

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Vision-based Smart Sprayer for Precision Farming

Thaidy Deguchi

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Filipe Baptista Neves dos Santos

Co-Supervisor: Hélio Mendes de Sousa Mendonça

July 24, 2023

Resumo

Atualmente, a sustentabilidade destaca-se como um dos principais tópicos de discussão. As alterações climáticas, os recursos limitados e o crescimento da população exigem uma gestão mais eficaz dos métodos agrícolas para minimizar o desperdício de alimentos e a degradação do solo.

Desde o aparecimento da agricultura, os seres humanos recorreram a produtos químicos para combater as pragas e reduzir as perdas na agricultura. Com o tempo, esta prática evoluiu para o que hoje designamos por pesticidas, que podem potencialmente danificar o solo devido à sua aplicação indireta. Por outro lado, a agricultura de precisão utiliza a IA e técnicas de processamento de imagem para regular a quantidade de produtos químicos utilizados em tipos específicos de plantas. Esta abordagem minimiza as perdas e otimiza a utilização dos recursos. Os estudos anteriores analisados nesta tese de mestrado demonstraram resultados positivos nesta área. No entanto, ainda há espaço para melhorias na precisão, exatidão e aspectos mecânicos do sistema de pulverização. Este trabalho tem como objetivo compilar, estudar, executar e testar estas melhorias para propor a solução " GraDeS " (Grape Detection Sprayer).

Para atingir o objetivo desta dissertação, que envolve o desenvolvimento de um algoritmo funcional para detetar e pulverizar pesticidas exclusivamente em cachos de uva dentro de uma cultura, bem como projetar um pulverizador com dois graus de liberdade, serão realizados os seguintes passos:

- Realização de uma revisão sistemática dos algoritmos de aprendizagem computacional existentes.
- Aperfeiçoar códigos anteriores para melhorar os seus resultados e exatidão.
- Investigar e projetar uma estrutura para o sistema de pulverização de precisão com dois graus de liberdade.
- Testar e validar o trabalho desenvolvido.

Abstract

Currently, sustainability stands out as one of the main topics of discussion. Climate change, limited resources, and population growth necessitate more effective management of agricultural methods to minimize food waste and soil degradation.

Since the advent of agriculture, humans have resorted to chemicals to combat pests and reduce losses in farming. Over time, this practice has evolved into what we now refer to as pesticides, which can potentially harm the soil due to indirect application. On the other hand, precision agriculture leverages AI and image processing techniques to regulate the amount of chemicals used on specific types of plants. This approach minimizes losses and optimizes resource utilization. Previous studies reviewed in this master's thesis have demonstrated positive outcomes in this area. However, there remains room for improvement in the sprayer system's precision, accuracy, and mechanical aspects. This work aims to compile, study, execute, and test these enhancements to propose the " GraDeS " (Grape Detection Sprayer) solution .

To achieve the objective of this dissertation, which involves spraying pesticides exclusively on grape bunches within a crop that are detected by an AI developed algorithm, as well as designing a sprayer with two degrees of freedom, the following steps will be undertaken:

- Conducting a systematic review of existing machine learning algorithms.
- Enhancing previous codes to improve their results and accuracy.
- Investigating and designing a structure for the precision spray system with two degrees of freedom.
- Testing and validating the developed work.

Graphical Abstract

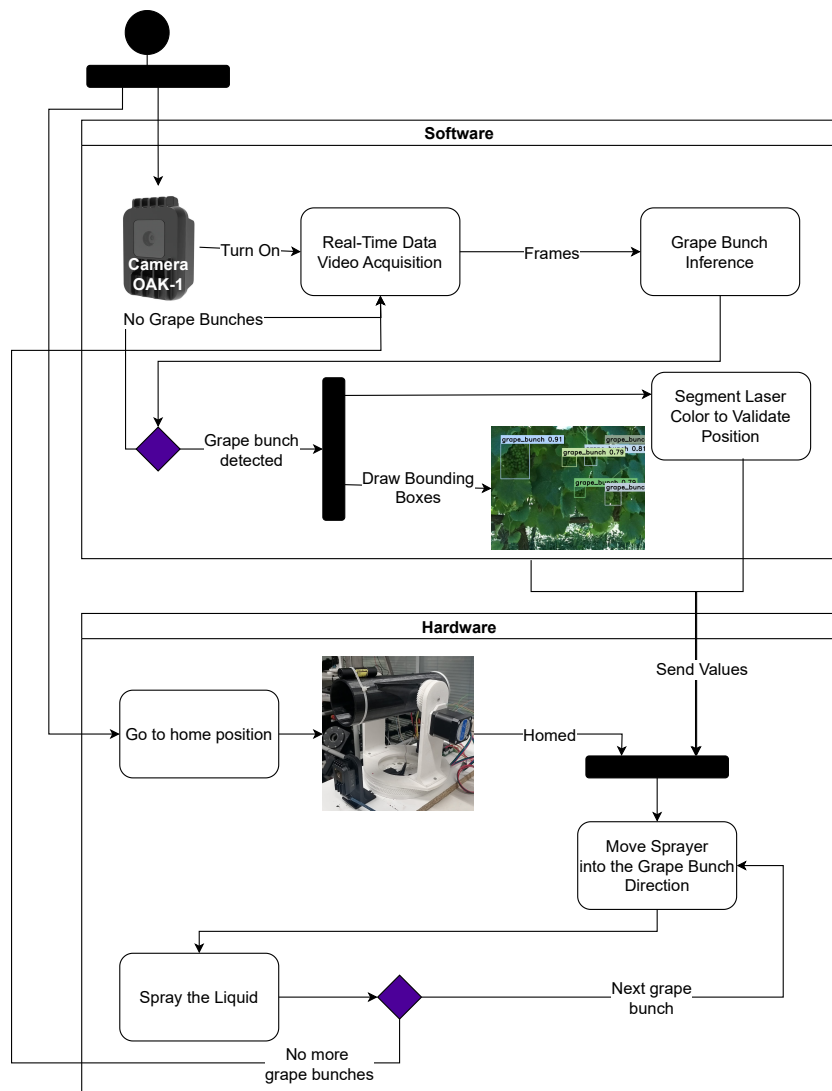


Figure 1: Graphical Abstract

Acknowledgments

First, I am very grateful to my advisor, Dr. Filipe Neves dos Santos. His immense knowledge and experience have encouraged me throughout my academic research and daily life. I would also like to thank Dr. Hélio Mendes de Sousa Mendonça for his accurate advice and encouragement during the dissertation work. Additionally, I would also like to thank MSc. André Rodrigues Baltazar for his invaluable guidance, continuous support, and patience during the development of the dissertation. He has always been available to help, remove doubts, and encourage me. Moreover, I express my thanks to all team members in the laboratory. They have been continuously available to assist me in the most diverse areas.

Furthermore, I would like to express my gratitude to my parents, Anna Cristina Borges Deguchi and Yoiti Deguchi, who allowed me to study and complete my degree at a renowned University in Portugal. They were not afraid to emigrate from Brazil to provide better opportunities for our family. Thanks to my family in Brazil for providing me with an excellent environment to grow full of love and friendship, and a special thanks to my grandma, Heloisa Wermelinger Borges, who always believed in me. I would also like to thank my girlfriend, Maria João Lourenço, for always being by my side during the good and bad moments, never letting me down or losing focus.

Also, I would like to thank the "Projeto Solução" institute, professors Carlos Eduardo Hespanha Matt and Ademilson Pereira Xavier Junior, and all my friends and colleagues for helping shape the person I have become through Judo and sport.

Finally, I thank my friends and the Tuna de Engenharia da Universidade do Porto. I could only complete my studies with your tremendous understanding and encouragement over the past few years.

Thaidy Deguchi

*“Let everything happen to you
Beauty and terror
Just keep going
No feeling is final”*

Rainer Maria Rilke

Contents

Resumo	i
Abstract	iii
Graphical Abstract	v
1 Introduction	1
1.1 Context	1
1.2 Motivation	2
1.3 Objectives	2
1.4 Thesis Contribution and Document Organization	3
2 State of the art and Theoretical Background	5
2.1 Introduction	5
2.2 Background Work	6
2.3 Software Theoretical Background	8
2.4 Hardware Theoretical Background	16
2.5 Conclusion	22
3 System Architecture and Requirements	23
3.1 System Architecture	23
3.2 System Requirements	24
3.3 Conclusion	26
4 Deep Learning Models for Grape Bunch Detection	27
4.1 Tools and Methodology	27
4.2 Dataset and Deep Learning Models Training.	30
4.3 Results	35
4.4 Conclusion	39
5 Precision Sprayer Hardware Development	41
5.1 Tools and Methodology	41
5.2 Camera Device	42
5.3 Material Test	43
5.4 PixelDriver v3	50
5.5 Spraying Nozzle and Fan	51
5.6 3D Model	52
5.7 Sprayer Operational Angle	53
5.8 Results	55

5.9	Conclusion	56
6	Hardware and Software Integration	57
6.1	Introduction	57
6.2	Tools and Methodology	57
6.3	Sprayer Input Value	58
6.4	Code Integration	59
6.5	Control Process	60
6.6	Results	61
6.7	Conclusion	62
7	Conclusions and Future Work	66
A	Filament Material	68
A.1	Introduction	68
B	PixelDriver3 Schematic	70
	References	72

List of Figures

1	Graphical Abstract	v
1.1	Different types of sprayers used in agriculture	2
2.1	Evolution of "Precision Agriculture" articles since 1990	5
2.2	Spray hardware 3D model and description from 2021 paper	8
2.3	SVM example in 2-dimensional space	9
2.4	AI field main division	10
2.5	Model of neural network	11
2.6	MobileNet model architecture	12
2.7	MobileNetV1 and MobileNetV2 results comparison	12
2.8	Architecture of YOLO model	13
2.9	Performance comparison of YOLO models	13
2.10	3D structure of the sprayer developed	17
2.11	Six freedom degree diagram	18
2.13	Matrix examples	20
2.14	Hall effect theoretical diagram	21
2.15	Hall effect position sensor example	22
3.1	GraDes architecture relationship	24
3.2	GraDes system workflow	25
4.1	RAM usage during the training process	29
4.2	FPS performance tests in the mobile device for different image sizes and models .	30
4.3	Training set image examples	31
4.4	Yolov8n.pt model architecture	32
4.5	Performance of different YOLOv5 models	33
4.6	Labeled images in YOLO	33
4.7	Prediction image after validation step	36
4.8	Prediction image results after running the inference in test set	37
4.9	Training results along the epochs	37
4.10	ROC curves	38
4.11	Results of grape bunch recognition with video capture	39
5.1	Pan and tilt mechanism	41
5.2	OAK-1 size comparison with a quarter coin	42
5.3	OAK-1 mechanical information	43
5.4	Stepper motor driver DVR8825 pin out	44
5.5	Motor experimental setup using an Arduino UNO	46
5.6	Physical motor test setup	47

5.7	Hall sensor experimental setup	48
5.8	Physical hall sensor test setup	48
5.9	Homing mechanism experimental setup	49
5.10	Homing mechanism experimental setup	50
5.11	Physical homing mechanism experimental setup	50
5.12	PixelDriver v3 INESC-TEC	51
5.13	Sprayer fan mechanism	52
5.14	Hall sensor and magnet structure examples	53
5.15	Sprayer 3d model	53
5.16	Sprayer angle model representation	54
5.17	Mechanism after homing process	55
5.18	Sprayer positions validation	56
6.1	Coordinate vector referential	58
6.2	Testing leafed background	61
6.3	First test results	63
6.4	Second test results	64
6.5	Third test results	65

List of Tables

2.1	Visual concept summary comparison table	15
4.1	FPS Performance comparison in the mobile device for different image sizes and models	30
4.2	Preliminary training parameters used in YOLO training	31
4.3	Different model sizes after 150 epoch training	32
4.4	Models performance before appending more images to the dataset	35
4.5	Models performance after appending more images to the dataset	36
4.6	Model final result after 300 epochs	38
4.7	Model SSD mobilenetV2 parameters and results	39
5.1	OAK-1 sensor characteristics	43
5.2	Stepper motor 42STH48-2504AC electric specifications	45
5.3	Driver DRV8825 carrier board specifications	45

List of Algorithms

4.1	FPS calculation algorithm	29
4.2	OpenCV Inference Process	34
4.3	Depthai Inference Process	34
4.4	Laser Color Segmentation	35
5.5	Motor testing	46
5.6	Hall sensor testing	47
5.7	Home position testing	49

Abbreviations and Symbols

AI	Artificial Intelligence
CIVE	Color Index of Vegetation Extraction
CNN	Convolutional Neural Networks
DL	Deep Learning
DETR	Detection with Transformers
DOF	Degrees of Freedom
ExG	Excess Green Index
ExGR	Excess Green minus Excess Red Index
ExR	Excess Red Index
GLI	Green Leaf Index
GPU	Graphics Processing Unit
HSV	Hue-Saturation-Value
ID	Identification number
IP	Ingress Protection
LBP	Local Binary Pattern
MExG	Modified Excess Green
NGRDI	Normalized green-red difference index
OpenCV	Open Source Computer Vision Library
PTO	Power take-off
RGB	Reg-Green-Blue
RGBVI	Reg-Green-Blue Vegetation Index
RI	Redness Index
SVM	Space Vector Machine
VEG	Vegetative Index
YOLO	You Only Look Once

Chapter 1

Introduction

Precision farming has evolved as a disruptive method that combines cutting-edge technologies to maximize resource efficiency and decrease environmental impact in the quest of sustainable agricultural practices and higher crop output. This master's thesis dissertation goes into the world of precision farming, with a particular emphasis on viticulture, the art and science of grape growing. The project aims to create an innovative smart sprayer system that uses deep learning algorithms to precisely detect, track, and spray grape bunches in farms.

