Politecnico di Torino

Collegio di ingegneria Informatica, del Cinema e Meccatronica
Master's degree in Mechatronic Engineering

Monitoring system for an apple orchard

Design, development and implementation of a monitoring system for an apple orchard, through the creation of an integral digital twin

Supervisors:
Elena Belcore
Andrea Botta

Candidate:
Martin Schneider Castro

April 2025

Abstract

The following thesis project describes the design, development and implementation of a monitoring system for an apple orchard through the creation of an integral digital twin (DT) of said orchard that supplies useful information about the apples, such as health and position. The work done for this thesis corresponds to the continuation of a project carried out by the Department of Environment, Land and Infrastructure Engineering (DIATI) and the Department of Mechanical and Aerospace Engineering (DIMEAS) of the Politecnico di Torino.

Data acquisition was performed using multispectral cameras and a stereo camera mounted on an agricultural UGV developed by DIMEAS, called Agri.Q, that was driven around the orchard by remote control. The multispectral cameras were used to measure the light reflected by the leaves during photosynthesis, indicating the health of every single tree. On the other hand, the stereo camera recorded an SVO file from which a point cloud was extracted and, thanks to the integrated Inertial Measurement Unit (IMU) and the possibility of using artificial intelligence (AI) and neural networks, apples were detected and located. In addition, a LiDAR scanner was used to obtain a georeferenced point cloud of the orchard.

The project required the design of the support for the sensors to be mounted on the UGV, taking into consideration the orientation and placement of the cameras needed to obtain the desired framing of the apple trees and the corresponding overlap to ensure a correct redundancy of the pictures. After data acquisition, point clouds were extracted from the outputs of the various cameras and sensors and were manipulated in order to obtain a single point cloud. It is interesting to mention the work done to recognise the apples: an AI model had to be trained to detect apples and then integrated into an application to recognise and locate the apples.

The aforementioned materials and procedures will be explained better throughout the thesis, together with the software and methods used. Furthermore, conclusions, limitations and future applications are also discussed at the end of the thesis.

Acknowledgements

This thesis would not have been possible without the guidance and support of my supervisors, Elena and Andrea. I am also deeply grateful to Kyra and Giovanni for their help, and to the team who authored the first paper of this project, whose groundwork and dedication greatly contributed to shaping this research.

A special thank you goes to the people of the Geomatics Lab, for their constant support, and for making this stressful period enjoyable.

Table of Contents

Chapter 1: Introduction

The agricultural sector is undergoing a technological transformation driven by precision agriculture and smart farming. This chapter provides an overview of key innovations in the field, starting with an introduction to robotics for agriculture, including UGVs (Uncrewed Ground Vehicles) and UAVs (Uncrewed Aerial Vehicles), followed by a discussion on crop monitoring. Various monitoring techniques are explored, such as remote sensing, range sensors, fruit detection using computer vision and AI, and digital twins. A literature review on orchard monitoring is then presented, highlighting existing approaches and technologies. Finally, the chapter identifies current limitations in the field and outlines the contributions of this research in addressing these gaps.

1.1 Precision agriculture and smart farming

Precision agriculture (PA) is *" the application of farming strategies and methodologies to do the right thing, in the right place and at the right time"* (Pierce & Nowak, 1999). In other words, PA is the employment of innovation technologies such as dedicated machines, sensors and information systems to improve the management and monitoring of agricultural production by collecting inputs that indicate the precise needs of the cultivated plants and soil (Vecchio et al., 2020). By monitoring in real-time their production, PA allows farmers to automate farming tasks and make better decisions regarding the management of their fields (Barrientos et al., 2011). The better management of farm data, ultimately, leads to better management of soil, resources and crops, while potentially boosting yields and profits, and also improving environmental quality (Botta et al., 2022). An interesting term in the context of PA is Management Zone (MZ). MZs are subregions of a field with relatively uniform yield-limiting factors and relatively homogeneous soil-landscape attributes, that can be used as baseline to make most agricultural decisions, allowing for the application of a single rate of a specific crop input to maximize input efficiency (Nawar et al., 2017).

Smart farming can be considered the evolution of PA, enhancing the principles of PA by integrating Information and Communications Technologies (ICT), including elements such as UGVs, Unmanned Aerial Vehicles (UAVs), machine learning, image processing and wireless sensor networks, among others (Moysiadis et al., 2021). This evolution occurs in the context of the era of Agriculture 5.0, in which the efficiency of agricultural production is increased by the application of emerging technologies such as AI, big data and the Internet of Things (IoT) (Ragazou et al., 2022).

Nowadays, food security is challenged by the intensification of global food production, consequence of the ever-growing global population and, therefore, food demand (Fróna et al., 2019). The global population is expected to grow to 9 billion people by 2050, and in consequence, agricultural consumption is estimated to increase by 69% (Sylvester, 2018). In addition, the scarce availability of skilled labour force increases the cost and reduces the sustainability of the harvest of specialty crops such as apples (Gongal et al., 2015). In particular, the apple industry is said to be in a state of hypercompetition, to the point that, when a harvest season begins, the fruit of the previous year's harvest has to be

thrown away (Harker et al., 2003). It is interesting to note that Italy is one of the main apple producers in Europe. Most of the apple production in Italy is concentrated in its northern regions. In fact, apple is the main fruit crop produced in Piedmont (Martino et al., 2024).

Smart farming and the integration of automation technologies into the modernisation of agricultural processes, especially harvesting methods, have played an important role in smartly and innovatively solving the aforementioned problems by quickly improving the efficiency and sustainability of food production (G. Zhang et al., 2024). To achieve this goal in a sustainable and environmental-friendly manner, special attention is paid to avoiding food waste caused by the decay of the product or its rejection due to the lack of the desired quality (Miranda et al., 2023). Here is where computer vision and artificial intelligence come into play.

1.2 Robotics for agriculture

The severe lack of farm workers and, in consequence, the increased need for efficient agricultural techniques can be supplied using intelligent machines. Robots are capable of doing repetitive tasks without ever losing precision, this presents them as a suitable option in the context of PA to apply advanced automated techniques and, ultimately, eliminating human involvement (Botta et al., 2022). These precision applications not only increase crop yield and quality but also reduce agriculture cost and environmental impact by optimising the use of water and nutrients (Bechar & Vigneault, 2016). Moreover, studies have shown that the use of robots or autonomous tractors saves fuel consumption and reduces air pollution (Gonzalez-de-Soto et al., 2016).

To justify, in most cases, the use of robots in agriculture, Bechar & Vigneault (2016) summarised a series of advantages:

- Employing robots is more economically convenient than employing other methods
- The use of robots increases production, profits, yields, product quality and uniformity
- Using robots minimises uncertainty and volatility
- Allows the farmers to make higher-resolution judgments
- Robots are able of performing dangerous tasks, or even tasks impossible to execute manually

As illustrated by Auat Cheein & Carelli (2013), the abilities and applications of agricultural robots can be grouped into four categories: mapping, action, guidance and detection. These implementation of service units are expected to work together, for they are essential one to another. For example, an action could not be executed if guidance was not successful; guidance would not be successful if mapping was not complete; mapping would not be complete if elements of the environment were not correctly detected (Auat Cheein & Carelli, 2013). Some of the precision applications carried out by agricultural UGVs are proximal sensing, seeding, planting, picking and harvesting, among others. However, in this project, special attention will be paid to the detection and proximal sensing applications of agricultural robots.

In this context, UGVs are being used to perform, predominantly, monitoring activities, in comparison to action applications (Smith et al., 2023). Agricultural UGVs exploited for monitoring the soil and the crop navigate through the field carrying sensors to collect data on plants and fruits conditions, presence of pests, composition of the soil and more (Vulpi et al., 2022). It is interesting to note the great versatility of the sensor configuration that can be mounted on a UGV, allowing to collect relevant data at different times using the necessary sensors, and adapting their configuration accordingly. This is the case of Agri.Q, the UGV used in this project, that, as it will be further explained in a later section, had its sensor layout reconfigured to satisfy the requirements.

Figure 1.1: Agricultural UGV Agri.Q equipped with robotic arm (Botta et al., 2022)

Another robotic tool largely used in PA applications is UAVs. UAVs are mostly used for vegetation and soil monitoring through Remote Sensing (RS), and for crop spraying (Radoglou-Grammatikis et al., 2020). While UAVs are widely used for aerial monitoring and large-scale data collection, UGVs offer a ground-based alternative with distinct advantages in certain applications. For example, UAVs can monitor individual plants, but their typical spatial resolution for Management Zones (MZs) is around 10 meters, which may not always provide enough detail for precise field management. Although advancements in UAV technology can enhance resolution, limitations persist. UGVs help address this issue by providing localized, high-precision data collection, allowing for more detailed assessments of plant and soil conditions (Botta et al., 2022). Moreover, despite their great mobility, UAVs are constrained by payload limitations, unlike UGVs, that have the advantage of being able to carry multiple sensors and robotic components, granting them greater versatility (Smith et al., 2023).

1.2 Crop monitoring

1.2.1 Remote sensing

RS and image analysing are a very popular tool in PA, allowing the assessment of soil conditions and vegetation health from a distance, using images most commonly obtained through aerial monitoring with satellites, manned aircrafts, UAVs, and, sometimes, ground vehicles, too (Botta et al., 2022).

The two main types of RS technologies are vision-based sensors and range sensors. Yandun Narvaez et al., (2017) divided optical-visible and near visible spectrum sensors and technologies into three groups:

- Structural characterisation: these sensors are used for estimating characteristics such as biomass, canopy volume, leaf area coverage, and plant height, among others.
- Physiology assessment: sensors are used to measure the physical response of the leaves to sunlight to obtain information about the health of the plant.
- Plant and fruit detection: physical characteristics of the plant and/or the fruits, usually colour and morphology, are detected to allow precise automation of agricultural actions such as harvesting and pruning.

As their name suggests, vision-based sensors are devices that capture and process visual information from the environment. Some common types of visual sensors are RGB cameras, stereo cameras, multispectral cameras, thermal cameras and Structured Light cameras. In this project, stereo cameras and multispectral cameras were used, and they will be explained thoroughly throughout the work, while the other types of sensors will be mentioned and commented for the sake of completeness.

The most basic vision-based sensors are RGB, or colour, cameras, used to obtain relevant information on physical characteristics of the terrain, plants or fruits. The most common use of colour cameras is the detection of fruits among the leaves of the trees. An important factor to be considered when using vision-based sensors is the influence of ambient lighting, which represents a major drawback of these applications. Detection is not possible using vision sensors alone, the integration with segmentation algorithms and AI detection algorithms, together with examples of applications, will be discussed further on.

Another type of vision-based sensors are stereo-vision systems. Stereo-vision systems use two or more cameras placed in different positions to capture images of the same scene, enabling depth perception. They allow the recreation of the environment in three dimensions through the creation of 3D point clouds. In particular, stereo-vision systems that use two cameras are designed to mimic human binocular vision. Studies that explore the use of this technology for fruit detection will be discussed further.

Some physiological parameters of plants and fruits can be studied through the measurement of radiation absorption and reflection in certain portions of the electromagnetic spectrum. For example, chlorophyll levels can be measured by measuring the reflection of green light, for chlorophyll absorbs red and blue light, while

it reflects green light. Other physiological parameters that can be studied include water stress and nitrogen content. The sensors that measure the reflectance of light are spectrometers or cameras that can be divided into multispectral or hyperspectral. Multispectral imaging measures the reflectance in a few, broad and not always continuous spectral bands. On the other hand, hyperspectral cameras measure the reflectance of light in a limited, continuous range of the electromagnetic spectrum. The parameters that researchers have studied using this kind of sensors will be discussed later in this work.

Thermal cameras are used for diagnostic and identification outside of the visible spectrum. Researchers have exploited plant temperature to study water availability (Baluja et al., 2012a), water stress (Jones et al., 2009), and fruit identification (Wachs et al., 2010). Another interesting application of these vision-based sensors is the enhancement of the ambient awareness of a robot through thermal imaging, allowing it to distinguish human operators or animals not visible due to deep vegetation (Reina et al., 2016).

Structured light cameras function by projecting an infrared pattern onto a surface and assessing the distortions in the reflected pattern to measure distances. Since their accuracy is highly dependent on lighting conditions, they are predominantly used in controlled environments like laboratories or greenhouses. These sensors are mainly employed to determine structural characteristics such as the size, height, and volume of plants and trees (Botta et al., 2022).

1.2.2 Range sensors

Ultrasound range sensing consists of a short-duration, high-frequency sound wave that travels through the air, reflects off a target, and returns as an echo. The sensor, then, calculates the distance from the target by considering the time delta between the moment the acoustic pulse was sent, and the moment it was received as an echo. This technique is commonly used for measuring volumes and densities. A similar technique, but with light pulses instead of acoustic pulses, is exploited by Time-of -Flight (ToF) cameras. The use of light instead of sound allows to obtain both 3D distance and intensity data, increasing the quality of plant characterisation and identification (Botta et al., 2022).

A similar principle to the one of ToF is shared by LiDAR (Light Detection and Ranging) sensors, with the main difference being that LiDARs typically use laser pulses to measure distances with higher precision and over longer ranges, making them more suitable for detailed 3D mapping and outdoor applications. There are 3D and 2D LiDARs, being the latter the most used, for it is less expensive and, with the right configuration, it can obtain 3D data, too. A particular kind of LiDAR is the hyperspectral LiDAR, that can recognise various wavelengths. In this project, a Terrestrial Laser Scanner (TLS), that is, a ground-based LiDAR, was used.

A summary of the aforementioned sensors, both remote and range, and the characteristics they measure agricultural applications is presented in Figure 1.2.

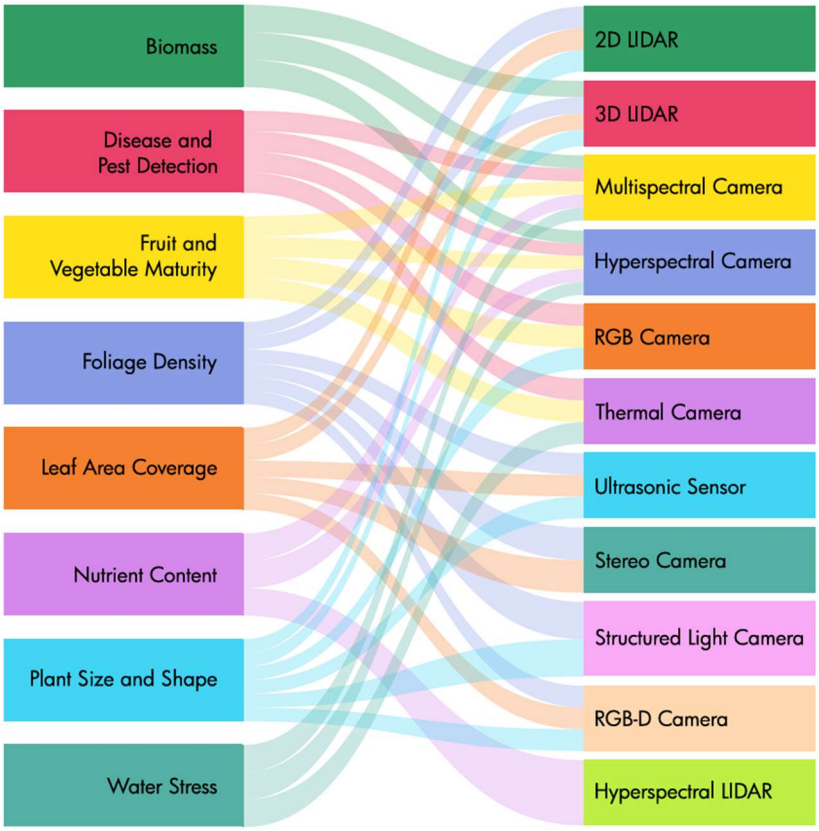

Figure 2.2: RS instruments and their uses (Botta et al., 2022)

1.2.3 Fruit detection through computer vision and AI

Several studies have been conducted regarding the use of neural networks and artificial intelligence to recognise apples and their characteristics, such as size, shape and colour, that indicate if the single fruit is an optimal candidate for harvesting and later commercialisation (Miranda et al., 2023).

