. on p. 26).

[37] Shashank Tripathi, Siddhartha Chandra, Amit Agrawal, Ambrish Tyagi, James M Rehg, and Visesh Chari. «Learning to generate synthetic data via compositing». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 461–470 (cit. on p. 26).

[38] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. «Playing for data: Ground truth from computer games». In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer. 2016, pp. 102–118 (cit. on p. 26).

[39] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. «The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3234–3243 (cit. on p. 26).

[40] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. «Virtual worlds as proxy for multi-object tracking analysis». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4340–4349 (cit. on p. 26).

[41] Mauro Martini, Marco Ambrosio, Alessandro Navone, Brenno Tuberga, and Marcello Chiaberge. «Enhancing visual autonomous navigation in row-based crops with effective synthetic data generation». In: *Precision Agriculture* 25.6 (2024), pp. 2881–2902 (cit. on p. 28).

[42] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. «Robot Operating System 2: Design, architecture, and uses in the wild». In: *Science Robotics* 7.66 (2022), eabm6074. DOI: 10.1126/scirobotics.abm6074. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074 (cit. on p. 30).

[43] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. «Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 12475–12485 (cit. on p. 39).

[44] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. «Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs». In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848 (cit. on p. 39).

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Spatial pyramid pooling in deep convolutional networks for visual recognition». In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916 (cit. on p. 39).

[46] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. «Encoder-decoder with atrous separable convolution for semantic image segmentation». In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818 (cit. on p. 40).

[47] Daniel Seichter, Mona Köhler, Benjamin Lewandowski, Tim Wengefeld, and Horst-Michael Gross. «Efficient rgb-d semantic segmentation for indoor scene analysis». In: *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2021, pp. 13525–13531 (cit. on p. 41).

[48] Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. «In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 12607–12616 (cit. on p. 41).

[49] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. «Rethinking atrous convolution for semantic image segmentation». In: *arXiv preprint arXiv:1706.05587* (2017) (cit. on p. 42).

[50] Jie Hu, Li Shen, and Gang Sun. «Squeeze-and-excitation networks». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141 (cit. on pp. 42, 43).

[51] Matteo Sodano, Federico Magistri, Tiziano Guadagnino, Jens Behley, and Cyrill Stachniss. «Robust double-encoder network for rgb-d panoptic segmentation». In: *arXiv preprint arXiv:2210.02834* (2022) (cit. on pp. 42, 43, 61).

[52] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. *Detectron2*. `https://github.com/facebookresearch/detectron2`. 2019 (cit. on p. 47).