1.1 Context

One of the foremost challenges faced today is the effective management of resource waste. In agriculture, it is crucial to capitalize on every available means to maximize production and minimize the loss of pesticides during spraying activities.

Driven by economic and cultural factors and the ever-increasing demand for food, research in this field continues to advance. Precision agriculture has undergone significant evolution to meet the requirements of fruit availability and preservation within specific periods of the year.

Moreover, using human labor in agricultural fields presents substantial challenges regarding efficiency and resource conservation. Spray operations rely on backpack or air blast sprayers attached to small tractors. However, both of these approaches have inherent limitations. Figure 1.1 illustrates the different types of sprayers. Unlike an autonomous machine, the manual approach significantly hampers the assurance of consistent quality and quantity of materials used. Factors such as distractions, negligence, fatigue, or a scarcity of local labor can contribute to these issues.

Conversely, such problems can be mitigated with the use of robots. On the other hand, the tractor-based approach suffers from issues such as off-target spraying and soil compaction, resulting in low spraying efficiency [1]. Consequently, the development of automated robots employing vision-based algorithms proves environmental impact and production efficiency advantageous.



(a) Manual Based Sprayer [2]



(b) Tractor Based Sprayer [3]



(c) Intelligent Sprayer developed by INESC TEC in 2021 [1]

Figure 1.1: Different types of sprayers used in agriculture

1.2 Motivation

Driven by this pressing societal challenge there are three key motivations underlying this dissertation. Firstly, it seeks to contribute to scientific progress by studying and implementing various machine-learning solutions, thereby aiding technological advancements in precision agriculture.

Secondly, a fundamental pillar of this work is reducing resource waste and excessive usage, which directly correlates with environmental damage. Chemicals and artificial products pose significant risks to human health and the environment. Excess pesticide residues in food can lead to various diseases in the human body. Similarly, the overuse of pesticides in the soil can damage its composition, deplete nutrients, and render it infertile. Moreover, excessive use of resources incurs additional costs and expenses. Therefore, the development of precision agriculture techniques to mitigate excess usage is of paramount importance.

By addressing these motivations, this work endeavors to make a meaningful contribution toward scientific advancement, environmental preservation, and resource efficiency.

1.3 Objectives

The primary objective of this dissertation is to develop an intelligent sprayer that enhances precision and targets specific areas of the plant, particularly grape bunches, through a two-degree-of-freedom system. This intelligent sprayer aims to cover a larger area of vegetation with a single spraying nozzle. An advanced perception system will be developed to achieve precise control over the mechanical system.

The perception system will detect the distinctive characteristics of grape bunches within a vineyard. It will then apply the appropriate pesticide dosage, significantly reducing pesticide waste to an optimal level that minimizes environmental and soil harm. By doing so, this solution aims to address the limitations associated with conventional approaches. Additionally, this work intends to enhance the previous algorithm and mechanical solutions to improve the overall effectiveness of the project.

The following specific goals must be accomplished to fulfill this objective:

- Study and design a structure for a precision sprayer with two degrees of freedom.

- Select the necessary hardware for controlling the motion of the precision sprayer.
- Implement motion control for the two-degree-of-freedom precision sprayer.
- Develop a vision-based perception algorithm capable of accurately tracking grape bunches.
- Integrate the perception algorithm into the precision sprayer.

By completing these objectives, the dissertation work will advance the field of precision agriculture and contribute to the development of an intelligent sprayer that optimizes resource usage and minimizes environmental impact.

1.4 Thesis Contribution and Document Organization

This dissertation significantly contributes to precision agriculture, robotics, and computer vision.

Firstly, it conducts a comprehensive literature review of critical concepts employed in this work, such as OpenCV and deep learning. It also analyzes various machine-learning approaches to determine the most suitable project. The conclusions drawn from this analysis can be valuable to similar research endeavors.

Furthermore, the dissertation provides two theoretical solutions for the problem at hand. Rather than focusing on a single technique, the background theory of both solutions is compiled in one place, facilitating researchers who may want to explore alternative approaches.

Moreover, the literature review of optimal mechanical approaches for the sprayer system enhances efficacy with the two degrees of freedom. The mechanism described can serve as a foundation for various other techniques, allowing for future upgrades and expansions, potentially involving additional degrees of freedom.

The conceptualization of a vision-based intelligent sprayer contributes to resource conservation, reduction of environmental damage, and technological advancement in the field.

Lastly, all codes, scripts, and algorithms developed for machine control, real-time tracking, and camera integration are publicly available on the INESCITEC GitLab page, ensuring accessibility and facilitating further research and development.

The dissertation is structured into seven chapters. In addition to Chapter 1, which serves as an introduction, Chapter 2 provides background work conducted over the past decades. It includes a theoretical exploration of different methods for extracting descriptors used in training and testing the algorithm and an overview of various architectures for implementing deep-learning object detectors. This chapter presents the formulas, theory, and benefits associated with these methods, along with a brief discussion of hardware theory.

Chapter 3 presents the proposed solution to the problem, showcasing the system architecture and requirements necessary for delivering the solution. It compiles all the essential steps required to produce a viable solution.

Chapter 4 details the software development process, including the methodology, tools, results, and conclusions. Similarly, Chapter 5 focuses on the hardware development aspects, thoroughly examining the work process and highlighting relevant points.

Chapter 6 integrates the main components of the work to validate the hardware and software in a semi-real environment. Finally, Chapter 7 concludes the report, presenting final considerations and outlining future avenues for research.

Chapter 2

State of the art and Theoretical Background

2.1 Introduction

Over the past decades, researchers have made continuous efforts to enhance precision agriculture techniques in order to minimize resource waste. This topic has gained increasing prominence in scientific research centers, as evidenced by the abundance of literature available. A search in the Scopus database using the query "precision" AND "agriculture" yielded more than 16,000 articles on the subject. Figure 2.1 illustrates the growth of published articles on precision agriculture from 1990 to 2022.

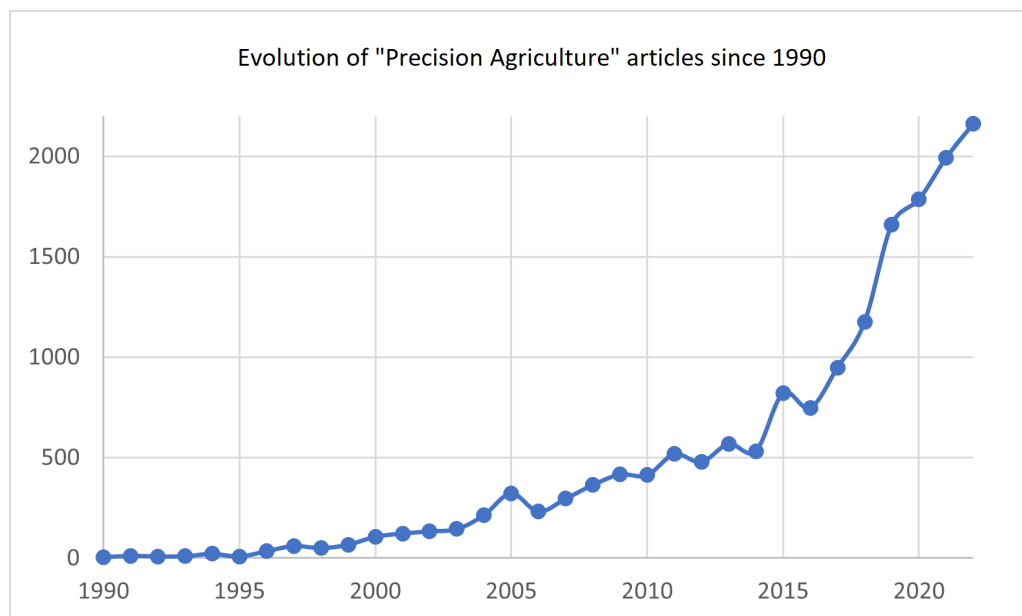


Figure 2.1: Evolution of "Precision Agriculture" articles since 1990

Today, numerous tools and technologies are available for manipulating various data types.

The field of machine learning has experienced exponential growth in recent years. Neural networks, support vector machines (SVM), and other techniques for training and testing datasets are commonly employed in precision agriculture research. Furthermore, advancements in computer vision have made it easier to process and interpret image information, offering valuable insights for precision agriculture projects.

This chapter provides an overview of different approaches to precision farming. It contextualizes the desired improvements envisioned by this work. The theoretical background analysis draws on various scientific articles and dissertations to contextualize the challenges faced in the field and their corresponding solutions. The theoretical background is divided into two main sections: software (2.3) and hardware (2.4). The software section explores various techniques for processing leaf images and identifying grape bunches, while the hardware section focuses on studying solutions for the two-degree-of-freedom sprayer system.

By reviewing the existing literature and understanding the advancements made in software and hardware, this dissertation aims to contribute to the ongoing progress in precision agriculture.

2.2 Background Work

The use of pesticides and chemicals in agriculture, while practical, can have harmful effects on the environment, soil, and food quality. These practices have led to regional and national extinctions of wild plants and animal species and have significantly altered agroecosystems [4]. Precision agriculture techniques have been developed and studied in response to these challenges over the past three decades.

In 1995, a study examined the consequences of varying herbicide dosages in precision agriculture. The research concluded that although there were advantages, unacceptable dynamic errors were observed, highlighting the need for improvements in control algorithms [5]. Subsequently, in 2009, precision spraying expanded to include pesticide treatments directly applied to agricultural plants, with the amount of insecticide determined by factors such as leaf shape and volume [4].

In 2015, a system was developed for providing tailored application rates based on canopy volume in pistachio orchards. This system utilized an electronic control system to detect and estimate tree canopy dimensions, leading to successful operation and application rate modification in an orchard setting [6]. Following this, in 2016, the first study on an automatic selective spraying system was conducted, focusing on disease detection in specialty crops. The results showed that the method could detect a significant percentage of the diseased area, and the robot reduced pesticide use by 65% to 85% [7] [8].

In 2021, the Precise Robotic Sprayer (PRySM) was introduced after years of research. This modular and precision terrestrial sprayer robot was designed to function independently over rough terrain. Tests on the developed sprayer demonstrated its potential to improve spraying accuracy and precision. Additionally, it was deemed suitable for use in small robots, although further improvements were identified for future work [1]. The technique employed in the PRySM system

was believed to enable proportional reductions in pesticide use without compromising biological efficacy.

The evolution of precision agriculture techniques and the development of innovative solutions like the PRySM demonstrates the ongoing efforts to mitigate the negative impacts of traditional pesticide spraying. These advancements aim to improve spraying accuracy, reduce pesticide use, and minimize the environmental consequences associated with conventional practices.

The 2021 paper proposes an autonomous robotic approach for precision spraying, utilizing a fully electrical sprayer instead of pneumatic sprayers and hydraulic actuators commonly used in other methods. This approach offers several advantages over alternative approaches, as reported in the study [1]. Some of the main advantages of a fully electrical spray system are:

- **Simplicity:** The fully electrical spray system offers a simpler design and operation compared to mechanical approaches, making it easier to use and maintain.
- **Enhanced maneuverability:** The electrical sprayer enables greater maneuverability, allowing the robot to navigate and spray in challenging environments more effectively.
- **Integration of variable-rate technology:** The electrical system facilitates the integration of variable-rate technology, which enables customized application rates based on specific crop needs, optimizing the use of pesticides and reducing waste.
- **Lightweight:** Electrical sprayers are generally lighter than their mechanical counterparts, resulting in reduced energy consumption and improved overall efficiency.
- **Compatibility with extremely low-volume spraying:** The fully electrical spray system is compatible with extremely low-volume spraying techniques, which can further minimize the use of pesticides and reduce environmental impact.
- **No need for power take-off (PTO) or hydraulic circuit:** Unlike mechanical sprayers that require a power take-off or hydraulic circuit from the tractor or machinery, electrical sprayers operate independently, providing more flexibility and autonomy.
- **Fewer repairs and leaks:** Electrical systems have fewer moving parts compared to mechanical systems, reducing the likelihood of breakdowns and leaks, and resulting in improved reliability and durability.

While there are a few disadvantages associated with fully electrical sprayers, such as the requirement for batteries (which can be more expensive) and the need for devices with high Ingress Protection (IP) ratings and resistance to extreme temperatures, the benefits outweigh the drawbacks. In comparison, mechanical approaches generally have more disadvantages than advantages in terms of complexity, maintenance, and efficiency.

The developed spray 3D model, as depicted in Fig. 2.2, provides a visual representation of the spray hardware, illustrating the innovative design of the fully electrical sprayer.

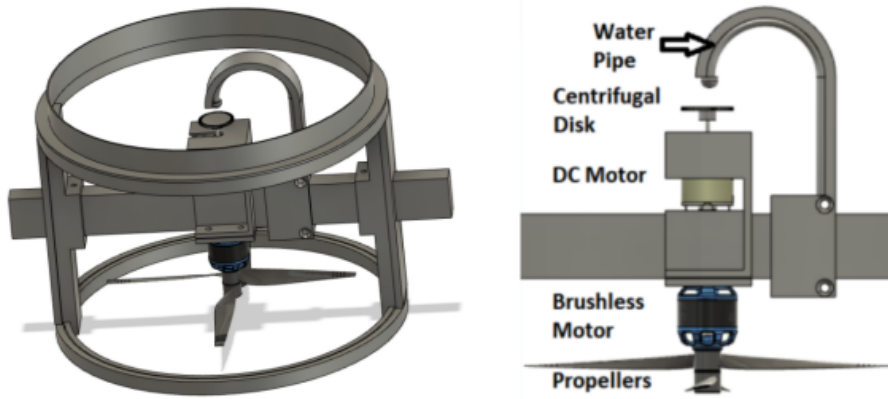


Figure 2.2: Spray hardware 3D model and description from 2021 paper[1]

In the previous work, the spray hardware model was designed to be fixed to an aluminum profile, which required manually adjusting the position of the spray drum to achieve different heights or applications. To address this inconvenience, the proposed modification suggests developing a model with two degrees of freedom, allowing more flexibility in adjusting the spraying parameters.