Taking a step backwards, computer vision algorithms depended on human vision to design a methodology for the identification and extraction of image features, such as borders and shapes, and the classification of these features to identify parts of the image (Nanni et al., 2017). These "handcrafted" algorithms are preferred when a small amount of data is needed to train them (compared to deep neural networks), and thus, low computer power and memory are available (C. Zhang et al., 2020). Gongal et al. (2018) described this method in a paper in which a colour charged coupled device (CCD) and a ToF camera were used for the detection of apples. Their image processing consisted of equalising the histograms of the images in HIS (hue, intensity and saturation) colour space to amplify the colour difference between the apples and the background; then, a series of filters were applied to obtain a binary image with clear differentiation between foreground and background, and to recognise round objects, thus recognising the apples. Finally, the size of the apples was estimated through two different methods: 3D coordinates and physical size of image pixels.

Miranda et al., (2023) summarise the handcrafted fruit detection algorithm into two main steps:

i. Generation of candidate region proposals
ii. Detection and recognition

The purpose of the generation of candidate region proposals is to binarize data present on the image to fully differentiate the object of interest from the background, usually through thresholding. Region selection methods and their corresponding criteria can be summarised as follows (Miranda et al., 2023):

- Thresholding
  - Fruit reflectance
  - Geometric features
  - Temperature
  - Area of pixels
  - Depth
  - Colour

- Application of machine learning classifiers
  - k-means algorithm
  - Bayesian probabilistic classifier
  - k-nearest neighbours (KNN)
  - Support Vector Machine (SVM) procedures
  - Euclidean cluster, when working with 3D point clouds
  - Density-based spatial clustering and application of white noise (DBSCAN), when working with 3D point clouds

Together with region selection, region description must be performed to extract features of the objects of interest. Common descriptors are colour, shape and texture. After these features are obtained and classified, it is possible to proceed with object detection and recognition. Classifiers used to distinguish fruit from background are (Miranda et al., 2023):

- SVM
- KNN
- Adaboost
- Random forest
- Backpropagation neural network (BPNN)
- Gaussian mixture model (GMM)

On the other hand, deep learning has revolutionized computer vision, significantly advancing fruit detection, convolutional neural networks (CNNs) being the most widely used. CNNs are a particular kind of artificial neural networks (ANNs) that use convolution operations in at least one of their layers and are very efficient for image classification, especially for fruit classification (Naranjo-Torres et al., 2020). Some common computer vision tasks for which CNNs are used are:

- Image classification: classifying the image in a particular class
- Object detection: locating the object of interest in a region of the image by identifying its bounding box
- Semantic segmentation: labelling each pixel in the image into a class
- Instance segmentation: assigning each pixel to a single detected object

CNNs used for image classification consist of convolutional layers that extract features from the input image, followed by fully connected convolutional layers that act as classifiers. The convolutional layers generate feature maps by detecting patterns in the input image and refining them into more discriminative features through learned weights. At the final stage, the fully connected layers process these extracted features and classify the image into one of the predefined categories in the output layer (Miranda et al., 2023).

CNNs used for object detection consist of two structures: the backbone (first layers of an image classification CNN) and the head, that predicts object locations and their respective classes. They can be one-stage networks or two-stage networks, depending on how the head operates. The head of a two-stage network has a module that proposes regions of interest, and a module that classifies the regions into objects of interest or background, while also refining the bounding boxes. Two-stage CNNs are also called region-based CNNs (R-CNNs). Some typical examples are Fast R-CNN and Faster R-CNN. On the other hand, one-stage networks have heads with a single module that simultaneously assign classes and bounding boxes, without the need for suggestions of regions of interest (Sun et al., 2024; Miranda et al., 2023). One-stage CNNs are also called single shot detectors (SSD). An SSD commonly used in agricultural applications is You Only Look Once (YOLO) algorithm (Jocher et al., 2023).

YOLO is a state-of-the-art, real-time object detection system, able to accurately identify and classify objects into various classes. Its name derives from the fact that, unlike traditional object detection methods, this model works with a single pass of the image, in other words, looking at the image only once; this outstanding capacity grants significant speed to the real-time detection (*YOLO Object Detection Explained*, 2024). The algorithm is based on a deep CNN that detects objects in the input image. It divides the input image into a grid and then looks for the centre of objects and predicts bounding boxes in every single cell of the grid, assigning confidence scores for the boxes. This unified approach allows YOLO to perform simultaneously both object localization and classification, streamlining the detection process (Kundu, 2023).

In this project, YOLOv8 was used. Key improvements of YOLOv8 with respect to previous versions are (Torres, 2024):

1. Improved backbone architecture: This version uses the CSPDarknet53 backbone architecture, that enables the algorithm to better recognise intricate details in the image

2. Integration of PANet: Path Aggregation Network integrates information from different scales in the image, enhancing the recognition of objects present in different sizes

3. Introduction of dynamic anchor assignment: adapts anchor box dimensions during training, allowing for better recognition of objects with different shapes and sizes

4. Improved training process

5. Advanced augmentation techniques: integration of mosaic augmentation and self-paced learning

YOLO is currently in its 11th version (YOLOv11). However, YOLOv8 was chosen for better integration with previous work (Smith et al., 2023) and the sensors chosen for this project.

The YOLO model can be integrated with the SAHI (Slicing Aided Hyper Inference) library, which optimises object detection algorithms by slicing images and running object detection on each slice, to finally integrate the results of each slide. Although it improves detection when the objects of interest are small, it is not suited for real time detection (Ultralytics, 2025).

Results obtained by using deep learning detection methods will be discussed and compared in section 1.3.

1.2.4 Digital twins

A Digital Twin (DT) is *"a dynamic virtual representation of a physical object or system, usually across multiple stages of its lifecycle, that uses real-world data, simulation, or machine learning models combined with data analysis to enable understanding, learning, and reasoning. DT can be used to answer what-if questions and should be able to present insights in an intuitive way"* (Stanford-Clark et al., 2019). In the agricultural context, DTs can be defined as virtual representations of real agricultural systems that are continuously updated using smart farming technologies and sensors, allowing the management and storage of vast amounts of crop data (Smith et al., 2023).

According to Grieves (2015), a DT is composed of three main parts:

- Physical component, whose data will be acquired
- Virtual component, which is the digital representation of the physical component
- The stream of data and information that connects both components

DTs are predominantly used in the automotive, manufacturing, and energy sectors, but seldomly used in the agricultural sector, raising questions about their possible contribution to farming applications (Pylianidis et al., 2021). Smith et al. (2023) suggest that orchards could be optimal environments for testing the creation of DTs of living systems since trees can easily be differentiated one from the other, always remain in the same position, and are usually organised in easy-to-recognise patterns. Moreover, DTs of living systems could help increase product quality, lower costs of production, and detect diseases and other hazards (Pylianidis et al., 2021).

There has been limited research conducted on the use of agricultural digital twins (DTs); however, some relevant studies and their findings will be presented later.

1.3 Orchard monitoring

Some of the aforementioned technologies have been used and integrated for the monitoring of orchards. For example, Wang et al. (2013) conducted a study to estimate the crop yield of an apple orchard, of both red and green apples. For this scope, they mounted a stereo rig, composed of two identical high-resolution monocular cameras, on an autonomous orchard vehicle. After data acquisition, apple detection, registration and count were done on MATLAB. For red apples, the researchers obtained estimations per row significantly below the actual number of apples, but a small 3,2% standard deviation, indicating that the system is consistent. On the other hand, for green apples, the error of raw counts and standard deviation were significantly large. Two sources of error were identified, first, the software encountered difficulties detecting visible apples, particularly when dealing with fruit clusters with more than two apples. The second source of error affects green apples only, for the software could not recognise those with sunburn on their skins, or the ones in overexposed images.

Multispectral cameras are commonly used to study various physiological parameters of crops, such as the normalised difference vegetation index (NDVI), which is associated with leaf chlorophyll levels, leaf area index, crop biomass accumulation and photosynthetically active radiation absorbed by the canopy (Zaman-Allah et al., 2015). For example, Baluja et al. (2012) used a UAV equipped with a multispectral sensor Multiple Camera Array to assess water variability in a vineyard. They measured multispectral indices related to vine vigour, leaf stomatal conductance and stem water potential. In particular, NVDI showed a high coefficient of determination with the stem water potential and the leaf stomatal conductance of $R^2 = 0,68$ and Pearson correlation value of $p < 0,05$; moreover, the ratio between transformed chlorophyll absorption in reflectance and optimized soil-adjusted vegetation index (TCARI/OSAVI) indicated a high coefficient of determination with the stem water potential and the leaf stomatal conductance of $R^2 = 0,84$ and $p < 0,05$. Other indices that showed significant coefficients of determination when linking them to the stem water potential were MSR (modified simple ratio) and SRI (simple ratio index).

A study to compare the dynamic accuracy of versions 4, 5 and 7 of YOLO was carried out by Abeyrathna et al. (2023). The study consisted of detecting and counting apples in real-time with a stereo camera mounted on a forward-moving tractor using YOLOv4, TOLOv5 and YOLOv7. The results obtained by the authors (presented in Table 1.1), comparing precision, recall, F1-score metric and mean average precision at IoU (intersection over union) at threshold of 0,5 (mAP@0,5), clearly show better performance by YOLOv7:

Model	Precision	Recall	F1	mAP@0,5
YOLOv4	0,840	0,790	0,810	0,840
YOLOv5	0,874	0,783	0,830	0,861
YOLOv7	0,892	0,828	0,860	0,905

Table 1.1: Performance evaluation of YOLO models (Abeyrathna et al., 2023)

Another relevant study was conducted by Chu et al. (2023) to confront the performance of a novel, self-developed, deep learning-based apple detection framework, Occluder-Occludee Relational Network (O2RNet), against other state-of-the-art models. All the models were given the same customised apple dataset. The evaluation metrics the authors compared were AP at different thresholds, average recall (AR) at different thresholds and the F1-score. The results are presented in Table 1.2. Although the researchers tested multiple versions of some of these models, only the results of the most efficient versions are presented. Also, only the average precision and recall at a threshold of 0,5 are presented.

Model	**AP**	**AP@0,5**	**AR**	**AR@0,5**	**F1**
FCOS	0,48	0,89	0,34	0, 87	0,80
YOLOv4	0,45	0,87	0,29	0,84	0,76
Faster R-CNN ResNet101	0,49	0,94	0,31	0,84	0,82
EfficientDet-b5	0,50	0,95	0,34	0,88	0,83
CompNet via RPN	0,51	0,95	0,35	0,94	0,86
O2RNet-ResNet101	0,52	0,96	0,36	0,94	0,86

Table 1.2: Performance evaluation of object detection models (Chu et al., 2023)

As mentioned before, not many papers that discuss the functioning DTs of living plants or trees have been published. However, Verdouw & Kruize (2017) present some interesting examples of DTs for farm management. An interesting project is OLIFLY that consists of the DT of olive trees to monitor the presence of olive flies and the pest traps through real-time image acquisition. A similar, but more rudimental, project is Open PD, used for pest detection, in which the DT consists of pictures and descriptions provided by the users. The authors also present other cases related to the monitoring of farm equipment and structures. Pylianidis et al. (2021) present other cases of DTs for agricultural applications. It is worth noting that no studies regarding a complete workflow for the creation and deployment of DTs of living systems have been found.

1.4 Current limitations and proposed contributions

Despite the great advances of the PA technologies, the reviewed literature showed some significant gaps and room for improvement that this work aims to fill.

First, there is a lack of fully integrated systems combining UGVs and real-time monitoring systems. While UGVs and monitoring technologies have been extensively studied separately, most studies focus on either mobility and automation or sensor-based data collection, but not on a seamless system that enables both. In addition, most of the current agricultural monitoring systems rely on data collected during specific survey periods and later processed offline. However, real-time data acquisition and processing could improve farm management efficiency, allowing for immediate intervention when hazards to the crops are detected, or identifying the best harvesting period, according to the physical characteristics of the fruits. This project could potentially solve these first two issues, by using an agricultural UGV mounted with stereo cameras and sensors able to detect and assess apples in real-time.

Many existing studies on fruit and crop monitoring focus on 2D image analysis, overlooking the possibility of 3D analysis, which provides valuable information but lacks depth perception. Here, dedicated instruments for 3D analysis of crops are integrated. In particular, a stereo camera and a TLS are used to provide 3D images of the fruits of interest, enabling better volume estimation, shape analysis, and occlusion handling, leading to more accurate assessments of fruit conditions.

Another gap noted among the reviewed works is the georeferencing of individual fruits, which is currently an emerging area of study. Through georeferencing, either in a global or local system, it is possible to co-register spatial information collected at different acquisition times. This enables orchard monitoring, for example, by performing change detection of the fruit present on each tree and health indices differences in time. One of the main scopes of this project is georeferencing apples thanks to the integration of YOLO and the stereo camera, which could benefit yield mapping, robotic harvesting, and logistics optimisation.

Finally, the scope of the work is to create a DT of the apple orchard, providing a workflow and basis on how to reproduce it. As already mentioned, there are few, if not at all, deployed DTs of living systems, such as trees.

This project proposes the creation of an automated real-time orchard monitoring system, obtaining a DT containing relevant information on the monitored apples, such as health and size. To this aim, several of the aforementioned PA technologies are integrated into a single system, such as the use of a dedicated UGV, stereo vision enhanced with AI, and multispectral cameras.

Chapter 2: Materials

To address the need for a real-time and 3D monitoring system in orchards and the creation of their DTs, this study employs a mobile robotic platform equipped with advanced monitoring instruments capable of capturing detailed data of the crops. This chapter outlines the design and specifications of the mobile robot, including its key features. Additionally, the instruments mounted on the robot, such as a stereo camera and multispectral cameras, are detailed, highlighting their roles in providing the high-resolution and multidimensional data essential for precise orchard monitoring. Finally, the steps previous to data acquisition are described.

2.1 Agri.Q

Agri.Q is an agricultural UGV developed by the DIMEAS department of Politecnico di Torino designed to achieve locomotion efficiency on unfavourable ground conditions, such as slippery, wet and or uneven terrain, when performing precision agriculture activities (Cavallone, Botta, et al., 2021).

It was developed for applications in orchards, olive groves and vineyards, that are agricultural productions largely exploited on the Italian territory. The characteristics of the terrain, such as steep hills and the width of the spacing of the tree lines and vine lines, arise the need for very specific design requirements, such as the ability to climb terrains with an incline of at least 25%, width under 1.5 m, top speed limited to 5 km/h, and the ability of being remotely controlled or autonomous, among other specifications that aim to allow a precise work and avoid damaging the terrain and the surroundings (Botta, 2022.).