Regarding the image processing and analysis aspect of the previous work, the researchers employed a method based on extracting descriptors from concatenated histograms. These histograms were generated from six visual concepts: LPB (Local Binary Patterns), GLI (Green Leaf Index), RGBVI (Red-Green-Blue Vegetation Index), NGRDI (Normalized Green Red Difference Index), Hue, and Average of vegetation indexes.

Combining these six histograms, the researchers derived descriptors and applied them in a Support Vector Machine (SVM) algorithm for classification or detection purposes. The best-achieved accuracy in their study was reported as 85%. Building upon these findings and considering the advantages of the fully electrical spray system, the main objective of the current work is to explore different detection algorithms to achieve improved accuracy results.

The proposed modification aims to introduce two degrees of freedom to the spray hardware model for enhanced flexibility. Additionally, the current research focuses on testing alternative detection algorithms to enhance accuracy, building upon the previous work's findings and leveraging the advantages of the fully electrical spray system.

2.3 Software Theoretical Background

While hardware components are essential for a robot's physical embodiment, the software provides the intelligence and autonomy required for complex robotic systems. In machine learning-based robots, the software is a crucial intermediary between the physical world and the algorithms that govern their decision-making processes. The following section examines and consolidates several approaches to existing machine learning techniques.

2.3.1 Support vector machines

Support vector machines (SVM) are algorithms commonly employed in supervised machine learning models. One of the key advantages they offer over other classification methods is their remarkable accuracy.

In essence, SVM partitions the dataset into distinct classes by employing a hyperplane, and it accomplishes classification by constructing multidimensional training data derived from the initial training data (Fig. 2.3). Unlike traditional approaches that involve calculating features from the input, SVM directly extracts features from the data itself, thereby leveraging inherent patterns [9].

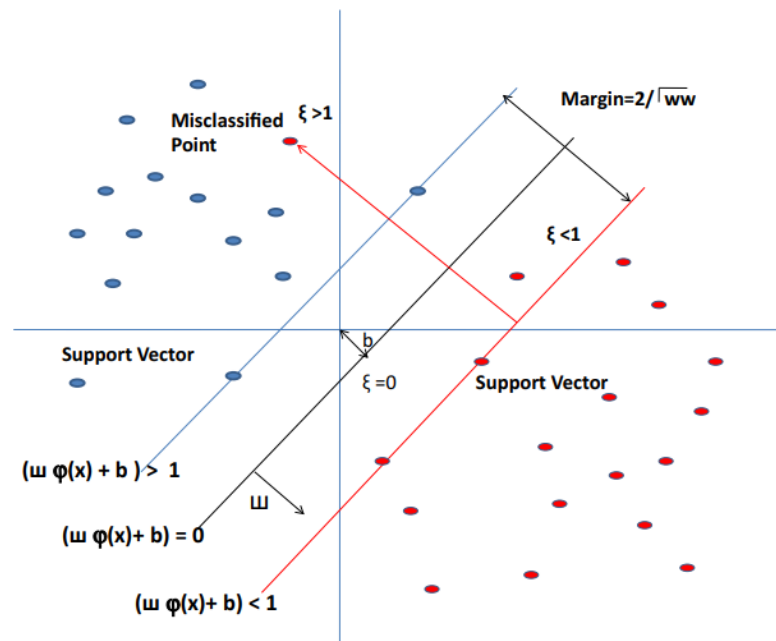


Figure 2.3: SVM example in 2-dimensional space [10]

The algorithm revolves around the following key ideas:

- Calculating an optimal hyperplane for linearly separable patterns.
- Maximizing the margin around the separating hyperplane, where a subset of training samples known as support vectors play a crucial role in determining the decision between two classes.
- Utilizing the Kernel function to extend the applicability of SVMs to handle non-linearly separable patterns. The kernel utilization is achieved by transforming and mapping the original data into a new space.

The hypothesis space is pivotal in minimizing the trade-off between empirical error and complexity. The trade-off is achieved by solving two minimization problems: SVM classification and SVM regression.

SVM Classification:

$$\min_{\mathbf{f}} \|\mathbf{f}\|_K^2 + C \sum_{i=1}^l |1 - y_i f(x_i)|_+$$

SVM Regression:

$$\min_{\mathbf{f}} \|\mathbf{f}\|_K^2 + C \sum_{i=1}^l |y_i - f(x_i)|_\varepsilon$$

The C constant, known as the "regularization parameter," is crucial in the SVM algorithm. This parameter allows control over the mentioned trade-off [11].

As mentioned earlier, in modern coding languages such as Python and Java, numerous libraries already provide pre-implemented solutions for the problem. Consequently, manual implementation of SVM for theoretical purposes is often unnecessary. Further details regarding the problem solution can be found in the referenced article [11].

Various SVM implementation techniques have emerged over the past few decades, including Twin SVM, Lagrangian SVM, Least Square SVM, Decision Tree SVM, DAG SVM, and Multi-Kernel Classification [10]. It is important to emphasize that only some categorization methods have been proven universally superior to others for all datasets. Each article adopts different strategies to enhance classification accuracy, sometimes employing a combination of approaches [10].

2.3.2 Deep Learning Methods

Within the realm of machine learning, there exists a prominent subfield known as deep learning (DL), which represents a division of artificial intelligence. Figure 2.4 visually represents the conceptual division between these fields.

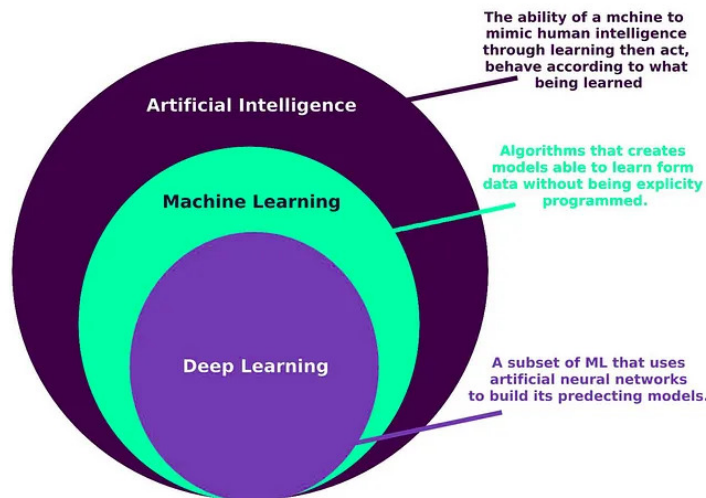


Figure 2.4: AI field main division [12]

Deep learning techniques leverage artificial neural networks to extract significant features or representations from input data automatically. Convolutional neural networks (CNN) are commonly employed in object detection and classification to learn hierarchical sets of relevant features for a given task.

At the core of deep learning lies the artificial neural network, which aims to emulate the human brain's information processing and decision-making mechanisms (deep learning) [13]. Perceptrons or neurons constitute the fundamental building blocks of neural networks. Furthermore, the model comprises three primary types of layers: the input layer, the output layer, and the hidden layers. Figure 2.5 illustrates the structure of a neural network model.

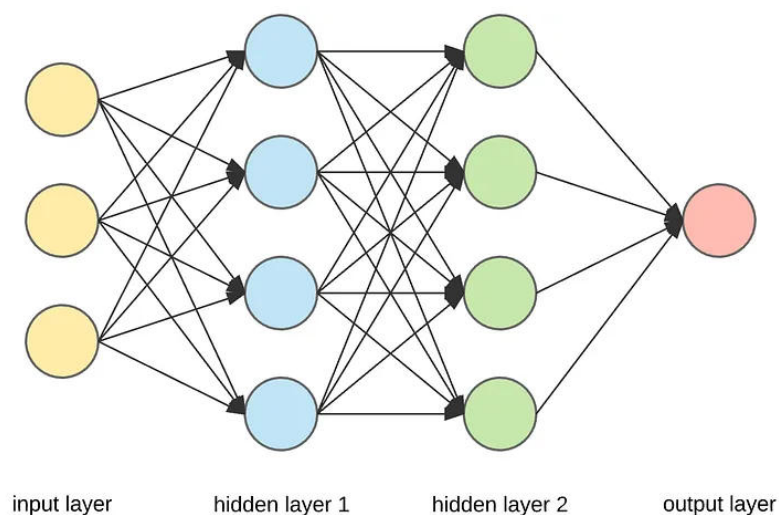


Figure 2.5: Model of neural network [13]

The remarkable progress and achievements in various domains of artificial intelligence can be attributed to the tremendous success and power of deep learning. DL has played a pivotal role in advancements such as email spam detection, stock value forecasting, image recognition, disease diagnosis, and self-driving cars.

These advancements rely on utilizing numerous neural networks, extensive datasets, and the ability to leverage GPUs for accelerated computing. These factors contribute to improved performance across a wide range of tasks, enabling the development of sophisticated AI systems that can tackle complex real-world challenges.

Detection Transformers (DETR)

This deep learning approach simplifies the detection pipeline by eliminating the need for several hand-designed components such as a non-maximum suppression procedure or anchor generation. The key components of this framework consist of a set-based global loss, which enforces unique predictions through bipartite matching, and a transformer encoder-decoder architecture. Notably, the proposed model is conceptually straightforward and does not rely on specialized libraries, distinguishing it from many other modern detectors [14].

MobileNet Single Shot Detector

The MobileNetSSDv2 (MobileNet Single Shot Detector) is an object detection model comprising 267 layers and 15 million parameters. It is specifically designed to facilitate real-time inference on devices with limited computing resources, like smartphones.

The MobileNetSSDv2 model is divided into two main parts. The first component is the base MobileNetV2 network, which incorporates an SSD layer responsible for identifying the objects in the input image. The MobileNet base network serves as a feature extractor, extracting relevant features for the subsequent object classification performed by the SSD layer [15]. Figures 2.6 and 2.7 visually compare the architecture and MobileNet results.

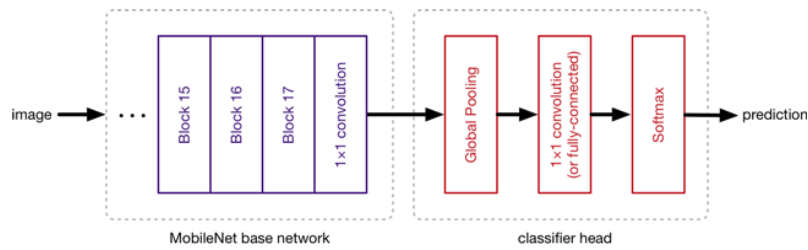


Figure 2.6: MobileNet model architecture [15]

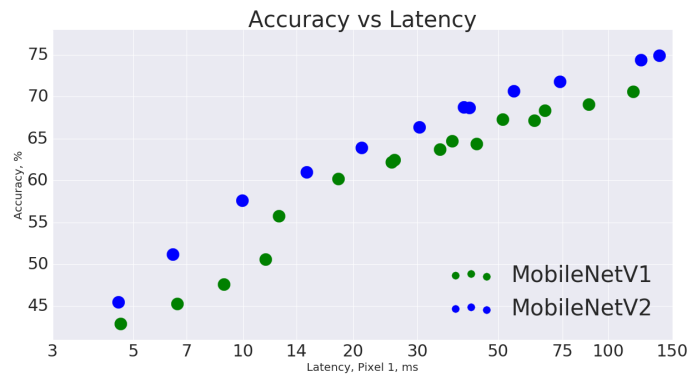


Figure 2.7: MobileNetV1 and MobileNetV2 results comparison [15]

You Only Look Once (YOLO)

One of the most potent real-time object detection systems is YOLO, which stands for "You Only Look Once." Unlike earlier object detection algorithms such as R-CNN or its upgraded version, Faster R-CNN, YOLO only requires a single pass of the image (or video) through its network.

This algorithm simplifies its construction and can be trained directly on entire images. Unlike classifier-based approaches, YOLO is trained using a loss function directly corresponding to detection performance. The model is trained jointly, encompassing object detection and classification aspects [16]. Figure 2.8 depicts the model's architecture.

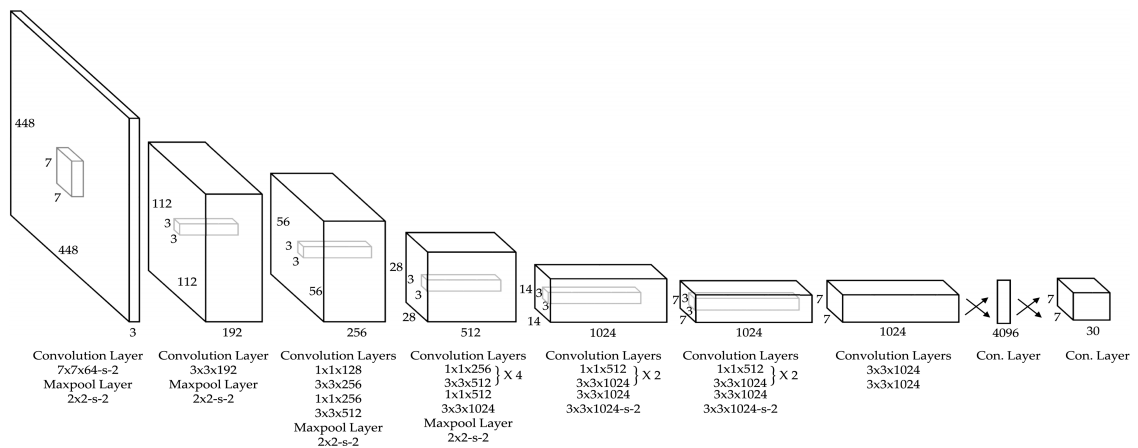


Figure 2.8: Architecture of YOLO model [16]

The YOLO code has undergone several updates throughout the years, progressing from its initial version published in 2016 to the latest release in 2022, known as YOLOv8 [16]. Figure 2.9 illustrates a performance comparison across the different versions. These updates have brought significant improvements to the model.

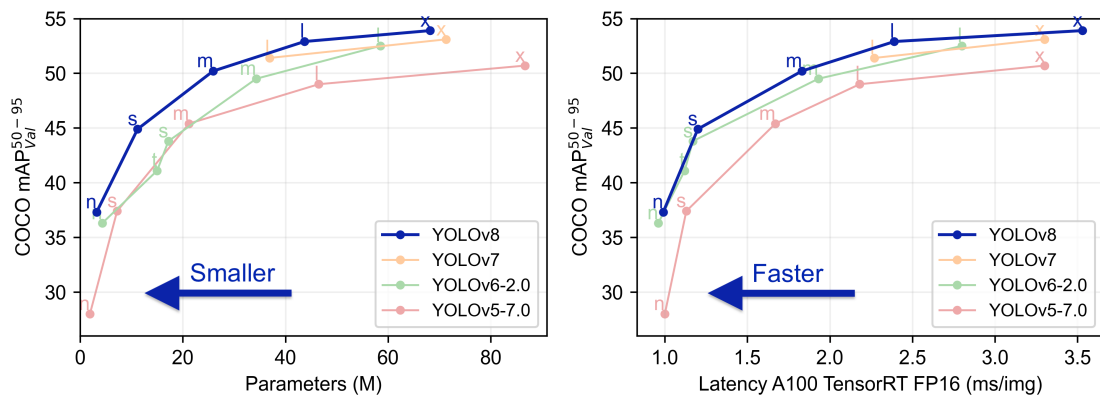


Figure 2.9: Performance comparison of YOLO models [17]

Some of the main advantages of YOLO are [18]:

- Speed;
- Detection accuracy;
- Good generalization;
- Open-source.

2.3.3 Evaluation Metrics

Multiple evaluation metrics can be utilized to assess the quality of an SVM or DL algorithm. In this work, the main metrics employed are as follows:

- Accuracy: It provides the percentage of correctly predicted data points. However, accuracy may need to be improved when dealing with imbalanced datasets.
- Recall: It represents the percentage of actual positive cases correctly predicted.
- Precision: It denotes the proportion of correctly predicted positive instances.
- F1 score: It is the harmonic mean of recall and precision, reaching its maximum value of 1 only when both recall, and precision are 100%. The F1 score is 0 if either recall or precision is zero. In cases where false positives and false negatives have different implications, a generalized version called F_β is suggested.
- Mean Average Precision (mAP): The area under the precision-recall curve is referred to as AP, and the average of all AP values across classes/categories is denoted as mAP.
- Intersection Over Union (IoU): It measures the extent of overlap between two bounding boxes, namely the predicted (PR) and ground-truth (GT) boxes. A higher IoU value indicates a larger area of overlap.

Once the algorithm is tested and trained, these metrics can be calculated using the following equations, where TP represents true positives, FP represents false positives, TN represents true negatives, FN represents false negatives, and 'n' is the number of classes [19, 20, 21, 22].

$$Precision = \frac{TP}{TP + FP} \quad (2.1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.2)$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.3)$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (2.4)$$

$$mAP = \frac{1}{n} \sum_{i=1}^n AP_i; \quad (2.5)$$

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union} = \frac{Area(GT \cap PR)}{Area(GT \cup PR)} \quad (2.6)$$

$$Confusion\ Matrix\ Accuracy = \frac{TP + TN}{TotalSamples} \quad (2.7)$$

2.3.4 Feature Extraction Metrics

As mentioned in sub-section 2.3.1, SVM is a type of machine learning algorithm that is used for classification, regression, and other tasks. SVMs work by finding a hyperplane that separates data points into different classes based on their features. However, SVMs require that the input data be transformed into a numerical feature vector representation before being used for classification

or regression. Here is where descriptor extraction comes in. Descriptor extraction transforms the raw input data into numerical feature vectors that SVMs can use for classification or regression. R, G, and B will be red, green, and blue standards. In table 2.1, there is a summary of the primary visual concepts used in feature extraction in precision agriculture field.

Visual Concept	Vegetation Purpose	Index Type	Plant Segmentation	Main Advantage	Formula
LBP[23][24]	No	GRAY	0	Simple	N/A ^a
HUE[25][26][27]	No	HSV	0	Easy setting threshold	$hue = \arctan(\frac{\sqrt{3}(R-G)}{R+G-2B})$
GLI[28][29]	Yes	RGB	+	Flexibe use	$GLI = \frac{2G-R-B}{2R+G+B}$
RGBVI[30][31]	Yes	RGB	+	Simple	$RGBVI = \frac{(G)^2-(B*R)}{(G)^2+(B*R)}$
NGRDI[32][33][34]	Yes	RGB	+	Overcome the differences in exposure settings	$NGRDI = \frac{G-R}{G+R}$
RI[35][36]	Yes	RGB	0	Simple	$RI = \frac{R-G}{R+G}$
ExG[33][37]	Yes	RGB	+	Good contrast between plants and non-plants	$ExG = 2g - r - b^b$
ExR[33][38]	Yes	RGB	0	Simple	$ExR = 1.3R - G$
ExGR[33][39]	Yes	RGB	+	Simultaneously apply two segmentations	$ExGR = ExG - ExR$
MExG[33][40]	Yes	RGB	++	Robust in changing illumination conditions	$MExG = 1.262G - 0.884R - 0.311B$
CIVE[33][41]	Yes	RGB	0	Great emphasis in green area	$CIVE = 0.441R - 0.811G + 0.385B + 18.78745$
VEG[30][42]	Yes	RGB	+	Robust in changing illumination conditions	$VEG = \frac{G}{R^a(B^{1-a})}^c$

^aHistogram based feature

$$b_r = \frac{R^*}{R^*+G^*+B^*}; g = \frac{G^*}{R^*+G^*+B^*}; b = \frac{B^*}{R^*+G^*+B^*}; R^* = \frac{R}{R_{max}}; G^* = \frac{G}{G_{max}}; B^* = \frac{B}{B_{max}}$$

^c $a = 0.667$

Table 2.1: Visual concept summary comparison table

2.3.5 Deep Learning Object Detection Method vs SVM

Considering various comparative studies of the best technique between DL techniques and SVM for classification and identification problems, it is possible to conclude some essential points to consider for this work.

First, in 2021 a study used Tanzania Sign Language (TSL) datasets to compare the performance of the SVM and the CNN machine learning techniques in translating TSL words spelling. The study concludes that the CNN model performs significantly better in translating TSL than when the SVM model was used [43].

Moreover, two studies were conducted to compare the use of both techniques in car detection. The DL technique chosen was the YOLO. First, Kaplan concluded that deep neural networks with YOLO performed more accurate results than the SVM model. It also performed faster, which makes it more suited for real-time tasks [44]. Moreover, Shan concluded that SVM has an acceptable detection accuracy but has two shortcomings. Due to the heatmap threshold application, the bounding box usually appears unstable and smaller than the actual size of object cars. The processing speed is only up to 2 fps for sliding window search. Even with GPU parallel computing, it is not favorable for real-time applications. YOLO is much preferable because of its strength against SVM's shortcomings. It has better detection, a more stable bounding box position, and a real-time processing speed of nearly 40 fps [45].

Finally, with an approach more similar to the one proposed by this work, an experiment was conducted by Huipeng Li et al. to identify different types of grapes in a vineyard. The investigation shows that the model achieves high precision and speed for identifying different varieties of table grapes and can accurately identify the occluded grapes[46].

2.4 Hardware Theoretical Background

Robotics has made significant advances in recent years, thanks to developments in hardware technologies. Hardware is critical in enabling the physical embodiment and execution of intelligent actions in the context of machine learning-based robotics. The hardware components provide the foundation for the robot's machine-learning capabilities, ranging from solid computers and sensors to specialized actuators and durable mechanical constructions. DOF and image formation are crucial to developing the intelligent sprayer. As this work requires a two-freedom degree robot and a camera, it is necessary to contextualize the theoretical base behind it. Figure 2.10 shows the 3D developed model of the sprayer to contextualize the hardware theory.

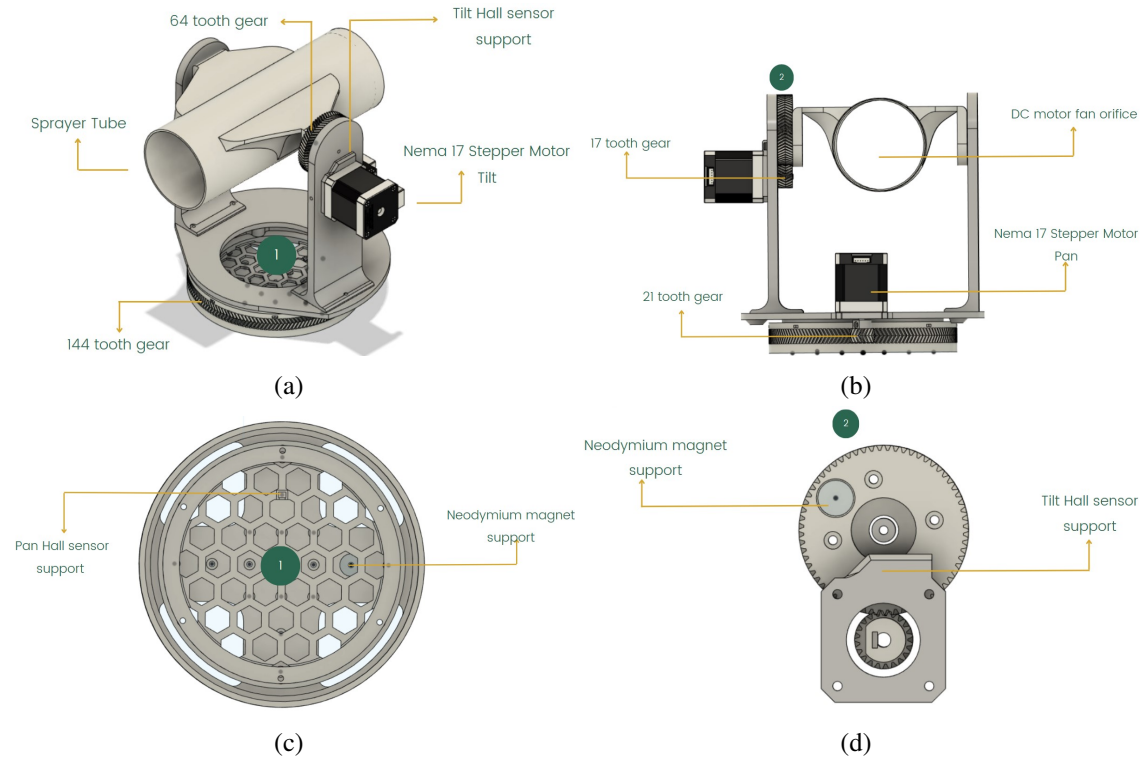


Figure 2.10: 3D structure of the sprayer developed

2.4.1 Degrees of Freedom

The minor independent variable needed to specify the location of a rigid body in space is known as the DOF. Alternatively said, the variety of motions a body may do. A *rigid body* is defined as a mechanism when the $DOF > 0$ [47].

The definition of the term "kinematic connections" is also crucial. Kinematic linkages are components that move motion from one place to another. Joints connect these links. DOF of a kinematic link can be computed using Grubler's rule. This formula states that the number of freedom degrees is determined by subtracting the number of independent constraints from the total of their freedoms.

$$DOF = \sum(\text{freedom of the bodies}) - \text{independent constraints} \quad (2.8)$$

Assuming N is the number of bodies (ground included), J is the number of joints, and m is the constant number that defines the number of degrees of freedom of a body (six for spatial bodies or three for planar bodies), the equation can be rewritten as [48]:

$$DOF = m(N - 1) - \sum_{i=1}^J c_i \quad (2.9)$$

Moreover, knowing the degree of freedom of movement of one link with another can be deduced by subtracting m from the number of constraints that the joint imposes ($f_i = m - c_i$), the

final equation obtained is [48]:

$$DOF = m(N - 1 - J) + \sum_{i=1}^J f_i \quad (2.10)$$

The two main movements to consider are rotation and translation.

The translation can be subdivided into three movements: Forward and Backward, Right and Left, and Up and Down. Moreover, the rotation can be subdivided into three movements: Pitch, Yaw, and Roll. Figure 2.11 exemplifies the six-freedom degree movement.

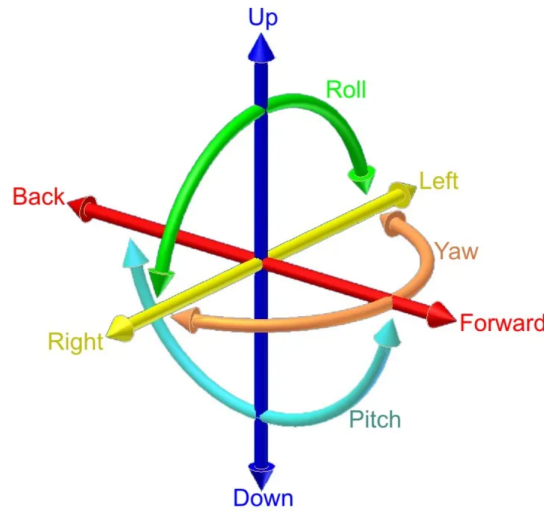


Figure 2.11: Six freedom degree diagram [49]

Considering the theoretical approaches, the next chapter will analyze the best solution to develop a two-freedom degree sprayer. As shown in Figure 2.2, the last spray hardware built was based on a fixed drum. The roll and pitch movements could significantly improve the drum base, allowing a better coverage area. However, a spray coupled to a two freedom degree robot arm would grant access to more dense and higher foliage.