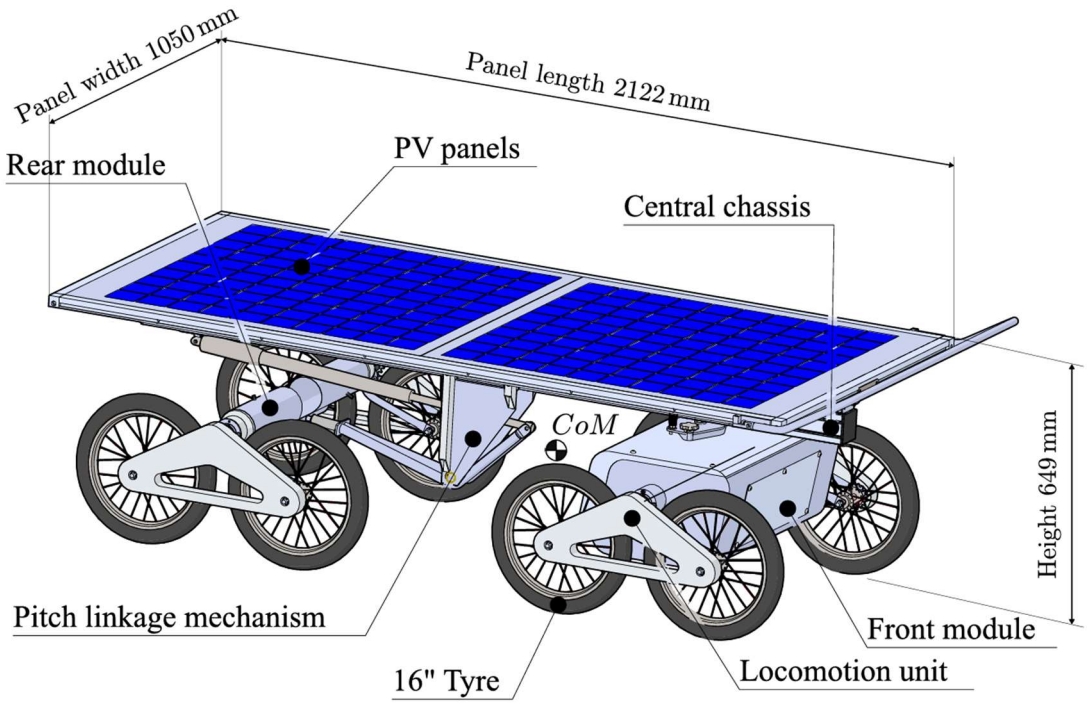

Figure 2.1: Dimensions and main parts of Agri.Q (Botta, 2022)

To ensure stability and optimal performance in an agricultural environment, the rover was designed with two front driving units and two rear driving units of two wheels each, for a total of eight off-road wheels. Each driving unit, equipped with a single drive motor, consists of a rocker arm connected to the rest of the robot body with a passive joint, granting a correct distribution of the static and dynamic forces on the surface. It is interesting to notice that this locomotion system was chosen as a convenient midpoint between a tracked system and a wheel system, for tracks are usually the preferred choice for this kind of rovers thanks to how they distribute the weight of the vehicle on the ground, but are extremely inefficient (Cavallone, Visconte, et al., 2021).

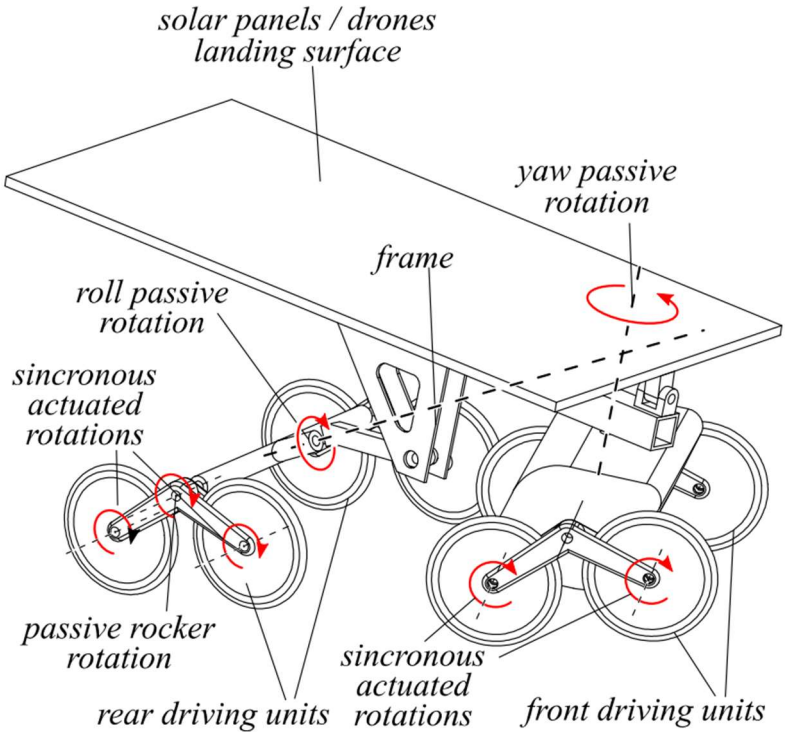

Figure 2.2: Agri.Q functional design (Cavallone, Botta, et al., 2021)

As observed in Figure 2.2, the frame of Agri.Q is attached to two solar panels that serve a double function: charge the battery of the rover and act as a landing platform for drones. This platform is connected to the frame by an active roll joint, thus making the back of the chassis orientable on the vertical axis, allowing to maximise sunrays collection and offer a horizontal surface for drones even on steep slopes(Cavallone, Visconte, et al., 2021).

Originally, Agri.Q was expected to perform manipulation tasks, such as grabbing objects and interacting with the environment or a drone on the platform. For this scope, a robotic arm was attached to the rear of the chassis, from where it could be lifted, thanks to the orientable platform, and its workspace could be increased(Cavallone, Visconte, et al., 2021).

However, for this project, the robotic arm was removed freeing the attachment point and allowing for greater flexibility in the system's configuration. Instead of the robotic arm, a sensor support platform was attached to the back chassis of Agri.Q, from where, analogously to how the workspace of the robotic arm could be increased, the field of view of the sensors could be increased too. The support platform was connected to the chassis using a hinge joint, allowing to keep the platform parallel to the ground.

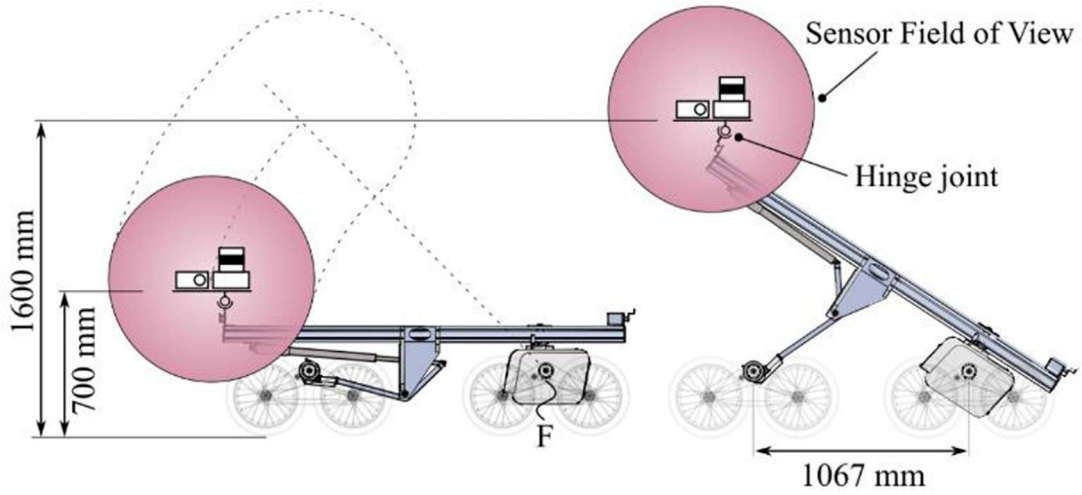

Figure 2.3: Inclination of the chassis of Agri.Q (Smith et al., 2023)

It is interesting to comment on the power efficiency of the rover, and thus its autonomy. The robot was tested while performing its typical activities, observing that in a sunny day it could sustain itself thanks to the contribution of the solar power collected by the panels. It was noted that, despite the fact that during movement the solar power is not enough to recharge the battery, it reduces its discharge. Thus, a correct planning of the rover activity to maximise its recharging period during peak solar irradiance, or the control of the panels orientation to follow the Sun during movement, would highly increase its efficiency (Botta & Cavallone, 2022).

Taking into consideration the aforementioned characteristics of Agri.Q, it was clear that it was the ideal rover to choose for this project:

- Its reduced dimensions allowed easy access and navigation through the apple orchard
- The locomotion system ensured optimal performance regardless of the conditions of the ground
- The possibility of controlling the speed of the rover, plus the stability of the locomotion system, could guarantee a correct data acquisition, avoiding the collection of blurry or shaken images
- Its high efficiency and autonomy would allow continuous work without needing to stop to wait for the battery to recharge
- The ability to lift the support panel provided greater flexibility in capturing the trees from different angles, allowing for a more adaptable data acquisition process

2.2 Optical sensors

Four optical sensors were mounted on the support platform of the UGV, one ZED 2 stereo camera developed by Stereolabs, and three MAPIR Survey3W multispectral cameras.

The ZED 2 camera was designed for the development of depth perception, object detection and 3D mapping applications by using a stereo vision system with two high-resolution RGB cameras spaced 12 cm apart, allowing it to compute depth information using disparity mapping. Additionally, it counts with an integrated Inertial Measurement Unit (IMU), useful for motion tracking. It was designed for easy integration with artificial intelligence algorithms through its dedicated Python API. The output of the camera corresponds to left and right images collected into an SVO (Stereolabs Video Output) file with a resolution of 1920 x 1080 at a rate of 30 frames per second (fps).

Figure 2.4: Stereolabs ZED 2 stereo camera

The ZED camera needs a power supply and a processing platform. For this scope, a NVIDIA Jetson Nano, a compact, power-efficient computer designed for AI computations, robotics and deep learning applications, was chosen. Since it can be powered by a medium-sized power bank, and thanks to its reduced dimensions, it is perfect for on-terrain applications. Before data acquisition, it was loaded with the dedicated ZED SDK and its corresponding Python API, from where the real-time streaming of the camera could be watched, and the recording could be controlled.

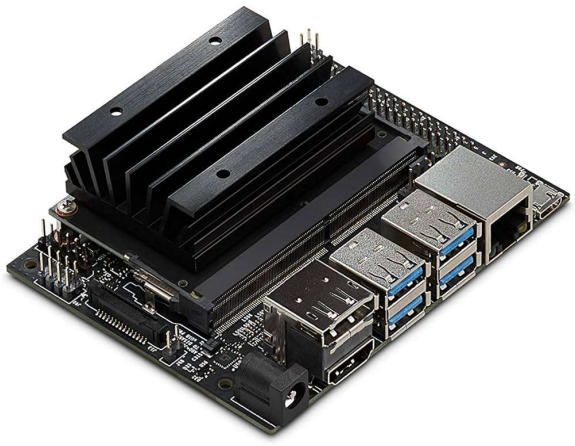

Figure 2.5: NVIDIA Jetson Nano

To protect the Jetson Nano from possible adverse weather conditions during data acquisition, the board was fixed into a custom 3D-printed case that was then fixed to the sensor support platform. In addition, a portable monitor and wireless mouse and keyboard were connected to the computer during data acquisition. It is worth noting that during the first tests in laboratory with the Jetson Nano connected to ZED camera and other peripherals, its heat sink reached very high temperatures, therefore a small fan was connected to the dedicated fan header.

Figure 2.6: NVIDIA Jetson Nano with fan and (open) custom case

The MAPIR Survey3W ("W" stands for "wide", for it has a wider field of view (FOV) than other models) is a compact low-cost multispectral camera designed for aerial and ground-based remote sensing applications. It captures images in the red (R), green (G) and near-infrared (NIR) bands with settable frequency and it has a FOV of 87°. The corresponding bandwidths and wavelengths are presented in Table 2.1. It is widely used in agriculture research projects due to its ability to capture high-quality multispectral imagery for vegetation analysis and terrain mapping, among other data-driven studies. In particular, it allows assessing the health, vigour and chlorophyll content of a plant or fruit by analysing the reflected green light. The dimensions of the MAPIR camera are presented in Figure 2.7.

Figure 2.7: MAPIR Survey 3W multispectral camera

Band	Bandwidth (nm)	Central wavelength (nm)
Red (R)	630 - 700	~670
Green (G)	500 - 580	~550
Near-infrared (NIR)	800 - 910	~850

Table 2.1: MAPIR Survey 3W RGN filter transmission

To complement the data acquired by the MAPIR cameras, each one was connected to a Survey3 Advanced GNSS Receiver, produced by the same company that produces the cameras. In particular, the receiver is based on a U-blox NEO-M8. This advanced, low-cost, GNSS receiver enhances image geolocation accuracy compared to the standard GPS, significantly improving latitude, longitude, and altitude precision. It counts with high sensitivity and fast acquisition times while maintaining low power consumption. Additionally, its sophisticated radio frequency architecture and interference suppression technology ensure optimal performance even in challenging GNSS environments. The integration of this module, and therefore the improved georeferencing, allows the Image Matching algorithm to perform better, leading to a more accurate and well-aligned point cloud reconstruction.

Figure 2.8: MAPIR Survey 3 Advanced GNSS receiver

In addition to the multispectral cameras and the stereo camera, a terrestrial laser scan (TLS) Leica RTC360 was used to obtain a 3D point of the orchard. This compact and portable TLS is a high-precision 3D mapping system designed for rapid and accurate point cloud acquisition, being able to capture up to 2 million points per second with a range of up to 130 meters and an accuracy of less than 2 mm at 20 meters. Thanks to a 360° horizontal × 300° vertical field of view, it provides comprehensive scene coverage. It counts with an integrated Visual Inertial System (VIS) and IMU that enhance registration efficiency, while HDR imaging enables colorized point clouds. Compact and portable, the RTC360 is widely used in geospatial mapping, construction, and agricultural studies for detailed environmental modelling.

Figure 2.9: Leica RTC360 TLS

For high precision positioning measures, an Emlid Reach RS2 was to be mounted on the sensor support platform of Agri.Q. This dual-frequency GNSS geodetic receiver can achieve centimetre-level accuracy using Real-Time Kinematic (RTK) corrections. Its high-precision positioning data was to be integrated with the ZED 2 stereo camera output data, enhancing the accuracy of sensor localization and improving the georeferencing of captured images for more precise point cloud reconstruction.

Figure 2.10: Emlid Reach 2 GNSS dual-frequency geodetic receiver

The complete list of used sensors and their functions is presented in Table 2.2.

Sensor Name	Sensor Type	Data Collected	Information	Coordinate Reference System
ZED 2	Stereo Camera	RGB images Depth images Colourised point cloud of orchard	Left RGB (stereo pairs) Right RGB (stereo pairs) Depth images	Local
MAPIR Survey 3W	Multispectral Camera	Multispectral images (R, G, NIR)	3-bands images: Near Infrared 850 nm, Red 660 nm, and Green 550 nm	Global, EPSG: 32632
Leica RTC360	TLS	Point cloud of orchard	X, Y, Z Intensity values Time stamps RGB values	Global, EPSG: 32632
Emlid Reach 2	GNSS Receiver	HLS position	East, North, elevation above the Ellipsoid Root Mean Squared Errors	Global, EPSG: 32632

Table 2.2: Sensors used for data acquisition

2. 3 Preliminary tests (Smith et al., 2023)

This project builds upon and enhances the work conducted in 2023 by the DIATI and DIMEAS departments of Politecnico di Torino. Accordingly, the setup and methodology were designed to leverage the strengths identified in the initial study while addressing its limitations and gaps.

For this preliminary test, only two optical sensors were used: one ZED 2 stereo camera and one MAPIR Survey3N ("N" stands for "narrow") multispectral camera with a FOV of 47°. Both cameras were oriented parallel to the row of trees, thus perpendicular to the direction of movement of the UGV.

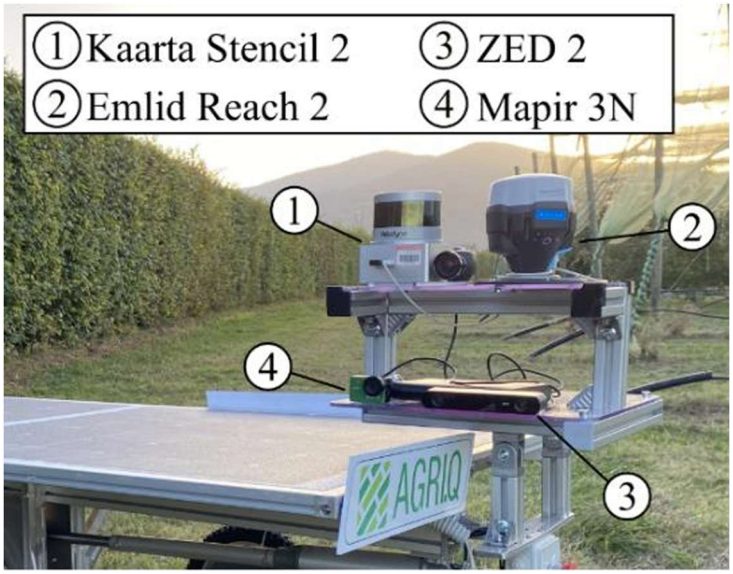

Figure 2.11: Sensor layout on Agri.Q during preliminary test (Smith et al., 2023)

Most of the limitations were due to the orientation of the optical sensors, for they correspond to not sufficient point clouds in which elements had undefined shapes and were difficult to distinguish, especially individual apples where several were growing closely together. These undefined point clouds were obtained from the SVO of the ZED 2 camera, in which images looked almost identical one from the other due to the recurring pattern of trees.