2.4.2 Coordinate Systems

Coordinate systems are critical in robotics because they allow for exact positioning and movement control of robotic devices in three-dimensional (3D) space. This section overviews standard coordinate systems in robotics and their applications in diverse automated tasks.

The Cartesian coordinate system is frequently used in robotics as a fundamental framework for describing locations and orientations in 3D space. It defines a point's site relative to an origin using three perpendicular axes (x, y, and z). Cartesian coordinates provide more precise control of robot manipulators and serve as a handy reference for explaining spatial relationships.

Robotic manipulators frequently use joint space to coordinate motion planning and control. The positions of a robot's separate joints are utilized to establish its configuration in the joint area.

Because it is intimately related to the actual structure of the robot, this representation simplifies tasks like trajectory planning, inverse kinematics, and collision avoidance.

Task space, commonly known as Cartesian space, is the coordinate system based on the location and orientation of the end effector. It enables robot movements to be specified regarding desired positions and orientations in 3D space. Task space coordinates are very helpful for establishing tasks that involve objects in the robot's workspace or when interacting with the environment.

The world coordinate system represents the global reference frame within which the robot operates. It establishes the robot's relationship with its surroundings. This coordinate system is frequently formed by selecting a fixed point or a specific object as the origin, and the axes are aligned with particular directions, such as the gravity vector or an environment reference coordinate system[50][51].

In the robot base coordinate system, the coordinate frame tied to the robot's base or any other preset reference point on the robot is called the robot base coordinate system. It serves as a local reference for the robot's movement and can be utilized as an intermediate coordinate system for transformations between other frames of reference[50].

Finally, the tool coordinate system defines a local reference frame tied to the robot's end effector or tool. It enables exact positioning and orientations relative to the gripper or instrument of the robot, which is critical for operations such as grasping, manipulation, and assembly[50][51].

2.4.3 Robotic Kinematics

A rotation matrix (R) describes the relative orientation of two such frames. The columns of this 3x3 matrix are the unit vectors along the axes of one reference frame relative to the other. There are two methods to understand rotation matrices. First, as a representation of the rotation of the first frame into the second (active interpretation) or of the mutual orientation of two coordinate systems (passive interpretation) [52] [53]. The following matrix represents the rotation matrix in each axis.

$$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(a) Rotation around z-axis

$$\begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$$

(b) Rotation around y-axis

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$$

(c) Rotation around x-axis

The following examples can represent the homogeneous matrix (H) of three-dimensional space by adding the translation part in the last column and then padding the matrix. In reality, the matrix must be adjusted to each problem specification [54].

$\begin{bmatrix} \cos \theta & -\sin \theta & 0 & Px \\ \sin \theta & \cos \theta & 0 & Py \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & Px \\ 0 & 1 & 0 & Py \\ 0 & 0 & 1 & Pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
(a) Translation in xy-plane with rotation in z-axis	(b) Translation in xyz without rotation	(c) rotation on xyz without translation

Figure 2.13: Matrix examples

Finally, applying the following equation to find a point's (n,o, a) position relative to your current coordinate frame is necessary.

$$P^{xy} = H \cdot P^{noa} \quad (2.11)$$

$$P^{xy} = \begin{bmatrix} n_x & o_x & a_x & Px \\ n_y & o_y & a_y & Py \\ n_z & o_z & a_z & Pz \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} P^n \\ P^o \\ P^a \\ 1 \end{bmatrix} \quad (2.12)$$

2.4.4 Image Formation

An important concept when choosing the camera hardware is the image formation theory and how to select a camera correctly. Today, many companies produce cameras to be used in computer vision applications. Furthermore, these companies also provide a native computer vision software library for the camera. Besides this facilitating factor, it is vital to highlight the basic concepts of the image formation problem.

First, the light spectrum we see is a continuous mixture of wavelengths altered by reflectance and absorption. The eye responds to the spectrum by producing a tristimulus, a three-vector representing the outputs of the three distinct types of cones in the human eye, which we interpret as color. Functionally, a digital color camera is similar. [55]

Moreover, a 2-dimensional surface is used to project the 3-dimensional world in an eye or a camera during the picture formation process. We can no longer determine from the image whether it is of a prominent object in the distance or a smaller, closer object because the depth information has been lost. Perspective projection is the term for converting three dimensions to two, which is why camera adjustment is required. It is also feasible to solve the opposite issue, which involves reconstructing 3-dimensional world points from a 2-dimensional image [55].

Furthermore, the hardware selection surpasses the theoretical analysis. It is necessary to study and analyze the project demands, the budget, external factors, the environment, and resolution requirements.

2.4.5 Hall Effect

Suppose an electric current passes through a conductor in a magnetic field. In that case, the magnetic field produces a transverse force on the moving charge carriers, pushing them to one side of the conductor. This effect is visible in a thin flat conductor, as shown in Figure 2.14.

A charge accumulation on the conductor's sides will balance this magnetic force, resulting in a detectable voltage between the two sides of the conductor. This voltage is named Hall voltage. This observable transverse voltage is known as the Hall effect, after its discoverer, E. H. Hall, in 1879 [56].

The following equation gives the Hall voltage:

$$V_H = \frac{I \cdot B}{ned} [56] \quad (2.13)$$

where n represents density of mobile charges, e is electron charge, I the current, B the magnetic field and d the conductor thickness.

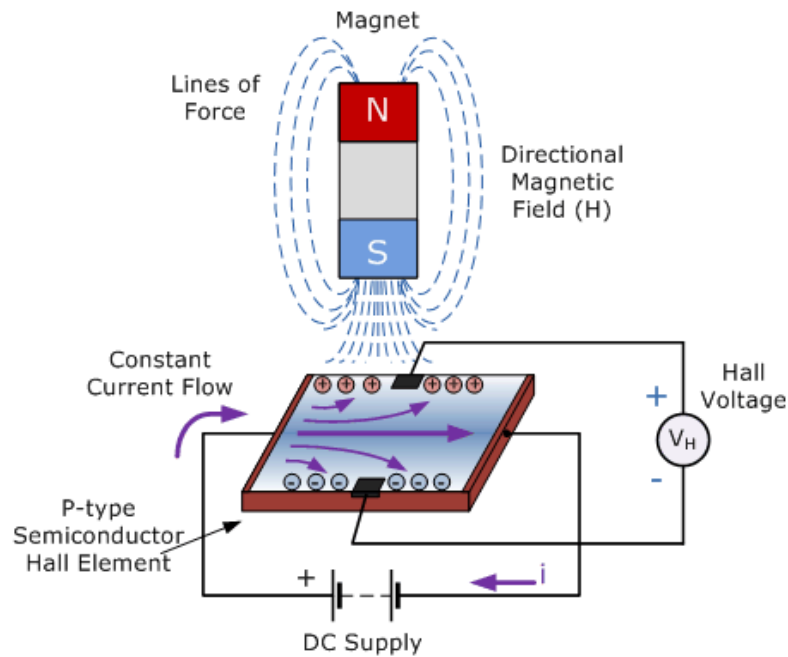


Figure 2.14: Hall effect theoretical diagram [57]

Hall effect sensors have several applications. The most common applications are position sensing, direct-current (DC) transformers, automotive fuel-level indicators, and keyboard switches. Hall effect sensors do not utilize a coil and generate a digital signal instead of an analog AC signal. To this work, positional sensing is particularly interesting to work with stepper motors. Figure 2.15 shows an example of a hall sensor application to measure a crankshaft's position. The digital output wave will be high when the shaft teeth pass into the sensor probe. In the opposite situation, the wave value will be low. In short, as the teeth pass the sensor, it turns on or off.

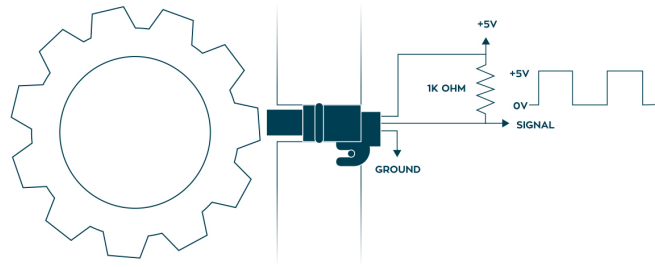


Figure 2.15: Hall effect position sensor example [58]

2.5 Conclusion

This section introduced several methods for extracting vegetation features from an image. All approaches have advantages and disadvantages for detecting foliage. Previous studies have shown that combining some strategies often produces positive results in canopy detection.

Analyzing the points presented in the subsections 2.3.1 and 2.3.2, comparing and concluding several advantages and disadvantages of both methods is possible.

In conclusion, while DL methods like CNN can automatically extract pertinent features from the input data, SVM needs manual feature extraction. SVM can be quicker to train and require less data than DL models, making them helpful when resources or data are limited.

On the other hand, DL techniques have several benefits over SVM. First, they can pick up on more intricate characteristics that might be challenging to extract by hand, like crucial textures or patterns. Also, they can learn to represent data more solidly and consistently, which will help them perform better when the incoming data is noisy or variable. Last but not least, DL models can frequently be trained end-to-end, meaning the stages of feature extraction and classification are learned simultaneously. Doing this process at the same time improves the effectiveness and efficiency of the training process. DL models can, however, also be more computationally demanding and need more data to train successfully.

In addition, it was presented the hardware mechanical concepts for the intelligent spray. The DOF's possible movements were discussed to validate the improvement possibilities for the robot spray. Moreover, the image formation theory helps with the visual hardware choice. Furthermore, the electromagnetic concepts of the hall effect are crucial factors when working with hardware that requires positional measurement. Also, the idea of robotic kinematics was studied to comprehend the different coordinate frames used in this work. The camera frame is responsible for acquiring the coordinates of the detected objects, and the sprayer frame is responsible for pointing the mechanism in the coordinate direction.

Chapter 3

System Architecture and Requirements

This chapter presents the proposed solution to be developed in the subsequent chapters. The solution, entitled 'Grape Detector Sprayer' (GraDeS), considers previous works to offer a viable and enhanced solution. The primary objective of this work is to upgrade and propose a novel approach for handling grape bunch spraying.

The continuous advancements in superior computer vision and deep learning systems have opened up new opportunities across various sectors. Through the study conducted in Section 2.3, it can be inferred that within the context of vineyard spraying, the utilization of computer vision techniques such as OpenCV, combined with deep learning algorithms like YOLO (You Only Look Once), can significantly enhance the efficiency and precision of grape bunch recognition and spray targeting. This chapter elucidates the system requirements and architecture for the 'GraDeS' solution. It employs computer vision and deep learning methodologies to identify grape bunches within a vineyard and guide a sprayer toward its location.

3.1 System Architecture

The system architecture delineates the organization and flow of components in the grape bunch recognition and spray targeting project, encompassing the integration of hardware, software, and data flow. It establishes a symbiotic relationship among three key elements: the real world, hardware, and software.

The real-world block represents the system's working environment. The OAK-1 camera captures real-time video footage of the vineyard, serving as the input for the grape bunch recognition algorithm. The camera's RGB sensors enable precise perception, facilitating accurate detection and localization of grape bunches.

The inference block processes the captured video frames through preprocessing using OpenCV and the native OAK library, 'depth.' The preprocessing includes applying a red mask to segment the laser color. The preprocessed video frames are fed into the deep learning model and implemented using a suitable deep learning framework. Trained on a dataset of grape bunch images, the

algorithm identifies and localizes grape bunches within each frame. Recognition results, comprising bounding box coordinates and bunch IDs, are generated for each identified grape bunch and sent to the hardware in the send data block.

Within the hardware block, the grape bunch recognition results are utilized to determine the direction and location of the bunches in the vineyard. An integrated sprayer interface receives guidance regarding the detected grape bunch centers and the bunch IDs. Utilizing this information, the control code adjusts the sprayer's direction with a pan and tilt movement to target the identified grape bunches simultaneously.

Subsequently, once the mechanism is directed towards the grape bunches, the sprayer nozzle and fan are activated to dispense liquid onto the targeted bunches. This process is then repeated for subsequent grape bunches.

Figure 3.1 illustrates the relationship among the main modules, while Figure 3.2 visually depicts the workflow of the system.

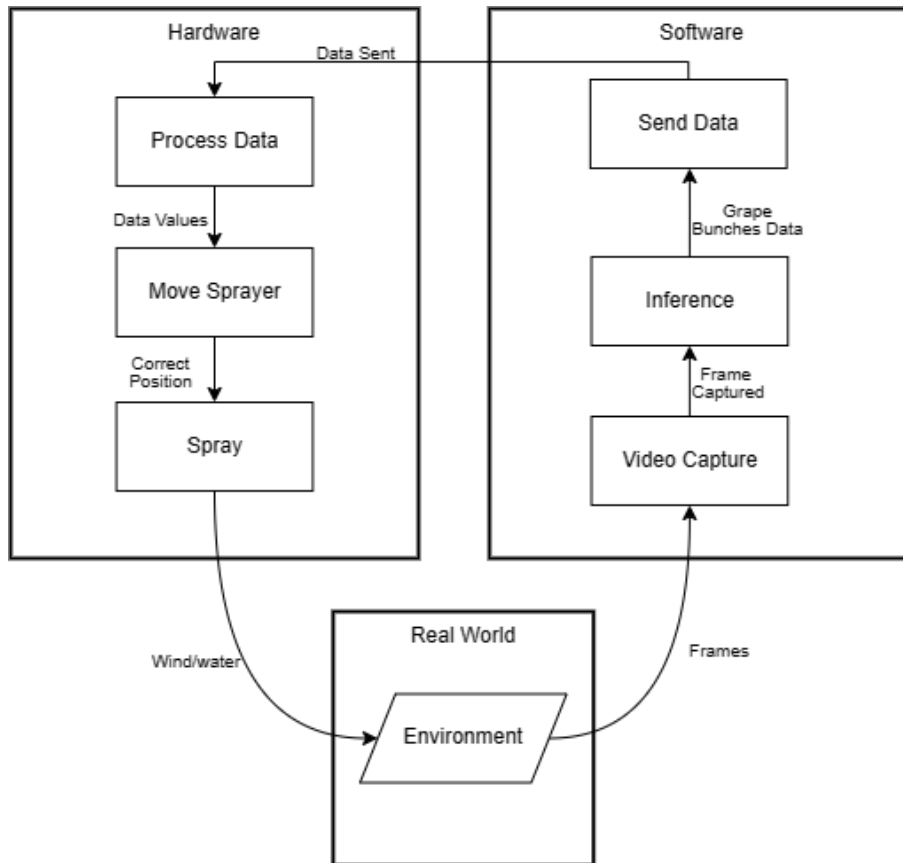


Figure 3.1: GraDes architecture relationship

3.2 System Requirements

The success of the grape bunch recognition and spray targeting project depends on achieving specified hardware and software requirements for efficient processing and real-time performance.

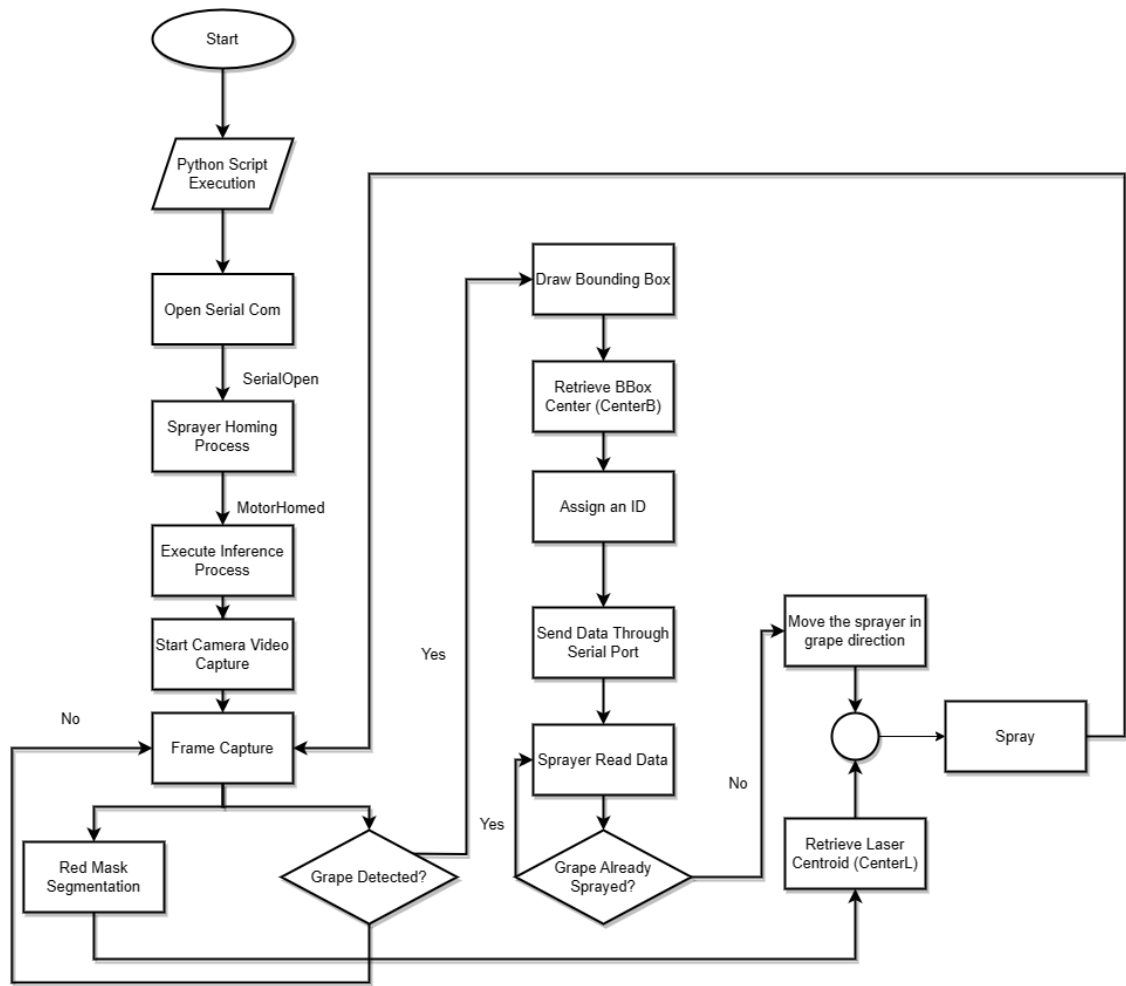


Figure 3.2: GraDes system workflow

The system must meet the following requirements:

3.2.1 Hardware Requirements

The system necessitates hardware capable of capturing real-time images and conducting the inference process. Employing a device with ample processing power is essential, as deep learning methods such as YOLO can be computationally demanding. A computer or embedded device with significant processing power must be utilized for inference processing to achieve real-time performance during grape bunch detection and spray guidance.

Additionally, a sprayer mechanism is required to interpret the data obtained from the camera and orient the tool in the direction of the grape bunches. To execute the sprayer control algorithm, a microprocessor with serial communication capabilities is essential for communication between the camera and the sprayer.

The sprayer system encompasses two stepper motors, each corresponding to a specific movement axis (pan and tilt). This architectural design is inspired by contemporary pan and tilt pho-

tography devices. Furthermore, the mechanism incorporates two hall sensors and two magnets to establish the initial positions of the motors. To execute the spraying action, a system comprising a spraying nozzle and a fan is employed to dispense the liquid. Finally, all of these components are assembled within a 3D-printed housing.

3.2.2 Software Requirements

The system necessitates the utilization of an open-source library for image-processing tasks. OpenCV, a renowned open-source computer vision library, is a fundamental component of the project. It provides a comprehensive set of image and video processing methods, including image identification, object detection, and segmentation. These capabilities are pivotal in the vineyard context for grape bunch recognition.

Moreover, an efficient detection algorithm is essential to differentiate grape bunches within the vineyard. Deep learning algorithms such as YOLO, SSD MobileNet, and DTR can be implemented using deep learning frameworks compatible with both C++ and Python. Popular frameworks such as TensorFlow, PyTorch, or Darknet can be employed for training and deploying the grape bunch recognition model.

The project integrates the C++ and Python programming languages to leverage C++'s performance and versatility for computationally intensive tasks while benefiting from Python's ease of development and prototyping.

Additionally, the system may require an interface to facilitate communication with the sprayer, enabling real-time guidance based on the results obtained from grape bunch recognition. The interface can be established through serial port communication between the microprocessor and the camera.

3.3 Conclusion

The system requirements and architecture lay the groundwork for the project's effective grape bunch recognition and spray targeting implementation. The system may achieve real-time performance and accurate grape bunch recognition by ensuring that the hardware and software components satisfy the necessary criteria and building an efficient and well-structured architecture. It allows for precision sprayer targeting in the vineyard.

Chapter 4

Deep Learning Models for Grape Bunch Detection

This chapter presents the software development objectives, tools, methodologies, and resources utilized in conceiving the GraDes solution. Firstly, a summary of the main goals of this project phase is provided. The subsequent sections delve into the methodology, explaining it in detail and offering a brief justification for selecting the employed tools, emphasizing their advantages. Furthermore, the algorithms, solutions, and results developed during this endeavor are presented in the subsequent sections.

The primary aim of this phase was to develop a perception algorithm capable of identifying and detecting grape bunches in a vineyard. Once the bunches are located, the code must instruct the hardware to orient the spray towards the grape bunches. Additionally, the hardware should direct a laser at the fruit. The code must then segment the laser color from the image to verify the position of the bunches. Upon completing these steps, the software should solely detect the grape bunches.

4.1 Tools and Methodology

A series of tools were used during the work process to develop this work. The main tools and software used were:

- Ubuntu;
- Visual Studio Code;
- Google Colab;
- Python and Anaconda;
- C++ and CMake;
- OpenCV;

- YOLO and PyTorch;
- Git and Github;
- DephtAI;
- CUDA and cuDnn;

Firstly, Ubuntu, Git/GitHub, and Visual Studio Code were chosen due to their familiarity and seamless integration with all aspects of the project. Python was selected for its extensive machine learning and data science library collection. Utilizing a virtual environment based on the Anaconda distribution helps enhance performance and facilitates package management. While Python allows for straightforward coding, its compilation performance is suboptimal. Hence, C++ is a preferable option for real-time robotics systems. Additionally, Luxonis camera hardware provides a native library for computer vision.

Regarding the conclusions presented in Chapter 2, particularly in Section 2.5, the DL (Deep Learning) method was chosen over the SVM (Support Vector Machine) technique due to the project's computational limitations, which are not strictly constrained. The solution deployment must be done on a mobile platform without a GPU. Hence, YOLO (You Only Look Once) was selected, as it supports various model sizes, which improve frames per second (FPS) and processing time. Moreover, given the substantial size of the dataset, manually annotating the training dataset would be unnecessary and unreliable.

The chosen DL technique for this work was YOLOv5 and YOLOv8. Comparing YOLOv5 with newer versions, the "mAP" (mean Average Precision) and real-time performance are slightly inferior to the YOLOv8 model. However, due to its older age, YOLOv5 has more extensive documentation, examples, and support. Since this work does not involve critical real-time systems like autonomous driving, the slight real-time performance differences between the two methods are not detrimental. In summary, both YOLOv5 and YOLOv8 are viable solutions for reliable object detection of small items. However, YOLOv5 may be better for deploying the solution on devices lacking GPU capabilities. Tests were conducted to determine which solution better fulfilled these assumptions.

Furthermore, the conducted tests demonstrated that the model size directly impacts inference performance. Consequently, the DETR (DEtection TRansformer) model was deemed unsuitable for this work's application. The official DETR Github repository shows that the model's size ranges from 150MB to 240MB, making it impractical for the mobile application. Additionally, as shown in Section 4.3, the SSD (Single Shot MultiBox Detector) Mobilenetv2 model yielded worse results than the YOLO model. Thus, it was also discarded.

Training and testing large datasets can be time-consuming, often taking hours or days. The specifications of the local machine directly influence the performance of training time. In this case, due to the local machine having only 8GB of RAM and a GPU with 4GB of RAM, the model was trained using the Google Colaboratory platform. This platform offers free access to a virtual machine hosted on Google servers, providing 13GB of RAM and a GPU with 15GB of

RAM, significantly more powerful than the local machine. Having a high-performance graphics processing unit (GPU) is advantageous. While central processing units (CPUs) excel at performing numerous general tasks quickly and sequentially, GPUs utilize parallel computing to handle massively complex problems through simultaneous calculations efficiently. This advantage makes them ideal for the demanding computational processes required in machine learning. Additionally, deep learning calculations often involve processing massive volumes of data, emphasizing the importance of a GPU's memory bandwidth.

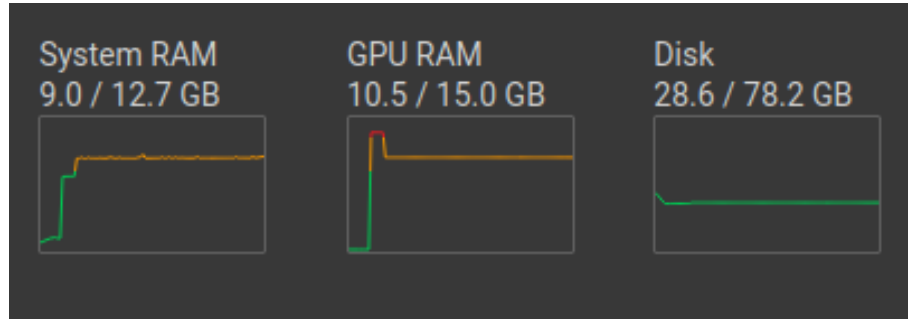


Figure 4.1: RAM usage during the training process

As shown in Fig. 4.1, the GPU RAM values achieve almost the maximum values that Google Colab allows. These values would never be possible in the actual local machine.

Unlike the DL training part, for the segmentation and video capture parts, the exigence o memory is not huge. So the local machine served well for all the requirements. The main problem while processing the neural networks in real-time video is the poor frame per second (FPS) rate. Moreover, to make the GraDeS solution portable, it was necessary to use a smaller YOLO model to fit in the processing power of the hardware device or a smaller image size. The computation of the FPS follows the algorithm 4.1:

Algorithm 4.1 FPS calculation algorithm

```

start ← TickCount
for  $i \leftarrow 0$  to TotalDetections do
    InferenceProcess
end for
end ← TickCount
TotalTime =  $\frac{\text{end} - \text{start}}{\text{TickFrequency}}$ 
FPS =  $\frac{1}{\text{TotalTime}}$ 

```

A small test was conducted for ten seconds with different YOLO models, and it was possible to retrieve the total inference time average and the FPS average. As seen in table 4.1 and in Fig. 4.2, the improvement of the FPS was significant when downsizing the image or model size. This improvement is essential when dealing with real-time image processing. In short, using a smaller model or image size raises the FPS performance on the device.