Moreover, the images obtained from the MAPIR camera were not included in the workflow for the images were practically undistinguishable one from the other because of the lack of GNSS localization and the repetitive pattern of trees between pictures, which prevented the Image Matching algorithm from properly working, therefore, a correct point cloud could not be obtained.

The conference paper published by Smith et al. (2023) suggests that different lighting conditions and orientations of the optical sensors could be tested to obtain better results.

2.4 Set-up and design

Following the suggestions of Smith et al. (2023) the position and orientation of the sensors on the support platform were designed to enhance spatial coverage and improve data consistency across the scene by ensuring the right redundancy in the captured images, avoiding as much as possible the repetitiveness of the pattern of trees.

A major change with respect to the work done by Smith et al. (2023) was the addition of two multispectral cameras, with the intention of capturing both rows of trees at the same time, instead of just one. Therefore, the cameras should be placed one next to the other, with the central camera pointing forward and the lateral cameras turned away from the central camera, pointing at the left row and right row, respectively. The position and orientation of the cameras had to satisfy an overlap of the FOVs between 50% and 60%, ensuring enough redundancy of the acquired data for image matching algorithm.

Considering the FOVs as cones, the intersection of the FOVs could be described as the intersection of one circle and two ellipses, where the circle corresponds to the central camera and the ellipses to the lateral cameras. Knowing the geometry of the cameras, their FOV angle, the distance between the tree rows and the average height of the apple trees to be captured, the distance between the cameras and the orientation of the lateral cameras could be calculated to obtain the desired overlap by using the characteristic equations of an ellipse and of a circle. A diagram representing the FOVs, object distance and radius of the FOV of the central camera is presented in Figure 2.12. To begin with the calculations, the following parameters were fixed:

Parameter	Value
FOV angle	87°
Objective distance (*h*)	2,5 m
Distance between cameras	15 cm
Distance of the cameras from the ground	1,1 m
Average tree height	4 m
Orientation of side cameras (with respect to central camera)	30°
Distance between the lenses of the cameras	20,9 cm

Table 2.3: Initial parameters for FOV intersection calculations

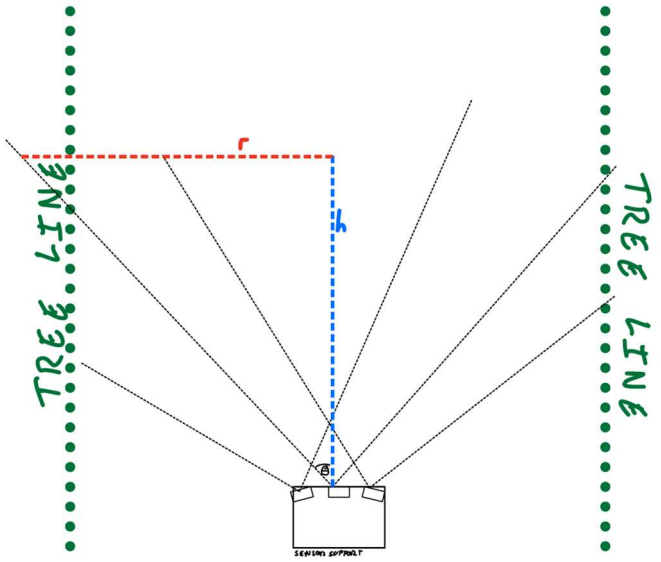

Figure 2.12: Diagram representing the FOVs

The FOV of the central camera can be considered as the base of a cone with height (h) 2,5 m, corresponding to the objective distance, and angle (θ) 43,5°, corresponding to half of the FOV angle. Therefore, the radius (r) of the base of the cone can be calculated using the sine law:

$$\frac{r}{h} = \tan(\theta)$$

$$r = \frac{h \cdot \sin(\theta)}{\sin(90° - \theta)}$$

$$r = \frac{2{,}5 * \sin(43{,}5°)}{\sin(56{,}5°)}$$

$$r = 2{,}37\ m$$

On the other hand, the FOV of the lateral cameras can be calculated in a similar way, but since they are slightly turned away from the central cameras, the projection of their FOVs would correspond to ellipses. The length of the semi-minor axis (b) corresponds to the radius of the centred cone. The length of the major axis was calculated considering the displacement of the centre of the ellipse (c) introduced by the extra 30°, using the sine law. A diagram helpful to better understand the calculations is presented in Figure 2.13

$$\frac{c}{h} = \tan(30° + \theta)$$

$$c = 1{,}44\ m$$

The remainder of the major axis was calculated in a similar way:

$$\frac{d}{h} = \tan(30° + \theta)$$

$$d = 8{,}44\ m$$

$$\frac{a}{h} = \tan(30° - \theta)$$

$$a = 0{,}6\ m$$

Then, the final length of the major axis is:

$$a + d = 9{,}04\ m$$

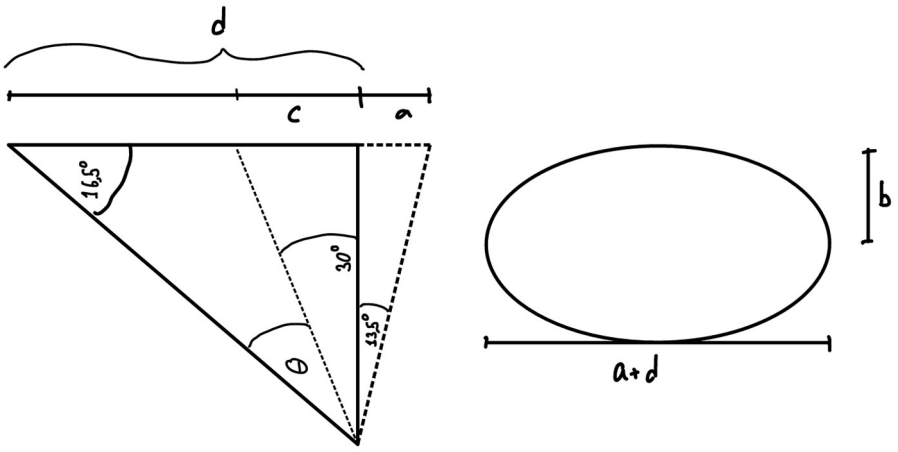

Figure 2.13: Diagram representing the ellipse characteristcs

Finally, the overlap of the FOVs was graphically represented using the graphic calculator Desmos (Figure 2.13). The y axis was centred on the lens of the central camera and the x axis was considered as ground level, therefore, each FOV has an offset of 1,1 m on the y axis, and each side camera has an offset of 0,209 m (distance between lenses) plus 1,44 m (displacement of the centre of the ellipse) on the x axis, on each side of the y axis. The corresponding characteristic equations are:

- Central camera (in red):

$$2{,}37 = x^2 + (y - 1{,}1)^2$$

- Right camera (in blue):

$$\left(\frac{x - 0{,}209 - 1{,}44}{4{,}52}\right)^2 + \left(\frac{y - 1{,}1}{2{,}37}\right)^2 = 1$$

- Left camera (in green):

$$\left(\frac{x + 0{,}209 + 1{,}44}{4{,}52}\right)^2 + \left(\frac{y - 1{,}1}{2{,}37}\right)^2 = 1$$

As observed in Figure 2.13, the cameras can easily cover the whole width of the space between the tree rows (4 m), and the trees. Moreover, the overlap of the FOVs was calculated using MATLAB, and the results, presented in Table 2.4, were deemed satisfactory (overlap between 50% and 60%). The complete script is presented in Annex 1.

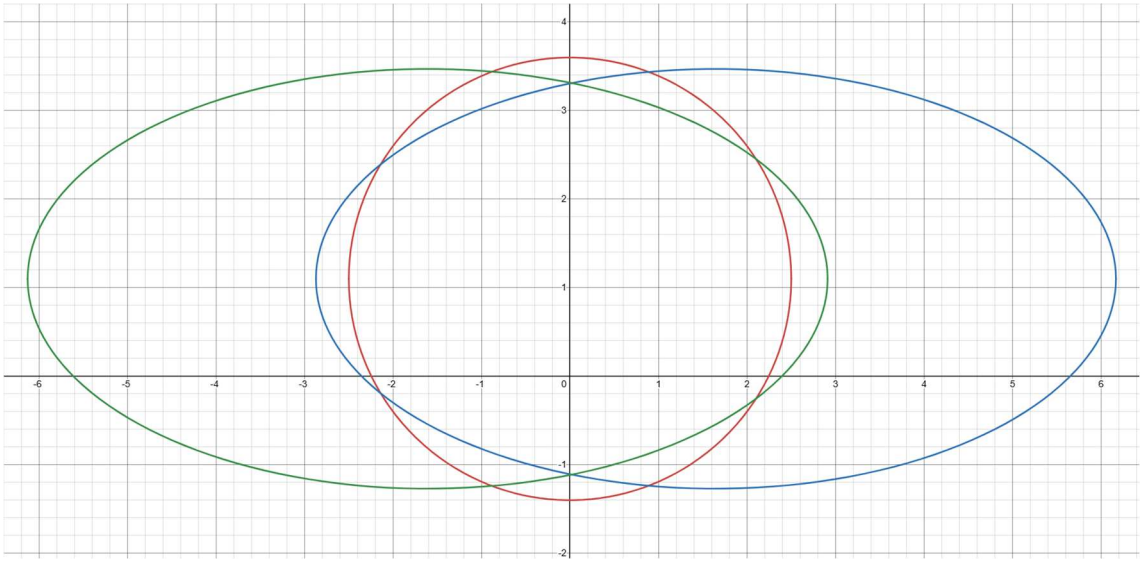

Figure 2.14: Graphical representation of the overlap of the FOVs of multispectral cameras

Measurement	Result
FOV of central camera	19,64 m^2
FOV of lateral cameras	33,69 m^2
Estimated overlapping area	17,06 m^2
Overlap as percentage of central camera	86,90%
Overlap as percentage of lateral camera	50,95%

Table 2.4: Areas and overlap of the FOVs of multispectral cameras

On the other hand, after analysing the outputs of the ZED 2 camera obtained by Smith et al. (2023), it was observed that a point cloud could better be made if the camera was able to distinguish something other than trees, for example, the end of the tree row. In consequence, it was decided that the camera should be turned 45° away from the tree row, capturing one end of it during the whole data acquisition process, while capturing as many apples as possible.

2.5 Field tests

Once the relative position of optics was mathematically identified, a first prototype was built and three field tests were conducted to evaluate the goodness of the orientation, and other relevant parameters, such as images acquisition frequency rate, ISO and exposure.

To evaluate the distribution of the MAPIR cameras, there was no need to use Agri.Q, therefore, the prototype consisted of an easy-to-carry sensor support platform. Hence, the three multispectral cameras and their corresponding GNSS receivers were mounted on a piece of cardboard, as observed in Figure 2.16.

The first two tests were conducted trying to simulate the characteristics of an orchard, that is, narrow and long spaces with repetitive textures. On the other hand, the third test was conducted on an actual apple orchard, located close to the orchard where data was to be acquired.

For the first test (Test N°1), it was decided to not include the GNSS receivers, for it was conducted in a corridor inside of Politecnico, where the GNSS signal was expected to be bad. The pictures taken by the multispectral cameras confirmed that the distance between them and their corresponding orientations was sufficient to cover the desired FOV in a narrow space. On the other hand, the creation of a point cloud from the picture was not possible for two reasons:

- The alignment was unsatisfactory due to the lack of GNSS data
- The creation of a 3D model was faulty because of the light reflected by the windows in the corridor

Figure 2.15: Test N°1

The second test (Test N°2) was conducted in a courtyard of Politecnico, between lightwells with plants between them. This test confirmed the configuration of the multispectral cameras for outdoor conditions, and the GPS receivers ensured a correct alignment of the pictures. Nonetheless, the light reflected by the glass of the lightwells and the windows on the buildings surrounding the courtyard did not allow the creation of a point cloud.

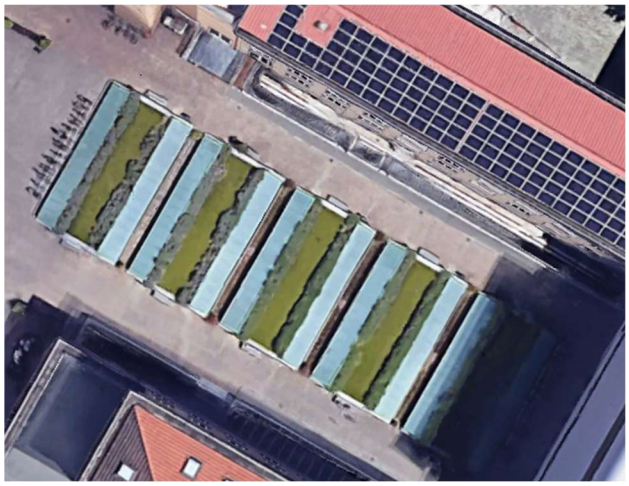

Figure 2.16: Politecnico courtyard

Finally, the third test (Test N°3) confirmed once more the configuration of the cameras. It was, also, useful to estimate the speed with which Agri.Q should move during data acquisition, since the quality of the point cloud was not sufficient due to the blurriness caused by the high speed and vibration with which the pictures were taken. The main characteristics of tests are summarised in Table 2.5.

Figure 2.17: Prototype during field Test N°3

Test N°	Capture frequency rate	ISO	Exposure	GNSS receiver	Image format	Total number of pictures	Satisfactory point cloud
1	1 sec	Auto	0,0	No	RAW	145	No
2	1 sec	Auto	0,0	Yes	RAW	391	No
3	1 sec	Auto	0,0	Yes	RAW	724	Yes

Table 2.5: Summary of field tests

2.6 Sensor support platform

The support platform used by Smith et al. (2023) had to be modified to meet the new requirements by drilling additional holes to fix the sensors in the desired position. The platform consisted of two plexiglass plates of different sizes, with the smaller plate mounted above the larger one using aluminium profiles.

Before drilling new holes in the plexiglass, the various optical sensors were presented on the support platform to control their FOVs, ensuring they did not capture the platform itself.

Specifically, the three multispectral cameras were fixed on the border of the bottom level of the platform, one at the centre of the plate, and the others as close as possible to the lateral borders. The GPS receivers connected to the MAPIR cameras were positioned on the bottom level, ensuring they remained unobstructed by the top level to maintain proper GPS reception.

Moreover, the ZED 2 camera was fixed on a corner of the top plate, as close as possible to the right side of the platform.

It is interesting to notice that the optical sensors were placed on top of a printed protractor template, centred on the screw that allowed for their rotation. Additionally, custom 3D-printed cases were designed for each camera, incorporating a needle to precisely indicate and adjust their orientation.

On the other hand, the MAPIR GPS receivers, the case of the Jetson Nano and the power bank were secured to the bottom plate using Velcro. The Emlid Reach 2 was not available at the time, but it was supposed to be mounted on the top plate, in order to maintain a stable GNSS reception.

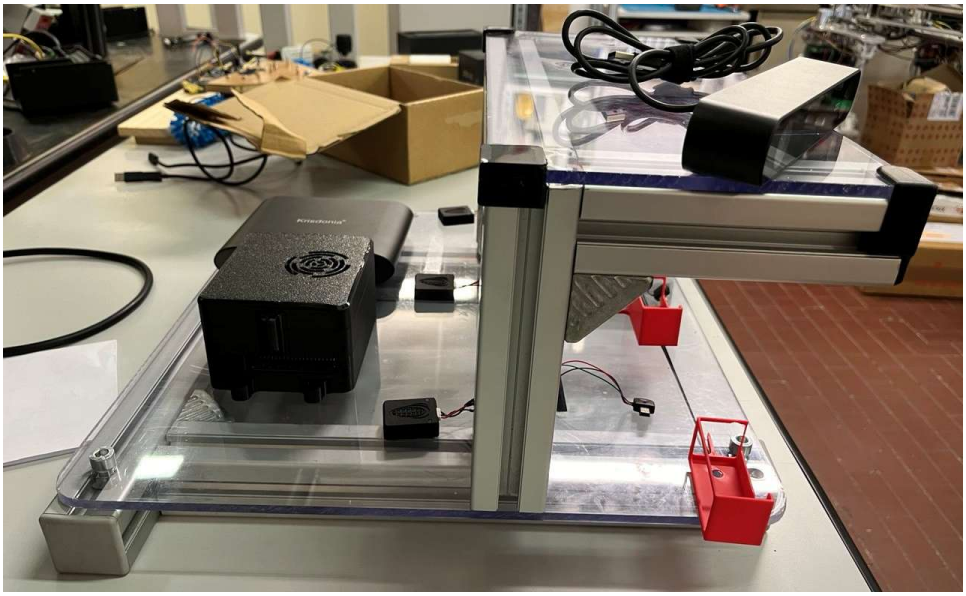

Figure 2.18: Lateral view of the sensor support platform while presenting components in their possible positions

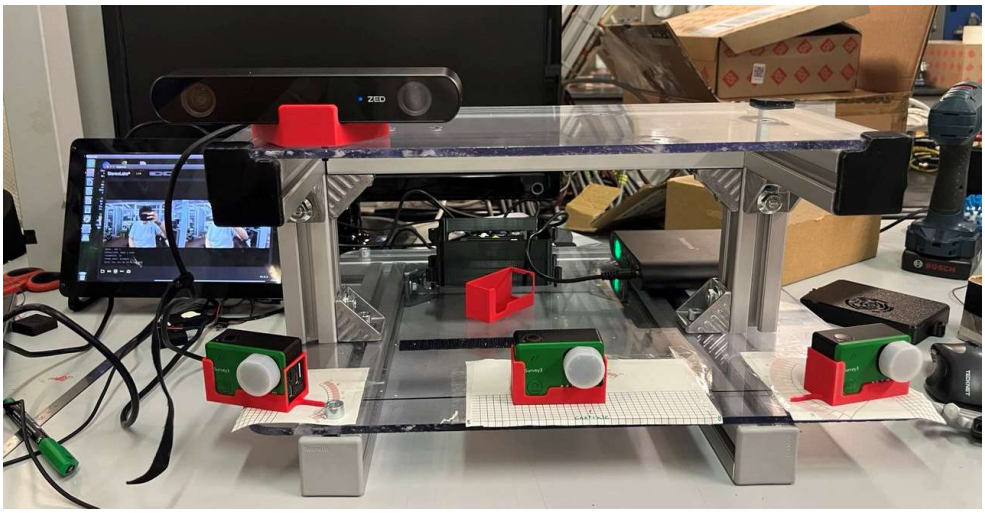

Figure 2.19: Frontal view of the sensor support platform with the optical sensors fixed to their definite positions

Chapter 3: Methods

This chapter outlines the methodology used in the data acquisition process and the generation of point clouds from the optical sensors. It then describes the apple detection approach using the stereo camera and YOLOv8. The workflow for processing and analysing the collected data is explained, including techniques for extracting relevant features from point clouds. Additionally, the methods used to evaluate the obtained results are presented.

3.1 Data acquisition

Data acquisition was performed in a commercial apple orchard in Manta di Saluzzo, Piedmont, Italy, (44°36'30"N 7°31'17"E) on October 30th, 2024. The apple trees were 4 meters tall, on average; the tree rows were 255 meters long and were distanced by approximately 4 meters. The tree lines were orientated from North to South, with a slight deviation towards the East (164,25°). The weather and ground conditions were optimal, with clear skies and firm terrain that was not excessively muddy, therefore, Agri.Q was expected to perform excellently.

As observed in Figure 3.2, a hail net covered the orchard. This net was unaccounted for, and its presence could disturb the GNSS reception and, more importantly, could present a difficulty when creating a clean point cloud of the orchard.

Figure 3.1: Location of the orchard in Piedmont, Italy

Figure 3.2: Agri.Q in the orchard

Before data acquisition, some of the support poles of the orchard were tagged with unique georeferenced QR codes to be used as visual cues during post processing.

To accurately reconstruct the orchard geometry and achieve high precision georeferencing, TLS and GNSS surveys were carried out. The objective was to use the obtained dataset as a reference for validating the acquired data and as an initial baseline survey for mapping the case study. 11 scans with Leica RTC360 were realised in the field with low density due to the closeness with the tree rows. Thanks to the IMU and visual odometry system of the TLS, the clouds were automatically pre-aligned and then refined manually using common points (markers) and the coordinates of the poles.

Using the GNSS receiver in RTK modality, the coordinates of the bases of the poles were measured in global reference system wgs84/32N. The quality of the measurements varies according to the GNSS signal and the number of satellites visible. Indeed, due to the geometry of the orchard, a small portion of the sky was visible, preventing the fixing of the ambiguity (necessary to obtain 3-5 cm precision) in some parts. Nevertheless, the measured and fixed points were sufficient for georeferencing the TLS cloud through a roto-translation based on 5 points.

To better understand the data acquisition procedure, it is first essential to consider the movement direction of Agri.Q and how this influenced the way the sensors captured data. Due to the design of Agri.Q and the position of the sensor support platform, as the rover moved forward, the sensors, mounted at the rear, gradually moved away from the starting point, continuously capturing data from an increasing distance.

A preliminary test was conducted to confirm the sensor layout, the camera parameters, the speed of the rover and the motion path. The rover was driven through two tree lines in only one direction. The real-time streaming of the ZED camera was continuously

controlled on the display connected to the Jetson Nano, while the pictures taken by the MAPIR cameras had to be downloaded into a laptop to be better controlled.

Moreover, it was observed that the orientation of the stereo camera allowed for a complete capture of only the tree row on the right. As a result, the original path was deemed insufficient for generating a complete point cloud. To address this, it was decided that the rover would traverse the space between two tree rows twice, following the same trajectory in both senses, as illustrated in Figure 3.3. Additionally, the tree rows were long enough to allow the acquisition of sufficient data of just two rows.

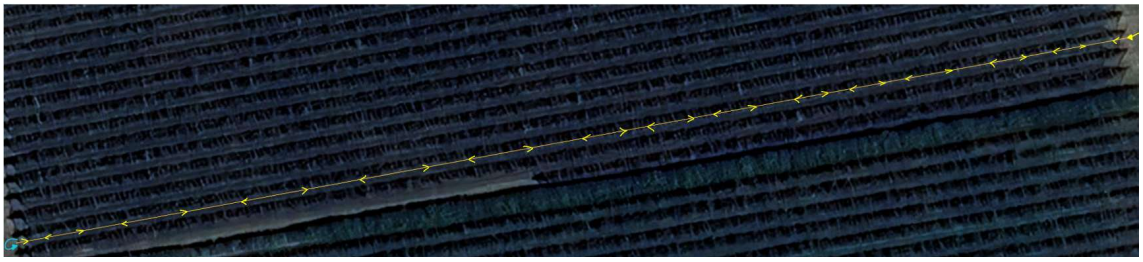

Figure 3.3: Definitive path

While preparing for the definitive run, a technical error occurred with one of the MAPIR cameras, nonetheless, after analysing again the pictures obtained during the preliminary test, it was decided that the coverage of just the central camera and the right camera would be enough to obtain a good point cloud. After analysing the output of the various sensors, the following parameters were set:

MAPIR Parameter	Value
Capture Frequency rate	0,5 s
ISO	Auto
Exposure	0

Table 3.1: MAPIR parameters

ZED 2 Parameter	Value
Brightness	4 (out of 8)
Contrast	4 (out of 8)
Hue	0
Saturation	4 (out of 8)
Sharpness	4 (out of 8)
Gamma	5 (out of 9)
White balance	auto adjust
Exposure	auto adjust

Table 3.2: ZED 2 parameters

Once the sensors were configured and working, two runs were performed. The rover was driven using a remote control, keeping a constant speed of 0,41 m/s and staying in the middle of the space between the two tree lines. The first run was made at 15:20 and consisted of a one-way trip of the tree line to obtain only images from the MAPIR cameras; 449 pictures were obtained. The second run began at 16:00 and consisted of a forward and return pass along the tree line; at the end of the tree line the rover was turned around and the correct position and orientation of the sensors were controlled. The streaming of the ZED camera was controlled on the monitor, while the function of the MAPIR cameras was controlled by listening to the beep they emitted when they took a picture. Data acquisition of the sensors was completed only when the rover had fully stopped back at the starting point. The results of the run were:

- Duration of data acquisition: 21 minutes
- Pictures captured by MAPIR cameras: 624
- Frames captured by ZED camera: 19899

Figure 3.4: Image from MAPIR multispectral camera

3.2 MAPIR point cloud and 3D model creation

The outputs of the multispectral cameras from the last two data acquisition runs were compared, and the images captured during the first run were selected over those from the second due to more favourable sunlight conditions. Additionally, although the second run included a round trip along the tree line, the images from the one-way pass were considered sufficient to generate a satisfactory point cloud of the trees.

The images taken by the two MAPIR cameras were added to an Agisoft Metashape project. Some of the first and last images that did not include apple trees were removed from the project, thus only 431 images were used.

Camera calibration on Metashape was performed according to the settings provided by the MAPIR website for the Survey3W model (Ramseyer, 2016), setting the following parameters:

- Camera type: frame
- Pixel size (mm): 0,00155 x 0,00155
- Focal length (mm) 3,37

Moreover, the site indicated that all camera parameters should be set to “fixed”, and that “Generic pre-selection” should be turned off during the Align photos step.

Alignment was performed with the following settings:

- Accuracy: high
- Generic preselection: No
- Key point limit: 40,000
- Tie point limit: 4,000

Moreover, a raster transform was applied to calculate the normalised difference vegetation index (NDVI), using its formula:

$$NDVI = \frac{NIR - RED}{NIR + RED}$$

This process was repeated on a small section of the orchard, with an additional step. Before generating the point cloud, a colour-based mask was applied to 112 images to minimise the presence of the hail net and the sky.

3.3 ZED point cloud creation and alignment with GNSS point cloud

Extraction of the point cloud from the SVO was performed on the Jetson Nano using the ZEDfu tool of the ZED SDK, as a fused point cloud. Due to memory limitations of the Jetson, extraction was not possible on a single instance, instead, it had to be performed continuously stopping and restarting the extraction to avoid crashes, obtaining different separate pieces of the point cloud.

Using CloudCompare, the point clouds generated by the ZED stereo camera were aligned and merged by referencing the point cloud obtained from the TLS, which served as a template. In other words, the TLS point cloud acted as a fixed reference onto which the different segments acquired with the ZED camera were manually overlaid and aligned. To simplify the process and improve accuracy, the two tree rows (left and right) were treated separately during the alignment procedure. As for the point cloud obtained from the multispectral cameras, the hail net had to be manually deleted from these point clouds, too. Overlapping and alignment were performed by choosing at least 4 equivalent point pairs on the TLS point cloud and on the ZED point cloud. Most of the visual cues chosen for alignment were support poles, particularly those tagged with a QR code. Once point pairs are selected, CloudCompare computes the rigid transformation matrix that best aligns the selected points using a least squares method.

3.4 Apple detection

As already mentioned, the ZED 2 stereo camera can be easily integrated with the YOLO algorithm. The algorithm can be run online in real time or afterwards using the SVO file given as output by the ZED camera. For this project, apple recognition was performed on the SVO file after data acquisition. Due to the extensive length of the tree lines, the SVO file was divided into several fragments of the orchard, from which the last fragment was selected for testing (7696 frames).

The YOLOv8 segmentation model used had already been trained by Smith et al. (2023) using images from the MinneApple data set (Hani et al., 2020) and RGB images previously exported by them from the stereo camera when performing the preliminary test. After trying the model on the fragment of tree line selected for testing, it was decided to further train the model with 33 RGB images obtained from fragments different from the test fragment, in order to minimise dataset bias. These images were sliced into smaller images to satisfy the YOLOv8 input size (640 x 640), from which only 27 could be labelled using V7 Darwin (*V7 | AI Document Processing & Data Labelling*, 2025), and then split into training (17), validation (6), and testing (4) datasets.

To run the detection model on the ZED camera, the object detection example Python script provided by the ZED Python SDK was used with some minor modifications, such as the display of the ID of the detected object, its coordinates and its dimension, plus saving them on a text file.

In addition to running detection on the SVO, inference was performed using YOLOv8, and YOLOv8 integrated with SAHI, on a single still frame.

3.5 Apple georeferencing

Before describing the georeferencing process, it is helpful to define two concepts relative to the functioning of the ZED 2 stereo camera (*Coordinate Frames - Stereolabs*, 2025):

1. *Camera Frame:* located at the back of the left lens of the device.

2. *World Frame:* describes the position of the camera with respect to a stationary point. By default, it corresponds to the point where the stereo camera began motion tracking and is oriented in the direction where it was looking at.

To further improve the work done by Smith et al. (2023), it was decided to also add georeferencing to the apple detection. Three options were proposed for this purpose:

1. GNSS integration: the output data of the ZED 2 camera can be integrated with a GNSS receiver that has followed the same trajectory of the stereo camera by combining visual odometry and GNSS position. This makes it easy to obtain the position of the camera, and thus the position of the detected objects, in a global reference frame.

2. Manually set the starting point coordinates: it is possible to set an initial world transform as the starting position and orientation of the Camera Frame in world space. From this point onwards, the position of the camera, and thus the position of the detected objects, can be obtained with respect to the starting point.

3. QR with known coordinates: following the same principle as in Option 2, the initial world transform could correspond to the known coordinates of a detected object that remains static, for example, a QR code attached to a post.

Option 1 was not possible due to bad satellite distribution during data acquisition and the presence of the hail net over the orchard that interfered with the GSM signal reception. While Options 2 and 3 operate in a similar manner, QR code recognition was set aside for future enhancements, leading to the selection of Option 2.

The same Python script used for apple detection was further modified for apple georeferencing. More precisely:

- Change of coordinate system: The ZED camera uses as an initial parameter, by default, a right-handed, y-down coordinate system. The coordinate system was changed to left-handed, z-up, in order to have x in the direction of the movement of the rover (the rover moves away from the origin), z as height, and y for the distance from the camera and the tree line. Different coordinate systems supported be the ZED 2 are shown in Figure 3.19.

- World reference: the ZED run time parameters set as default the Camera Frame as reference, therefore, to obtain the position of the detected objects with respect to the starting point, the reference was set to World reference. Then, when retrieving the camera position, it must be specified that it is with respect to the World Reference Frame.

- Positional tracking parameters: the UTM coordinates of the starting point must be set as the origin of the world frame when setting the positional tracking parameters. This is done by creating a translation vector with the UTM coordinates, assigning it to a transform object, and then setting this transform as the initial world transform in the positional tracking parameters. However, since the GNSS data was not satisfactory, the vector was set to a generic origin (0,0,0), creating a reference frame from the starting position of the ZED camera.

- Create LAS point cloud: using the *laspy* Python library, a LAS point cloud containing the positions and the dimensions of the detected apples was created. It was noted that when printing the positions of the apples, some of them slightly changed when detected in different frames. In fact, when generating the first point clouds, a dragging effect was observed, for the same apple was represented several times. To solve this, the position of the apple was saved only the first time its ID was detected. The resulting point cloud is presented in Figure 3.20 and Figure 3.21.

The complete Python script us presented in Annex 2.