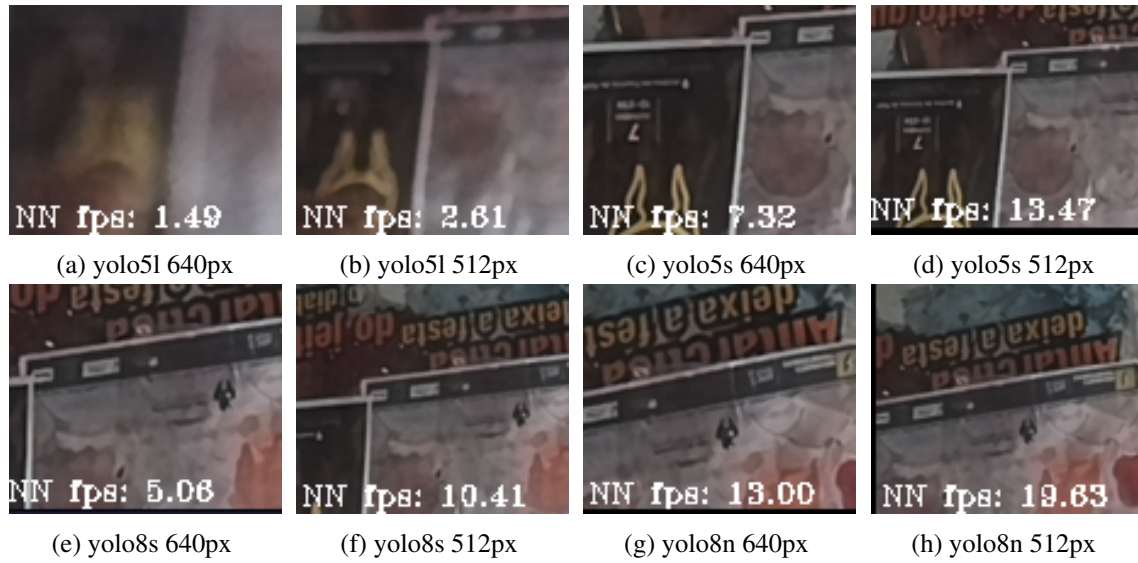


Figure 4.2: FPS performance tests in the mobile device for different image sizes and models

	yolov5l.pt	yolov5s.pt	yolov8s.pt	yolov8n.pt
640 pixels	2.7	6.4	4.7	12.6
512 pixels	1.3	12.3	9.0	18.6

Table 4.1: FPS Performance comparison in the mobile device for different image sizes and models

4.2 Dataset and Deep Learning Models Training.

The dataset used for the CNN training was a mix of several datasets studied [59]. The dataset is open source and can be found on the Roboflow website. The images were separated adequately into three classes: train set, validation set, and test set. Amongst the photos are a folder with the correct class annotations and bounding box position in YOLOv5 format.

As shown in Fig. 4.3, the main advantage of using this set of images is the excellent variety of grape bunch images. The set contains purple, white, and black grapes. Moreover, the dataset presents pictures of tiny, medium, close, far, covered bunches, and different resolution images. The most common type of vineyard in Portugal is the one presented in Subfig. 4.3d. With this mindset, it was appended in the previous dataset, another one only with this variety of vines [60]. Performing this addition helps to improve the final results of recall and precision. Furthermore, the dataset is medium size, so the training process is more time efficient considering the RAM used to create a cache for the images. The train set contains 6935 images, the validation set contains 977, and the test set 128.

In the previously studied datasets, the main problem found was the singularity of images. The model did not recognize green grape bunches when training only with purple grape bunches (as evident). When training with partially covered grapes bunches, the model needed help identifying whole bunches. The inverse was also checked; a dataset with only whole grape bunches failed

to recognize partially covered fruits. Therefore, the best solution was to choose a set with mixed types of images. As seen in Section 4.3, the results with this approach proved efficient.

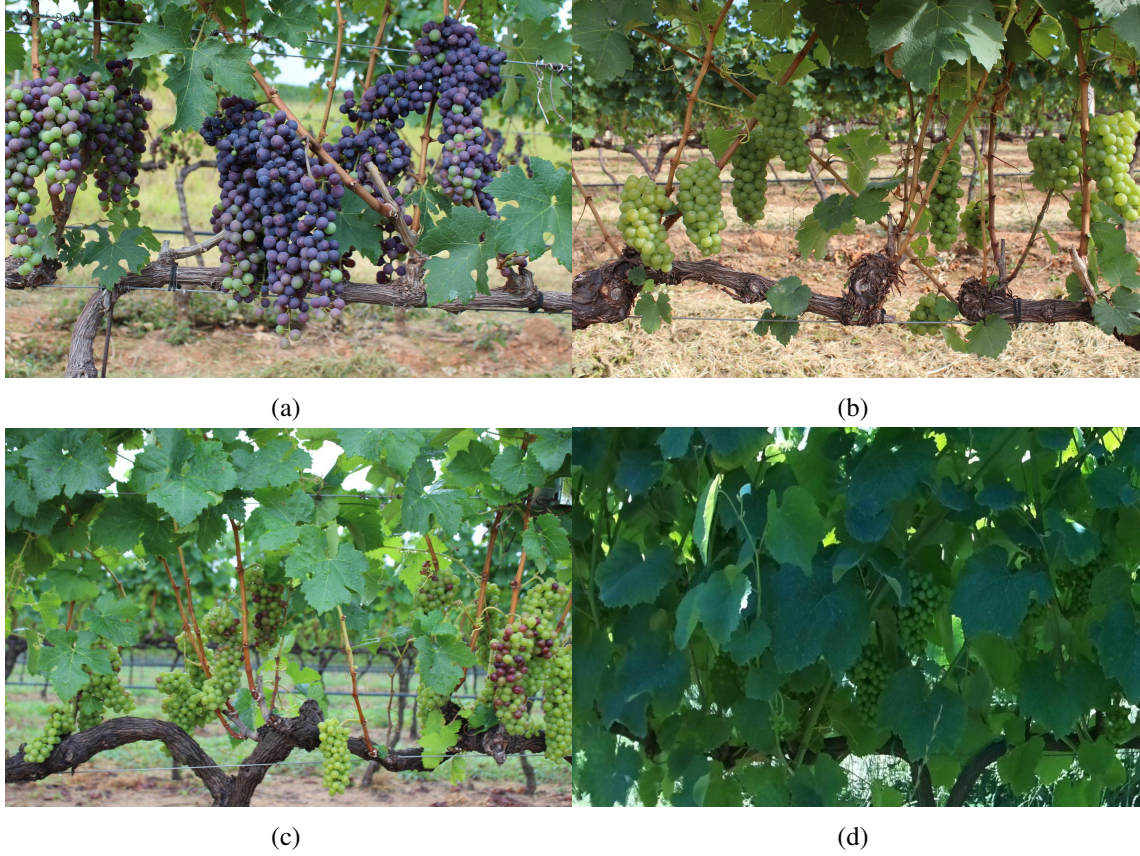


Figure 4.3: Training set image examples

4.2.1 Grape Bunches Recognition

The YOLOv5 official GitHub page provides the notebooks templates used [61] and YOLOv8 official GitHub page [62]. Due to the limit of GPU RAM usage of the free account of Google Colab, the training was done in parts using the training set checkpoints. The dataset reference can be found in Section 4.2. Table 4.2 shows the parameters used in each training. Due to the necessity of deploying in mobile solutions, the model "yolov8n.pt" was chosen based on the FPS performing comparison presented in table 4.1 and the results in Section 4.3.

Batch_size	Epochs	Img_size	Model	Fine Tuning	Freeze
64	150	640	yolov5l.pt	True	False
64	150	640	yolov5s.pt	True	False
32	150	640	yolov8s.pt	N/A	N/A
32	150	640	yolov8n.pt	N/A	N/A

Table 4.2: Preliminary training parameters used in YOLO training

4.2.2 YOLOv8 pre trained model

The used architecture is a pre-trained model in the source code of Ultralytics GitHub page [17]. As mentioned before, the model "n" was chosen because of the necessity of good performance in a mobile solution. Table 4.3 shows the difference between the model sizes. As shown, the size has significant differences that could affect the performance of the inference process. Section 4.3 offers that, besides the size difference, the "mAP," the recall, and precision are not so different.

Model	Size (MB)
yolov5l.pt	93.1
yolov5s.pt	56.7
yolov8s.pt	44.7
yolov8n.pt	6.2

Table 4.3: Different model sizes after 150 epoch training

Fig. 4.5 shows the speed performance of the different models in YOLOv5, which is analogous to YOLOv8. Additionally, Fig. 4.4 shows the architecture of the model divided into two parts:

- Backbone is a DL architecture that functions as a feature extractor. In other words, it is a CNN that accumulates and produces visual features at various levels of granularity. All of the backbone models are classification models.
- Head is also called the object detector since it discovers the region where the item could be present but does not specify which object is in that region. Two-stage and one-stage detectors are available, with the latter further classified into anchor-based and anchor-free detectors. In other words, it consumes features and performs box and class prediction steps.

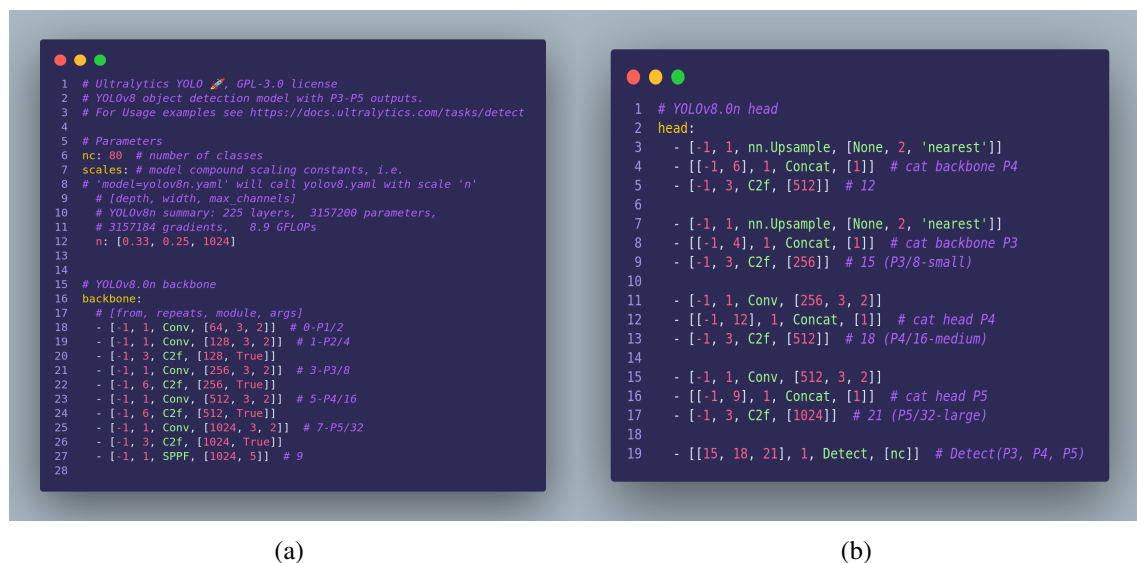


Figure 4.4: YOLOv8n.pt model architecture

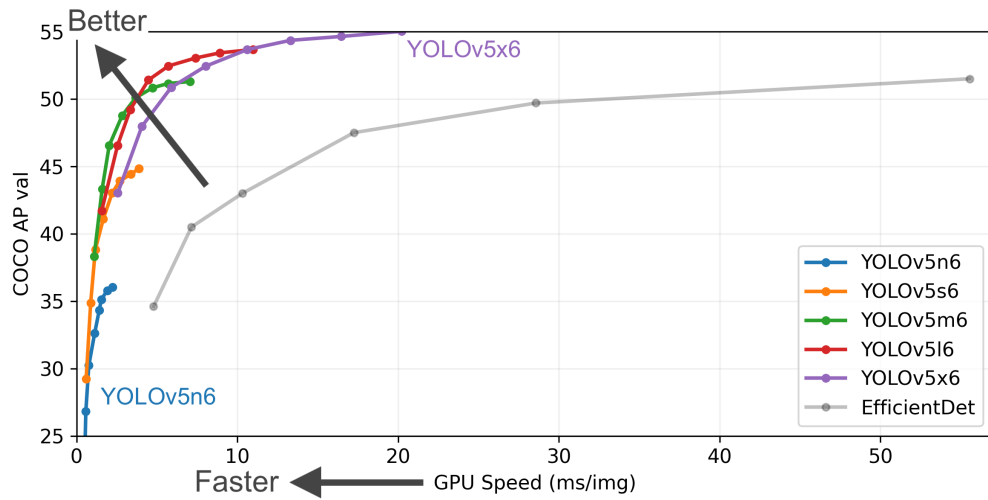


Figure 4.5: Performance of different YOLOv5 models[17]

4.2.3 Labels

The YOLO training algorithm performs the feature extraction based on previous annotations labeled in the training set. The labels provide the algorithm information about the class and the position of the bounding box in the image. As mentioned before, each picture is associated with a label file. The validation process occurs by comparing the labeled image with the predicted image. Fig. 4.6 shows the example of labeled images.