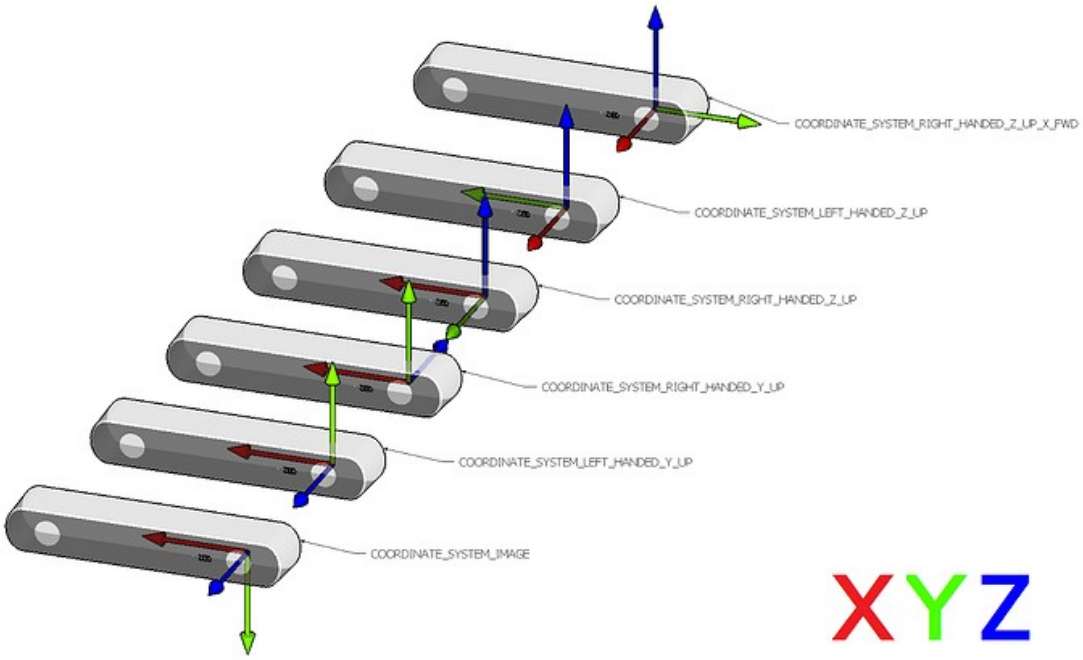

Figure 3.5: ZED 2 possible coordinate frames (*Coordinate Frames - Stereolabs*, n.d.)

Chapter 4: Results

This chapter presents the outcomes of the methods applied in the study, including the performance of the detection model, the characteristics of the generated 3D models and point clouds, and the alignment process.

4.1 MAPIR point cloud and 3D model creation

After uploading the images captured by the multispectral cameras on Metashape, the 431 obtained images were correctly aligned, and 311,548 tie points were found. From these tie points, the point cloud was created and manually cleaned to delete as many points as possible that corresponded to the hail net and points far from the tree lines, obtaining 2,689,341,944 points. The 3D model was built using these points, obtaining 17,510 faces. A fragment of the obtained model is presented in Figure 4.1.

Figure 4.1: 3D model obtained from MAPIR images

The 3D model obtained after applying the NDVI formula presented in section 3.2 is presented in Figure 4.2:

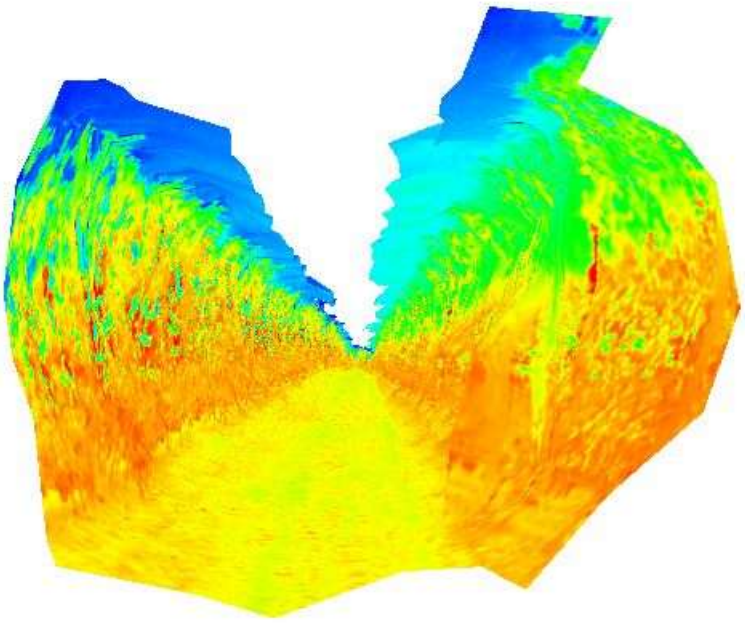

Figure 4.2: 3D model obtained from MAPIR images NDVI

The colour-based mask applied to a smaller fragment of the orchard is presented in Figure 4.3. After applying the mask 82,652 tie points were obtained, the point cloud consisted of 2,541,037 points and the 3D model had 4,146,159 faces. Fragments of the 3D model are presented in Figure 4.4 and Figure 4.5.

Figure 4.3: Mask applied to eliminate hail net and sky

Figure 4.4 3D model obtained after masking by colour

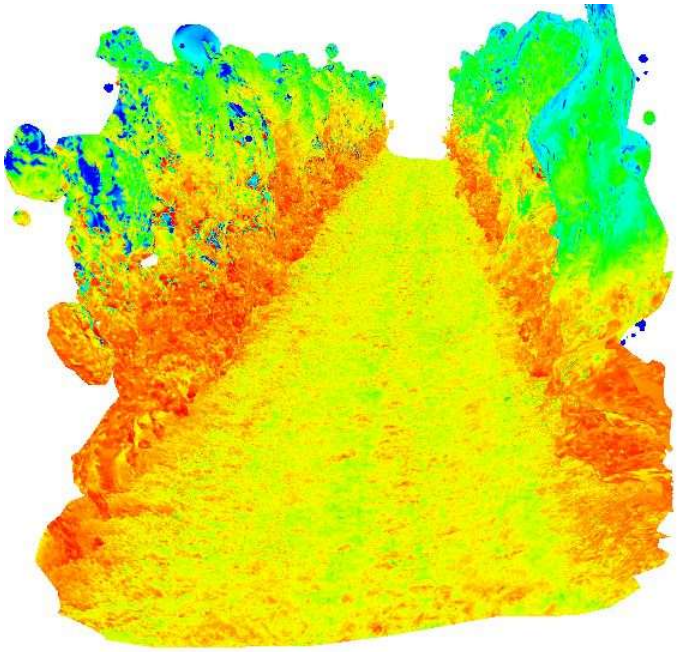

Figure 4.5: 3D model after masking by colour NDVI

4.2 ZED point cloud creation and alignment with GNSS point cloud

As discussed in section 3.3, the point clouds obtained from the ZED data and the point cloud obtained from the TLS were aligned using CloudCompare. The final RMSE on the 11th cloud obtained from the TLS dataset, was of 10 cm. The final point cloud was filtered and cleaned of the hail nets, setting a threshold on the radiometric response of points. Also, it was then subsampled to lighten the file size (final size 1,01 GB). The average Root Mean Square Error (RMSE) obtained from the alignment of the different segments of the point cloud was of 20 cm. Fragments of the various point clouds are presented below.

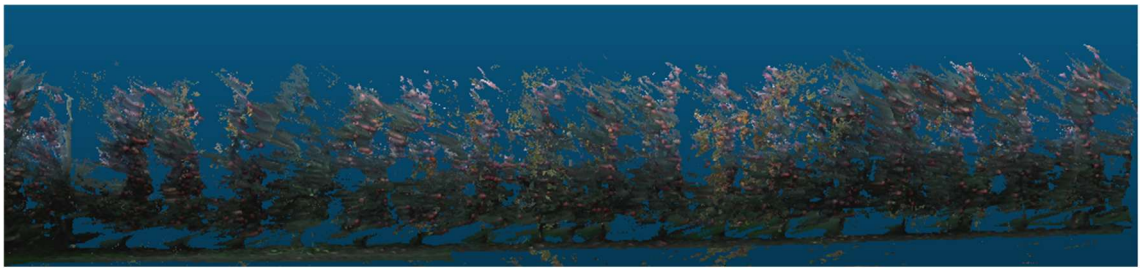

Figure 4.6: Fragment of final point cloud obtained from the ZED point cloud and the TLS point cloud

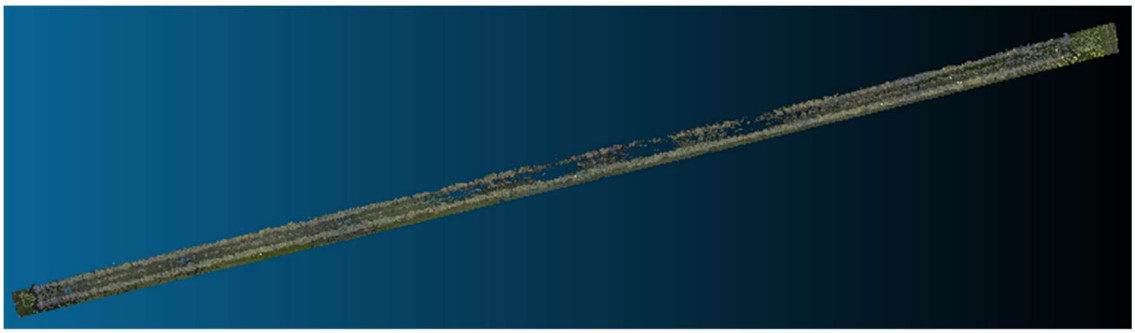

Figure 4.7: Top view of the TLS point cloud

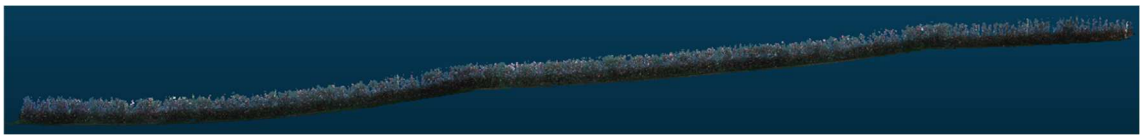

Figure 4.8: Lateral view of the ZED point cloud

Figure 4.9: Top view of the ZED point cloud

Figure 4.10: Fragment of the ZED point cloud

4.3 Apple detection

The performance metrics on training and validation of the YOLOv8 model are presented in Table 4.1.

Metric	Value
Recall	0,918
Precision	0,884
F1-score	0,900

Table 4.1: Training and validation performance metrics

The outputs obtained from detection on SVO, inference using YOLOv8 and inference using YOLOv8 plus SAHI are presented in Figure 4.11, Figure 4.12 and Figure 4.13, respectively. It should be noted that the same frame was used to run inference with and without SAHI. The printed output of the apple size and positions in the local reference frame is presented in Figure 4.14.

Figure 4.11: Apple detection model run on SVO

Figure 4.12: Inference using YOLOv8

Figure 4.13: Inference using YOLOv8 integrated with SAHI

```
0: 256x416 15 apples, 131.7ms
Speed: 2.0ms preprocess, 131.7ms inference, 6.0ms postprocess per image at shape (1, 3, 256, 416)
Results saved to runs\segment\predict84
1 [   0.079246    0.066315    0.079247] [    0.97372      1.3768     0.75291]
8 [   0.081824    0.057982    0.081824] [     1.1051      1.1968     0.13339]
7 [   0.071763    0.071918    0.071763] [     1.0527      1.1143     0.71592]
4 [   0.085049    0.059586    0.085049] [     1.0882      1.2768     0.34349]
2 [   0.089362    0.078216    0.089362] [     1.0966      1.2356     0.16265]
6 [   0.094795    0.070024    0.094795] [      1.133      1.2412    -0.10721]
0 [   0.082465    0.051914    0.082465] [     1.0879      1.3272     0.64566]
5 [   0.087632    0.071383    0.087632] [     1.1304      1.3553     0.30906]
9 [   0.091732    0.080995    0.091732] [     1.1683      1.0046    0.076839]
3 [   0.083413    0.071273    0.083413] [     1.1502      1.1046     0.55268]
camera positon: <built-in method get_position of pyzed.sl.Camera object at 0x000001B7E172ABB0>
```

Figure 4.14: Output of apple dimensions and positions in local frame

4.4 Apple georeferencing

After performing apple recognition and georeferencing, a point cloud representing the positions of the apples was obtained. The point cloud contains 1,162 points and is presented in Figure 4.15 and Figure 4.16

Figure 4.15: Top view of the point cloud containing the positions of the detected apples

Figure 4.16: Fragment of the point cloud containing the positions of the detected apples

Chapter 5: Discussion

This chapter presents and assesses the results presented in the previous chapter. Moreover, the limitations observed during data acquisition and post processing will be discussed, along with proposed improvements and a brief consideration of potential directions for further research.

The quality and reliability of the data collected were influenced by several practical limitations encountered during data acquisition. These constraints, related mostly to environmental characteristics, had a direct impact on the performance of the sensors and the accuracy of the resulting models.

Due to the orchard's North–South orientation with a slight tilt towards the East, sunlight exposure during data acquisition was uneven across the tree rows, leading to inconsistent lighting conditions that affected image quality and, consequently, model rendering and detection performance. The effect of lighting conditions can be observed in the images captured by the stereo camera, in which the colour of the apples and the leaves are not bright and, thus, hardly distinguishable. In fact, apple detection was not satisfactory, for only a very small fraction of the apples were correctly detected, resulting in a large number of false negatives. Although the integration of SAHI improved inference, still a large proportion of apples remained undetected. Moreover, as mentioned in section 1.2.3, SAHI is not suitable for real time detection, thus, it cannot be applied on an SVO, nor the real time streaming of the ZED camera. It is interesting to note that this problem was also encountered by Smith et al. (2023). To mitigate the effects of uneven lighting conditions, Wang et al. (2013) conducted data acquisition during nighttime using ring flashes mounted with the cameras to illuminate the orchard. Nonetheless, this solution could only be applied during nighttime, which is not sufficient if constant real time monitoring is desired.

It is important to note that the unsatisfactory apple detection does not concur with the values of the training and validation metrics presented in section 3.4, which are, indeed, high. This discordance could be due to the difference in quality and conditions between the training dataset and the actual field scenario. In addition to the limitations due to the quality and clearness of the images, apples that were small, partially occluded, or clustered together proved to be more difficult for the model to detect in real-world conditions.

Another limitation introduced by the characteristics of the orchard was the hail net covering the trees, which affected both model construction and apple georeferencing. When creating the 3D model of the orchard from the images captured by the MAPIR cameras, Metashape recognised the hail net as a roof, creating a "tunnel effect". This effect limited the 3D reconstruction of the orchard, in which the trees seem enclosed, and their shape is not clear. The removal of the hail net through masking-by-colour significantly improved 3D reconstruction, in fact, the shape and volume of the canopy and of the fruits are clearly distinguishable. Moreover, the NDVI output appears coherent and consistent with the vegetative structure of the orchard, indicating a correct implementation of the index, especially for the model created after applying the mask. Nonetheless, performing radiometric calibration of the multispectral cameras could improve the results of the NDVI model.

Additionally, the point cloud created from the SVO file presents both tree rows curved, which completely differs from the distribution of the orchard. This anomaly could be due to the IMU in the ZED camera, as it could have had certain difficulties to track a straight line for long distances.

On the other hand, as mentioned in section 3.5, the presence of the hail net disturbed GNSS reception by the GNSS receiver, which drastically limited the apple georeferencing. Also, the geometry of the orchard itself limits the GNSS constellation visibility, leading to conditions similar to the *urban canyon* and increasing the possibility of multipath-induced uncertainties. These factors made the data collected by the GNSS not compatible with the data retrieved by the IMU present in the ZED camera. If both outputs could have been merged, the path followed by the ZED would have been automatically georeferenced and corrected, and thus, the exact positions of the apples could have been obtained in world coordinates. Instead, the positions of the apples were obtained in a local frame with the origin at the starting point of the ZED camera, as discussed in section 3.5. Nonetheless, the obtained point cloud of the positions of the apples can be deemed satisfactory, for their distribution resembles the shape of the point cloud of the orchard, including the curves introduced by the IMU.

However, a possible solution to correct the error introduced by the path registered by the IMU could be to constantly update the position of the ZED camera through the recognition of visual cues with known exact positions, such as georeferenced QR markers. This solution would also require the exact starting point of the camera. In such a way, the drift introduced by the IMU would be reduced, and the position of the detected objects could be obtained with respect to the most recent position of the camera in the world frame.

While the images captured by the MAPIR and ZED cameras were suitable for analysis, their clarity was compromised by blur caused by the movement of the rover during acquisition. Even if this was not an impediment to the creation of the 3D model and the apple detection, slightly reducing the speed of Agri.Q during data acquisition could increase the quality and clearness of the final models. Despite of the aforementioned issue, the performance of Agri.Q was optimal.

Moreover, the georeferenced point cloud captured by the TLS was optimal, for it correctly represented the shapes and colours of the trees and apples.

Taking into consideration the results and limitations discussed above, the workflow for creating a DT can be considered functional but incomplete. These missing aspects can be solved by further improving the workflow. Table 4.1 summarises the encountered limitations and the proposed solutions.

Limitation	Proposed solution
Incorrect shape of point cloud created with the stereo camera	Integration of ZED 2 with GNSS receiver that follows the same path, self-repositioning of the stereo camera
Apple detection	Constant artificial lighting during data acquisition, better trained detection model
Apple georeferencing	Integration with GNSS data, self-repositioning of the stereo camera

Table 4.1: Limitations and proposed solutions

Further research on this work could include the integration of information on the trees and fruits, such as the colour intensity of the apples, water status of the orchard and the presence of pests or diseases. These additions could be developed by adding classes to the detection model run on the stereo camera, as well as the contribution of ground data collected by farmers. Greater autonomy of Agri.Q could ensure constant monitoring without the need for an operator to drive it. The introduction of the detection of markers, besides improving georeferencing, could also enhance the autonomy of the rover, aiding it to follow predefined paths. Better integration of the sensors and a more performing onboard processing system could allow for real time updates of the orchard, eliminating the need for downloading the data for processing.

Chapter 6: Conclusions

This final chapter consists of a summary of the thesis project, recalling the contents of each chapter and the relevance of the project.

This thesis project corresponds to the continuation of the work done by Smith et al. (2023), and aims to propose a workflow for the creation of a DT of an apple orchard. The development of DTs aligns with the goals of precision agriculture and smart farming, which seek to optimise resource use, increase yield quality, and reduce environmental impact through data-driven decision-making. By integrating robotics, remote sensing, and computer vision, this work contributes to the advancement of automated, efficient, and sustainable agricultural practices. Moreover, the creation of DTs of living systems is a field not yet fully developed; therefore, this work also contributes to exploring its potential and related challenges and limitations.

Chapter 1 introduces the context of the project, exploring the concepts of precision agriculture and the research performed regarding it. Special attention is paid to the tools and instruments used in the agricultural field, and the application of more recent technologies, such as artificial intelligence and DTs.

Chapter 2 presents in depth the materials used for data acquisition, explaining their characteristics and their expected behaviours and outputs. It also describes the preparations previous to data acquisition, such as the field tests performed with a prototype and the adjustment of the sensor-carrying platform.

Chapter 3 describes data acquisition, the procedures followed to obtain digital representations of the orchard from the outputs of the optical sensors, and the process of apple detection and georeferencing.

Chapter 4 presents the results of the methods discussed in Chapter 3. In particular, the 3D models and point clouds created from the outputs of the cameras, and the detection and georeferencing of the apples.

Chapter 5 discusses the quality of the results obtained in Chapter 4. Limitations encountered and proposed solutions are also discussed, along with considerations for future research.

The materials used for data acquisition were the same as the ones used by Smith et al. (2023), but with a different configuration: one ZED 2 stereo camera and three MAPIR Survey3w, instead of one, were used. The configuration and orientation of the optical sensors were chosen in order to obtain sufficient overlap of the captured scene, and to avoid framing only the repetitive pattern of the tree rows. The optical sensors were mounted on Agri.Q, an agricultural rover with ideal characteristics for navigation in orchards.

Although a final DT integrating all relevant orchard information was not fully achieved, the methodology followed for data acquisition and processing allowed for the generation of 3D representations of the orchard, despite the fact that not all were entirely accurate. Additionally, apple detection and georeferencing were successfully performed, yet both processes leave room for significant improvement.

Annexes

Annex 1: MATLAB script for computation of overlap of multispectral cameras

```
clc
clear all
close all

syms x y

h = 2.5
theta = (43.5)*pi/180;

r = h*sin(theta)/sin((pi/2)-theta)

% displacement of the centre of the ellipse:

c=h*sin(30*pi/180)/sin(60*pi/180)

% rest of major axis:

d=h*sin(theta+(30*pi/180))/sin((pi/2)-(theta+(30*pi/180)))

a=h*sin(theta-(30*pi/180))/sin((pi/2)-(theta-(30*pi/180)))

major_axis=d+a

% Define the implicit equations of the two ellipses and the circle
a1 = major_axis/2; b1 = r;  % Semi-axes of ellipse 1
a2 = major_axis/2; b2 = r;  % Semi-axes of ellipse 2
r = h;           % Radius of the circle
%%
y_offset=1.1;
x_offset=(0.209+c);

ellipse1 = (x-x_offset/ a1)^2 + ((y-y_offset)/ b1)^2 - 1;  % Equation of left ellipse
ellipse2 = ((x + x_offset)/ a2)^2 + ((y - y_offset)/ b2)^2 - 1; % Equation of right
ellipse
circle = (x)^2 + (y - y_offset)^2 - r^2;  % Circle equation

% Solve the intersection points numerically
eq1 = solve([ellipse1 == 0, circle == 0], [x, y]);
eq2 = solve([ellipse2 == 0, circle == 0], [x, y]);
eq3 = solve([ellipse1 == 0, ellipse2 == 0], [x, y]);

% Convert solutions to numerical values
points1 = double([eq1.x, eq1.y]);
points2 = double([eq2.x, eq2.y]);
points3 = double([eq3.x, eq3.y]);

% Monte Carlo Integration to estimate overlapping area
numSamples = 1e6; % Number of random points
xMin = -max([a1, a2, r]); xMax = max([a1, a2, r]) + 3;
yMin = -max([b1, b2, r]); yMax = max([b1, b2, r]) + 3;

randX = xMin + (xMax - xMin) * rand(numSamples, 1);
randY = yMin + (yMax - yMin) * rand(numSamples, 1);

insideEllipse1 = ((randX - x_offset) / a1).^2 + ((randY - y_offset) / b1).^2 <= 1;
insideEllipse2 = ((randX + x_offset) / a2).^2 + ((randY - y_offset) / b2).^2 <= 1;
insideCircle = (randX.^2 + (randY-y_offset).^2) <= r^2;

overlapPoints = insideEllipse1 & insideEllipse2 & insideCircle;
overlapArea = sum(overlapPoints) / numSamples * (xMax - xMin) * (yMax - yMin);
```

```
% Compute the area of each shape
ellipse1Area = pi * a1 * b1;
ellipse2Area = pi * a2 * b2;
circleArea = pi * r^2;

% Compute overlap percentage relative to each figure
overlapPercentageEllipse1 = (overlapArea / ellipse1Area) * 100;
overlapPercentageEllipse2 = (overlapArea / ellipse2Area) * 100;
overlapPercentageCircle = (overlapArea / circleArea) * 100;

% Display results
fprintf('Estimated Overlapping Area: %.4f\n', overlapArea);
fprintf('Area of Ellipse 1: %.4f\n', ellipse1Area);
fprintf('Area of Ellipse 2: %.4f\n', ellipse2Area);
fprintf('Area of Circle: %.4f\n', circleArea);
fprintf('Overlap as Percentage of Ellipse 1: %.2f%%\n', overlapPercentageEllipse1);
fprintf('Overlap as Percentage of Ellipse 2: %.2f%%\n', overlapPercentageEllipse2);
fprintf('Overlap as Percentage of Circle: %.2f%%\n', overlapPercentageCircle);
```

Annex 2: Python script for apple detection and georeferencing

```
import sys
import numpy as np

import argparse
import torch
import cv2
import pyzed.sl as sl
from ultralytics import YOLO

from threading import Lock, Thread
from time import sleep

import ogl_viewer.viewer as gl
import cv_viewer.tracking_viewer as cv_viewer

import os
from datetime import datetime

import pyproj
from pyproj import Proj, transform

import open3d as o3d

import laspy


lock = Lock()
run_signal = False
exit_signal = False

camera_height = 1.6 #meters
# using UTM coordinates

altitude = 383.73 # meters above sea level
utm_x = 32507.127
utm_y = 4940652
# utm_z = 320.325
# utm_z= altitude + camera_height


def xywh2abcd(xywh, im_shape):
    output = np.zeros((4, 2))

    # Center / Width / Height -> BBox corners coordinates
    x_min = (xywh[0] - 0.5*xywh[2]) #* im_shape[1]
    x_max = (xywh[0] + 0.5*xywh[2]) #* im_shape[1]
```

```
    y_min = (xywh[1] - 0.5*xywh[3]) #* im_shape[0]
    y_max = (xywh[1] + 0.5*xywh[3]) #* im_shape[0]

    # A ------ B
    # | Object |
    # D ------ C

    output[0][0] = x_min
    output[0][1] = y_min

    output[1][0] = x_max
    output[1][1] = y_min

    output[2][0] = x_max
    output[2][1] = y_max

    output[3][0] = x_min
    output[3][1] = y_max
    return output


def detections_to_custom_box(detections, im0):
    output = []
    for i, det in enumerate(detections):
        xywh = det.xywh[0]

        # Creating ingestable objects for the ZED SDK
        obj = sl.CustomBoxObjectData()
        obj.bounding_box_2d = xywh2abcd(xywh, im0.shape)
        obj.label = det.cls
        obj.probability = det.conf
        obj.is_grounded = False
        output.append(obj)
    return output


def torch_thread(weights, img_size, conf_thres=0.2, iou_thres=0.45):
    global image_net, exit_signal, run_signal, detections

    print("Intializing Network...")

    model = YOLO(weights)

    output_dir = "saved_frames"
    os.makedirs(output_dir, exist_ok=True)  # Create directory if it doesn't exist

    frame_counter=0
    while not exit_signal:
        if run_signal:
```

```
            lock.acquire()

            img = cv2.cvtColor(image_net, cv2.COLOR_RGBA2RGB)
            # https://docs.ultralytics.com/modes/predict/#video-suffixes
            #save each frame:
            det = model.predict(img, save=True, imgsz=img_size, conf=conf_thres,
iou=iou_thres)[0].cpu().numpy().boxes
            #det = model.predict(img, imgsz=img_size, conf=conf_thres,
iou=iou_thres)[0].cpu().numpy().boxes

            # Save the image with a unique filename (using timestamp + frame number)
            timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
            filename = os.path.join(output_dir,
f"frame_{timestamp}_{frame_counter}.jpg")
            #cv2.imwrite(filename, img)
            #print(f"Saved {filename}")

            frame_counter += 1  # Increment frame count

            # ZED CustomBox format (with inverse letterboxing tf applied)
            detections = detections_to_custom_box(det, image_net)
            lock.release()
            run_signal = False
        sleep(0.01)


def main():
    global image_net, exit_signal, run_signal, detections

    apple_positions = []
    apple_dimensions = []
    utm_positions = []

    capture_thread = Thread(target=torch_thread, kwargs={'weights': opt.weights,
'img_size': opt.img_size, "conf_thres": opt.conf_thres})
    capture_thread.start()

    print("Initializing Camera...")

    zed = sl.Camera()

    input_type = sl.InputType()
    if opt.svo is not None:
        input_type.set_from_svo_file(opt.svo)

    # Create a InitParameters object and set configuration parameters
    init_params = sl.InitParameters(input_t=input_type, svo_real_time_mode=False)
    init_params.coordinate_units = sl.UNIT.METER
    init_params.depth_mode = sl.DEPTH_MODE.ULTRA  # QUALITY
```

```
    init_params.coordinate_system = sl.COORDINATE_SYSTEM.LEFT_HANDED_Z_UP
    init_params.depth_maximum_distance = 40

    runtime_params = sl.RuntimeParameters()
    runtime_params.measure3D_reference_frame = sl.REFERENCE_FRAME.WORLD
    status = zed.open(init_params)

    if status != sl.ERROR_CODE.SUCCESS:
        print(repr(status))
        exit()

    image_left_tmp = sl.Mat()

    print("Initialized Camera")

    # Create initial world transform matrix
    initial_position = sl.Transform()
    initial_translation = sl.Translation()
    #initial_translation.init_vector(utm_x, utm_y, altitude)
    initial_translation.init_vector(0, 0, 0)

    initial_position.set_translation(initial_translation)

    positional_tracking_parameters = sl.PositionalTrackingParameters()
    positional_tracking_parameters.set_initial_world_transform(initial_position)
    positional_tracking_parameters.enable_area_memory = False

    print("Initial position set to:", initial_translation)

    positional_init = zed.enable_positional_tracking(positional_tracking_parameters)

    if positional_init != sl.ERROR_CODE.SUCCESS:
        print("[ZED][ERROR] Can't start tracking of camera : " + repr(status) + ".
Exit program.")
        exit()

    obj_param = sl.ObjectDetectionParameters()
    obj_param.detection_model = sl.OBJECT_DETECTION_MODEL.CUSTOM_BOX_OBJECTS
    obj_param.enable_tracking = True
    obj_param.enable_segmentation = False  # designed to give person pixel mask with
internal OD
    zed.enable_object_detection(obj_param)

    objects = sl.Objects()
    obj_runtime_param = sl.ObjectDetectionRuntimeParameters()

    # Display
    camera_infos = zed.get_camera_information()
    camera_res = camera_infos.camera_configuration.resolution
```

```
    # Create OpenGL viewer
    viewer = gl.GLViewer()
    point_cloud_res = sl.Resolution(min(camera_res.width, 720),
min(camera_res.height, 404))
    point_cloud_render = sl.Mat()
    viewer.init(camera_infos.camera_model, point_cloud_res,
obj_param.enable_tracking)
    point_cloud = sl.Mat(point_cloud_res.width, point_cloud_res.height,
sl.MAT_TYPE.F32_C4, sl.MEM.CPU)
    image_left = sl.Mat()
    # Utilities for 2D display
    display_resolution = sl.Resolution(min(camera_res.width, 1280),
min(camera_res.height, 720))
    image_scale = [display_resolution.width / camera_res.width,
display_resolution.height / camera_res.height]
    image_left_ocv = np.full((display_resolution.height, display_resolution.width,
4), [245, 239, 239, 255], np.uint8)


    # Utilities for tracks view
    camera_config = camera_infos.camera_configuration
    tracks_resolution = sl.Resolution(400, display_resolution.height)
    track_view_generator = cv_viewer.TrackingViewer(tracks_resolution,
camera_config.fps, init_params.depth_maximum_distance)
    track_view_generator.set_camera_calibration(camera_config.calibration_parameters)
    image_track_ocv = np.zeros((tracks_resolution.height, tracks_resolution.width,
4), np.uint8)
    # Camera pose
    cam_w_pose = sl.Pose()


    f=open("results.txt", "w")

    detected_ids = set()

    while viewer.is_available() and not exit_signal:
        if zed.grab(runtime_params) == sl.ERROR_CODE.SUCCESS:
            # -- Get the image
            lock.acquire()
            zed.retrieve_image(image_left_tmp, sl.VIEW.LEFT)
            image_net = image_left_tmp.get_data()
            lock.release()
            run_signal = True

            # -- Detection running on the other thread
            while run_signal:
                sleep(0.001)

            # Wait for detections
```

```
            lock.acquire()
            # -- Ingest detections
            zed.ingest_custom_box_objects(detections)
            lock.release()
            zed.retrieve_objects(objects, obj_runtime_param)


            # print id and position for all objects in list
            for object in objects.object_list:
                print("{} {} {}".format(object.id, object.dimensions,
object.position))
                f.write("{} {} {}\n".format(object.id, object.dimensions,
object.position))

                if object.id not in detected_ids:
                    apple_pos = object.position
                    apple_size = object.dimensions
                    if np.all(np.isfinite(apple_pos)):
                        apple_positions.append([apple_pos[0], apple_pos[1],
apple_pos[2]])
                    if np.all(np.isfinite(apple_size)):
                        apple_dimensions.append([apple_size[0], apple_size[1],
apple_size[2]])

                    detected_ids.add(object.id)

            # print("\n ids: \n", detected_ids)
            apple_positions = [pos for pos in apple_positions if
np.all(np.isfinite(pos))]

            # Convert local positions to UTM

            for pos in apple_positions:
                x_local, y_local, z_local = pos
                # ZED: X = forward, Y = right, Z = up (LEFT_HANDED_Z_UP)
                utm_x_pos = utm_x + x_local
                utm_y_pos = utm_y + y_local
                utm_z_pos = altitude + camera_height + z_local  # Altitude above sea
level
                utm_positions.append([utm_x_pos, utm_y_pos, utm_z_pos])

            utm_np = np.array(utm_positions)
            print("\n pos_np: \n", utm_np )

            apple_dimensions = [size for size in apple_dimensions if
np.all(np.isfinite(size))]
            sizes_np = np.array(apple_dimensions)
            print("\n size_np: \n", sizes_np)
            ### create point cloud
```

```
            # Split into X, Y, Z
            xs = utm_np[:,0]
            ys = utm_np[:,1]
            zs = utm_np[:,2]

            # print("\n zs: \n", zs)
            # Create header
            header = laspy.LasHeader(point_format=3, version="1.4")
            header.offsets = np.min(utm_np, axis=0)
            header.scales = np.array([0.001, 0.001, 0.001])  # scale to mm precision

            # Create point cloud
            las = laspy.LasData(header)
            las.x = xs
            las.y = ys
            las.z = zs

            las.red[:] = 255
            las.green[:] = 0
            las.blue[:] = 0

            # Add custom extra dimension: "apple_size"
            las.add_extra_dim(laspy.ExtraBytesParams(name="apple_size",
type=np.float32))
            las.apple_sizes = sizes_np

            # Save to .las
            las.write("apples_ip.las")

            # -- Display
            # Retrieve display data
            zed.retrieve_measure(point_cloud, sl.MEASURE.XYZRGBA, sl.MEM.CPU,
point_cloud_res)
            point_cloud.copy_to(point_cloud_render)
            zed.retrieve_image(image_left, sl.VIEW.LEFT, sl.MEM.CPU,
display_resolution)
            zed.get_position(cam_w_pose, sl.REFERENCE_FRAME.WORLD)
            print("camera positon: {}\n".format(zed.get_position))


            # 3D rendering
            viewer.updateData(point_cloud_render, objects)
            # 2D rendering
            np.copyto(image_left_ocv, image_left.get_data())
            cv_viewer.render_2D(image_left_ocv, image_scale, objects,
obj_param.enable_tracking)
            global_image = cv2.hconcat([image_left_ocv, image_track_ocv])
            # Tracking view
```

```
            track_view_generator.generate_view(objects, cam_w_pose, image_track_ocv,
objects.is_tracked)


            cv2.imshow("ZED | 2D View and Birds View", global_image)
            key = cv2.waitKey(10)
            if key == 27 or key == ord('q') or key == ord('Q'):
                exit_signal = True

        else:

            exit_signal = True


    viewer.exit()
    exit_signal = True
    zed.close()


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', type=str, default='apple_seg4.pt',
help='model.pt path(s)')
    parser.add_argument('--svo', type=str, default='test.svo', help='optional svo
file, if not passed, use the plugged camera instead')
    parser.add_argument('--img_size', type=int, default=416, help='inference size
(pixels)')
    parser.add_argument('--conf_thres', type=float, default=0.4, help='object
confidence threshold')
    opt = parser.parse_args()


    with torch.no_grad():
        main()
```

Bibliography

Abeyrathna, R. M. R. D., Nakaguchi, V. M., Minn, A., & Ahamed, T. (2023). Recognition and Counting of Apples in a Dynamic State Using a 3D Camera and Deep Learning Algorithms for Robotic Harvesting Systems. *Sensors*, *23*(8), 3810. https://doi.org/10.3390/s23083810

Auat Cheein, F. A., & Carelli, R. (2013). Agricultural Robotics: Unmanned Robotic Service Units in Agricultural Tasks. *IEEE Industrial Electronics Magazine*, *7*(3), 48–58. https://doi.org/10.1109/MIE.2013.2252957

Baluja, J., Diago, M. P., Balda, P., Zorer, R., Meggio, F., Morales, F., & Tardaguila, J. (2012a). Assessment of vineyard water status variability by thermal and multispectral imagery using an unmanned aerial vehicle (UAV). *Irrigation Science*, *30*(6), 511–522. https://doi.org/10.1007/s00271-012-0382-9

Baluja, J., Diago, M. P., Balda, P., Zorer, R., Meggio, F., Morales, F., & Tardaguila, J. (2012b). Assessment of vineyard water status variability by thermal and multispectral imagery using an unmanned aerial vehicle (UAV). *Irrigation Science*, *30*(6), 511–522. https://doi.org/10.1007/s00271-012-0382-9

Barrientos, A., Colorado, J., Cerro, J. D., Martinez, A., Rossi, C., Sanz, D., & Valente, J. (2011). Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, *28*(5), 667–689. https://doi.org/10.1002/rob.20403

Bechar, A., & Vigneault, C. (2016). Agricultural robots for field operations: Concepts and components. *Biosystems Engineering*, *149*, 94–111. https://doi.org/10.1016/j.biosystemseng.2016.06.014

Botta, A. (2022). *Sustainable Rover for Precision Agriculture*.

Botta, A., & Cavallone, P. (2022). Robotics Applied to Precision Agriculture: The Sustainable Agri.q Rover Case Study. In G. Quaglia, A. Gasparetto, V. Petuya, & G. Carbone (Eds.), *Proceedings of I4SDG Workshop 2021* (Vol. 108, pp. 41–50). Springer International Publishing. https://doi.org/10.1007/978-3-030-87383-7_5

Botta, A., Cavallone, P., Baglieri, L., Colucci, G., Tagliavini, L., & Quaglia, G. (2022). A Review of Robots, Perception, and Tasks in Precision Agriculture. *Applied Mechanics*, *3*(3), 830–854. https://doi.org/10.3390/applmech3030049

Cavallone, P., Botta, A., Carbonari, L., Visconte, C., & Quaglia, G. (2021). The Agri.q Mobile Robot: Preliminary Experimental Tests. In V. Niola & A. Gasparetto (Eds.), *Advances in Italian Mechanism Science* (Vol. 91, pp. 524–532). Springer International Publishing. https://doi.org/10.1007/978-3-030-55807-9_59

Cavallone, P., Visconte, C., Carbonari, L., Botta, A., & Quaglia, G. (2021). Design of the Mobile Robot Agri.q. In G. Venture, J. Solis, Y. Takeda, & A. Konno (Eds.), *ROMANSY 23—Robot Design, Dynamics and Control* (Vol. 601, pp. 288–296). Springer International Publishing. https://doi.org/10.1007/978-3-030-58380-4_35

Chu, P., Li, Z., Zhang, K., Chen, D., Lammers, K., & Lu, R. (2023). *O2RNet: Occluder-Occludee Relational Network for Robust Apple Detection in Clustered Orchard Environments* (No. arXiv:2303.04884). arXiv. https://doi.org/10.48550/arXiv.2303.04884

Coordinate Frames—Stereolabs. (n.d.). Retrieved 8 March 2025, from https://www.stereolabs.com/docs/positional-tracking/coordinate-frames

Gongal, A., Karkee, M., & Amatya, S. (2018). Apple fruit size estimation using a 3D machine vision system. *Information Processing in Agriculture*, *5*(4), 498–503. https://doi.org/10.1016/j.inpa.2018.06.002

Gonzalez-de-Soto, M., Emmi, L., Benavides, C., Garcia, I., & Gonzalez-de-Santos, P. (2016). Reducing air pollution with hybrid-powered robotic tractors for precision agriculture. *Biosystems Engineering*, *143*, 79–94. https://doi.org/10.1016/j.biosystemseng.2016.01.008

Grieves, M. (2015, March). *Digital Twin: Manufacturing Excellence through Virtual Factory Replication*.

Hani, N., Roy, P., & Isler, V. (2020). MinneApple: A Benchmark Dataset for Apple Detection and Segmentation. *IEEE Robotics and Automation Letters*, *5*(2), 852–858. https://doi.org/10.1109/LRA.2020.2965061

Harker, F. R., Gunson, F. A., & Jaeger, S. R. (2003). The case for fruit quality: An interpretive review of consumer attitudes, and preferences for apples. *Postharvest Biology and Technology*, *28*(3), 333–347. https://doi.org/10.1016/S0925-5214(02)00215-6

Jocher, G., Qiu, J., & Chaurasia, A. (2023). *Ultralytics YOLO* (Version 8.0.0) [Python]. https://github.com/ultralytics/ultralytics (Original work published 2022)

Jones, H. G., Serraj, R., Loveys, B. R., Xiong, L., Wheaton, A., & Price, A. H. (2009). Thermal infrared imaging of crop canopies for the remote diagnosis and quantification of plant responses to water stress in the field. *Functional Plant Biology*, *36*(11), 978. https://doi.org/10.1071/FP09123

Kundu, R. (2023, January 17). *YOLO Algorithm for Object Detection Explained [+Examples]*. https://www.v7labs.com/blog/yolo-object-detection

Martino, I., Agustí-Brisach, C., Nari, L., Gullino, M. L., & Guarnaccia, V. (2024). Characterization and Pathogenicity of Fungal Species Associated with Dieback of Apple Trees in Northern Italy. *Plant Disease*, *108*(2), 311–331. https://doi.org/10.1094/PDIS-04-23-0645-RE

Miranda, J. C., Gené-Mola, J., Zude-Sasse, M., Tsoulias, N., Escolà, A., Arnó, J., Rosell-Polo, J. R., Sanz-Cortiella, R., Martínez-Casasnovas, J. A., & Gregorio, E. (2023). Fruit sizing using AI: A review of methods and challenges. *Postharvest Biology and Technology*, *206*, 112587. https://doi.org/10.1016/j.postharvbio.2023.112587

Moysiadis, V., Sarigiannidis, P., Vitsas, V., & Khelifi, A. (2021). Smart Farming in Europe. *Computer Science Review*, *39*, 100345. https://doi.org/10.1016/j.cosrev.2020.100345

Nanni, L., Ghidoni, S., & Brahnam, S. (2017). Handcrafted vs. Non-handcrafted features for computer vision classification. *Pattern Recognition*, *71*, 158–172. https://doi.org/10.1016/j.patcog.2017.05.025

Naranjo-Torres, J., Mora, M., Hernández-García, R., Barrientos, R. J., Fredes, C., & Valenzuela, A. (2020). A Review of Convolutional Neural Network Applied to Fruit Image Processing. *Applied Sciences*, *10*(10), Article 10. https://doi.org/10.3390/app10103443

Nawar, S., Corstanje, R., Halcro, G., Mulla, D., & Mouazen, A. M. (2017). Delineation of Soil Management Zones for Variable-Rate Fertilization. In *Advances in Agronomy* (Vol. 143, pp. 175–245). Elsevier. https://doi.org/10.1016/bs.agron.2017.01.003

Pierce, F. J., & Nowak, P. (1999). Aspects of Precision Agriculture. In *Advances in Agronomy* (Vol. 67, pp. 1–85). Elsevier. https://doi.org/10.1016/S0065-2113(08)60513-1

Pylianidis, C., Osinga, S., & Athanasiadis, I. N. (2021). Introducing digital twins to agriculture. *Computers and Electronics in Agriculture*, *184*, 105942. https://doi.org/10.1016/j.compag.2020.105942

Radoglou-Grammatikis, P., Sarigiannidis, P., Lagkas, T., & Moscholios, I. (2020). A compilation of UAV applications for precision agriculture. *Computer Networks*, *172*, 107148. https://doi.org/10.1016/j.comnet.2020.107148

Ragazou, K., Garefalakis, A., Zafeiriou, E., & Passas, I. (2022). Agriculture 5.0: A New Strategic Management Mode for a Cut Cost and an Energy Efficient Agriculture Sector. *Energies*, *15*(9), 3113. https://doi.org/10.3390/en15093113
Ramseyer, N. (2016, August 2). *Processing Survey Camera Images in Agisoft Metashape/Photoscan*. Processing Survey Camera Images in Agisoft Metashape/Photoscan. https://www.mapir.camera/en-gb/blogs/guide/processing-survey-camera-images-in-agisoft-photoscan
Reina, G., Milella, A., Rouveure, R., Nielsen, M., Worst, R., & Blas, M. R. (2016). Ambient awareness for agricultural robotic vehicles. *Biosystems Engineering*, *146*, 114–132. https://doi.org/10.1016/j.biosystemseng.2015.12.010
Smith, K., Botta, A., Colucci, G., Piras, M., & Quaglia, G. (2023). *Orchard digital twin: A prototype for smart agricultural monitoring*.
Stanford-Clark, A., Frank-Schultz, E., & Harris, M. (2019, November 13). *What are digital twins?* IBM Developer. https://developer.ibm.com/articles/what-are-digital-twins/
Sun, X., Zheng, Y., Wu, D., & Sui, Y. (2024). Detection of Orchard Apples Using Improved YOLOv5s-GBR Model. *Agronomy*, *14*(4), 682. https://doi.org/10.3390/agronomy14040682
Sylvester, G. (Ed.). (2018). *E-agriculture in action: Drones for agriculture*. Food and Agriculture Organization ofn the United Nations and International Telecommunication Union.
Torres, J. (2024, March 22). *YOLOv8 Improvements: Key Architectural Enhancements*. https://yolov8.org/yolov8-improvements/
Ultralytics (Director). (2025, March 5). *How to Use SAHI with Ultralytics YOLO11 for Object Detection in Drone Footage | Real-time or Not?* 🚀 [Video recording]. https://www.youtube.com/watch?v=ILqMBah5ZvI
V7 | AI Document Processing & Data Labelling. (n.d.). Retrieved 11 March 2025, from https://www.v7labs.com/
Vecchio, Y., Agnusdei, G. P., Miglietta, P. P., & Capitanio, F. (2020). Adoption of Precision Farming Tools: The Case of Italian Farmers. *International Journal of Environmental Research and Public Health*, *17*(3), 869. https://doi.org/10.3390/ijerph17030869
Verdouw, C., & Kruize, J. W. (2017). *Digital twins in farm management: Illustrations from the FIWARE accelerators SmartAgriFood and Fractals*. https://doi.org/10.5281/zenodo.893662
Vulpi, F., Marani, R., Petitti, A., Reina, G., & Milella, A. (2022). An RGB-D multi-view perspective for autonomous agricultural robots. *Computers and Electronics in Agriculture*, *202*, 107419. https://doi.org/10.1016/j.compag.2022.107419
Wachs, J. P., Stern, H. I., Burks, T., & Alchanatis, V. (2010). Low and high-level visual feature-based apple detection from multi-modal images. *Precision Agriculture*, *11*(6), 717–735. https://doi.org/10.1007/s11119-010-9198-x
Wang, Q., Nuske, S., Bergerman, M., & Singh, S. (2013). Automated Crop Yield Estimation for Apple Orchards. In J. P. Desai, G. Dudek, O. Khatib, & V. Kumar (Eds.), *Experimental Robotics* (Vol. 88, pp. 745–758). Springer International Publishing. https://doi.org/10.1007/978-3-319-00065-7_50
Yandun Narvaez, F., Reina, G., Torres-Torriti, M., Kantor, G., & Cheein, F. A. (2017). A Survey of Ranging and Imaging Techniques for Precision Agriculture Phenotyping. *IEEE/ASME Transactions on Mechatronics*, *22*(6), 2428–2439. https://doi.org/10.1109/TMECH.2017.2760866
YOLO Object Detection Explained: Evolution, Algorithm, and Applications. (2024, April 4). https://encord.com/blog/yolo-object-detection-guide/

Zaman-Allah, M., Vergara, O., Araus, J. L., Tarekegne, A., Magorokosho, C., Zarco-Tejada, P. J., Hornero, A., Albà, A. H., Das, B., Craufurd, P., Olsen, M., Prasanna, B. M., & Cairns, J. (2015). Unmanned aerial platform-based multi-spectral imaging for field phenotyping of maize. *Plant Methods*, *11*(1), 35. https://doi.org/10.1186/s13007-015-0078-2

Zhang, C., Zou, K., & Pan, Y. (2020). A Method of Apple Image Segmentation Based on Color-Texture Fusion Feature and Machine Learning. *Agronomy*, *10*(7), Article 7. https://doi.org/10.3390/agronomy10070972

Zhang, G., Tian, Y., Yin, W., & Zheng, C. (2024). An Apple Detection and Localization Method for Automated Harvesting under Adverse Light Conditions. *Agriculture*, *14*(3), 485. https://doi.org/10.3390/agriculture14030485