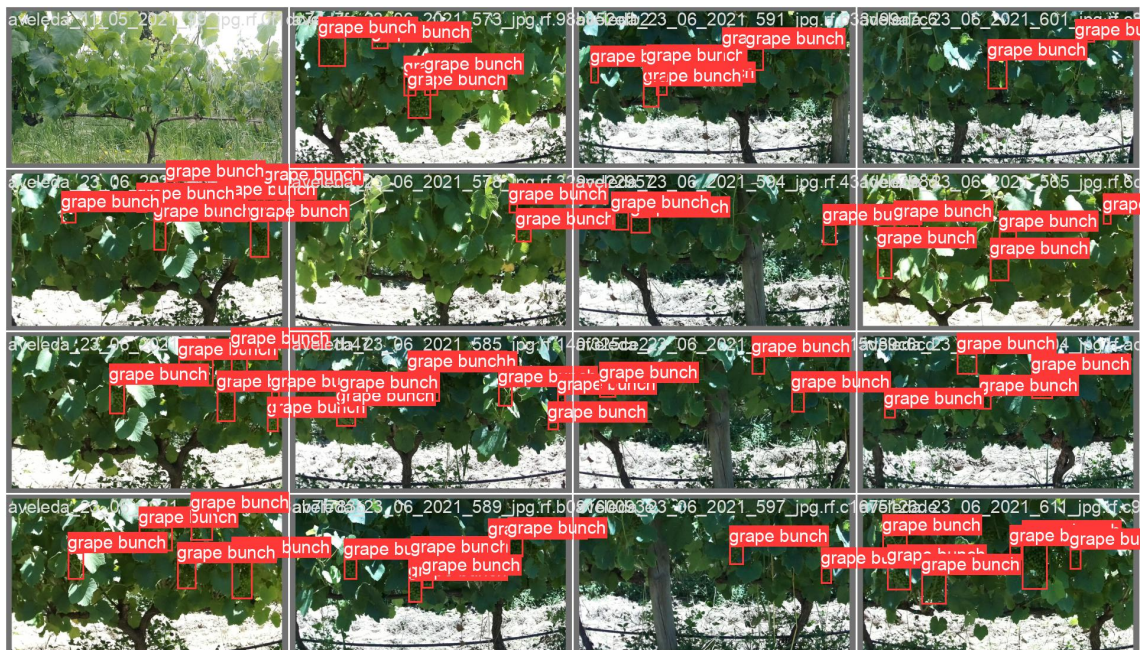


Figure 4.6: Labeled images in YOLO

A red bounding box surrounds each feature, indicating the exact position of the desired attribute. The label format presents the class ID (in this case, there is only one class) and the

corresponding coordinates of the box points. For example, a grape bunch in the left upper corner of a 300x300 image should have a label format, hypothetically, as "0 30 30 50 50", representing the item class, xmin, ymin, xmax, and ymax, respectively.

4.2.4 Camera Integration

Running the inference model using the camera hardware present in the computer (such as the built-in webcam) or in external cameras (such as OAK-1) is a relatively simple task. Due to the big OpenCV and Depthai libraries support, several tools help to perform this integration.

The OpenCV method, without the native library of OAK-1, follows the algorithm 4.2. This method can use as well with the webcam, as images or external devices, but requires a CPU.

Algorithm 4.2 OpenCV Inference Process

```

modelPath ← ModelOnnxPath
runOnGpu ← DesiredHardware
Class Inference ← modelPath, imageSize, basePath, classFile,
runOnGpu
cap ← videoCapture
while true do
    cap ← frame
    for  $i \leftarrow 0$  to TotalDetections do
        BoundingBoxDraw
    end for
    ShowFrame
end while

```

Since "depthai.h" is a native library of Luxonis OAK cameras, this method only works with devices OAK devices. It has a lower processing capacity but does not require a CPU; the AI inference runs inside the device.

Algorithm 4.3 Depthai Inference Process

```

modelPath ← ModelBlobPath
camRgb ← pipelineCreateColorCamera
detector ← pipelineCreateYOLODetector
xoutRgb ← pipelineCreateXlinkOutRgb
nnOut ← pipelineCreateColorXlinkOutNN
camRgb ← setProperties
detector ← networkSettings
deviceConnection
for  $i \leftarrow 0$  to TotalDetections do
    BoundingBoxDraw
end for
while true do
    ShowFrame
end while

```

4.2.5 Laser Segmentation

For laser color segmentation, simple techniques of color segmentation were used in image processing. The laser hardware emits an infrared beam toward the grape bunch. When the spray receives the command to point in the bunch direction, the laser color must be segmented to confirm the position. The mask value used in the HSV spectrum was from min = (146, 175, 50) to max = (180, 255, 255). The values can be versatile since red stands out from the green background.

Algorithm 4.4 Laser Color Segmentation

```

cap ← videoCapture
while true do
  frame ← videoCapture
  frame ← frameHSV
  frame ← mask
  point ← momentsCentroid
  frame ← point
end while

```

4.3 Results

First of all, the test and validation were performed with images. In short, the results obtained were satisfactory. The following images illustrate the analysis of the results. With the interpretation of Fig. 4.7 and comparing it with Fig. 4.6, it is possible to observe that the model predicts most of the features correctly. Moreover, it is possible to notice that the incidence of true positives is higher in images with low resolution and low illumination conditions. This problem happens due to the color similarity of green grape bunches with shadows and leaves.

The different types of models retrieved different results. As mentioned before, the inference process in the device requires a model with a small size and good performance in the detection. First, tests with the dataset were performed before appending more images to the dataset. The results of table 4.4 show slightly better results to model "5s". However, due to being more recent, the project's lifetime support with version "8s" would be more significant.

	mAP	Recall(R)	Precision(P)	F1_Score	PR mAP@50
yolov5l.pt	0.72	0.82 (at 0)	1 (at 0.984)	0.71 (at 0.517)	0.710
yolov5s.pt	0.74	0.90 (at 0)	1 (at 0.902)	0.73 (at 0.370)	0.758
yolov8s.pt	0.74	0.86 (at 0)	1 (at 0.933)	0.74 (at 0.431)	0.756

Table 4.4: Models performance before appending more images to the dataset

With the decision to use model eight versions, more image data was appended to the dataset (explained in Section 4.2) to raise the performance. Moreover, the model downsizes of the "s" model to the "n" model was also studied to see the cost-benefit of performing this operation. Table 4.5 shows the study results.



Figure 4.7: Prediction image after validation step

	mAP	Recall(R)	Precision(P)	F1_Score	PR mAP@50
yolov8s.pt	0.81	0.86 (at 0)	1 (at 0.962)	0.81 (at 0.491)	0.824
yolov8n.pt	0.79	0.90 (at 0)	1 (at 0.931)	0.78 (at 0.424)	0.810

Table 4.5: Models performance after appending more images to the dataset

With very similar performance results, and considering the FPS performance results in table 4.1, the chosen model, as said before, was the "8n" option.

This model choice modified the training to 300 epochs with early stopping with training results presented in table 4.6. The results in images with good illumination conditions are almost perfect. Fig. 4.8 shows six pictures retrieved from the detection test of YOLOv8. It is possible to see that the results are outstanding since the bunches in the images were correctly detected without any false positives.

Moreover, the subsequent figures show the result in chart format. Fig. 4.9 shows the evolution of the training process over the epochs. Furthermore, Fig. 4.10 show the results of table 4.4 and 4.5 with the x-axis representing the confidence.

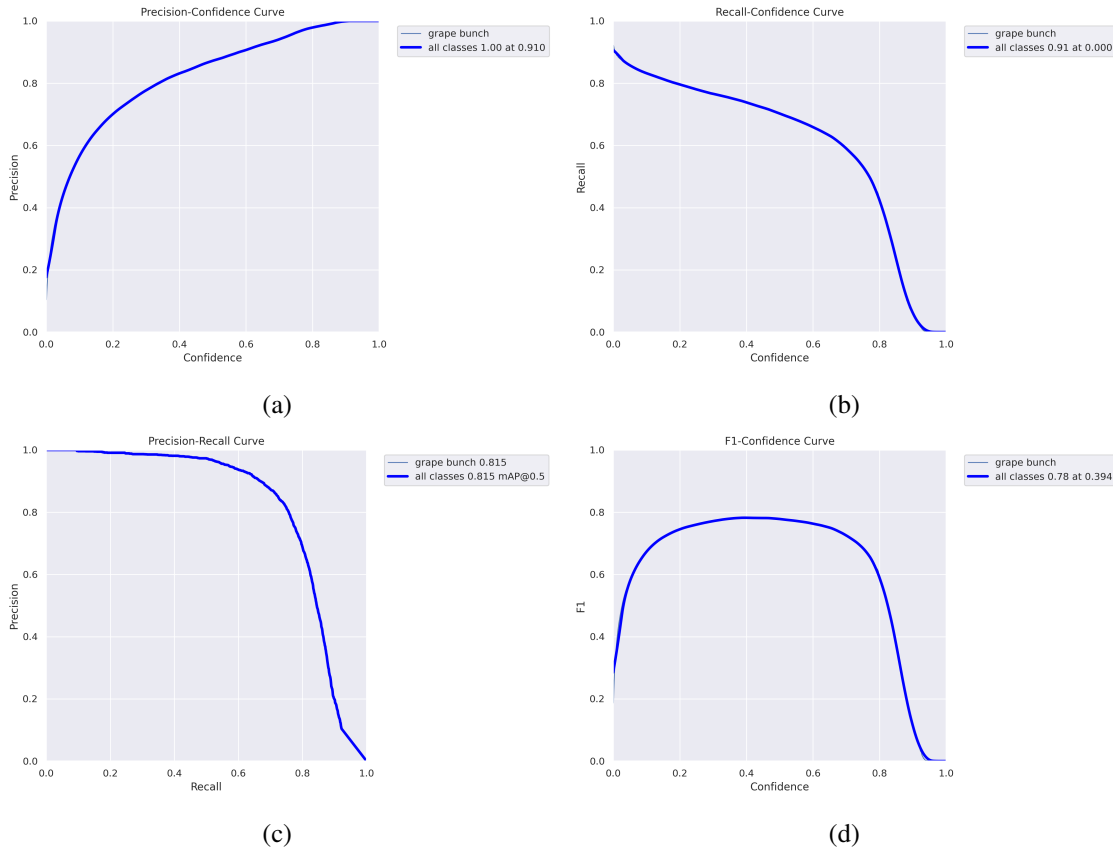


Figure 4.10: ROC curves

After testing with pictures, the model testing was performed with video capture using the final model. Using the algorithm 4.2, it was possible to connect the OAK-1 camera and use the ".blob" model retrieved from the trained model to identify the features. Fig. 4.11 shows the results with a green grape bunch. As can be seen, the results are promising and successfully detects the whole grape bunch and partially covered grape bunch.

These results were obtained in a controlled test environment, using loose grape bunches to test the inference process in the camera device. Later, in chapter 6, the integration between the hardware and software parts will be tested to validate the work in a real environment.

	mAP	Recall(R)	Precision(P)	F1_Score	PR mAP@50
yolov8n.pt	0.80	0.91 (at 0)	1 (at 0.910)	0.78 (at 0.394)	0.815

Table 4.6: Model final result after 300 epochs

Finally, after obtaining good results with YOLOv8, an SSD mobilenetV2 model was also trained. The train was performed using the same dataset. Table 4.7 shows the training parameters and results. As shown, the results are much worst than the YOLOv8 model result. Therefore, the YOLO family model was maintained.

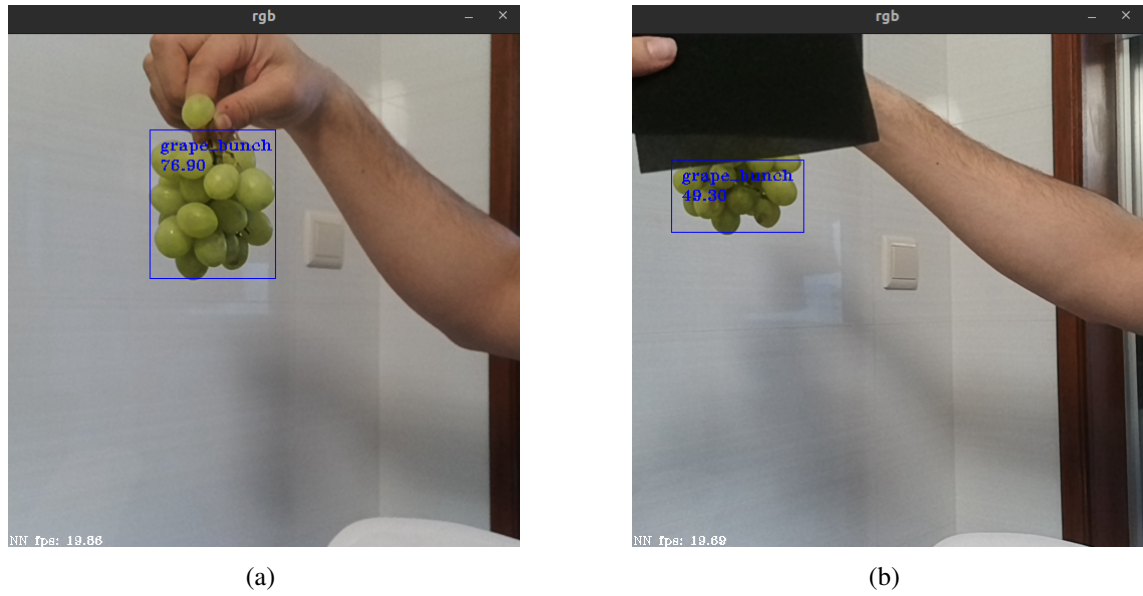


Figure 4.11: Results of grape bunch recognition with video capture

	Batch Size	Steps	Model Size	Learning Rate	PR mAP@50
SSD mobilenetV2	8	200000	12.1Mb	0.08	26.78

Table 4.7: Model SSD mobilenetV2 parameters and results

4.4 Conclusion

A comparison of popular models used in various approaches today revealed that the YOLOv8 model slightly outperformed others for this work application. This advantage stems from its ability to deliver similar or even better accuracy results without compromising the frames per second (FPS) performance. YOLOv8's status as the most recent iteration in the YOLO model series suggests a longer project lifespan. The decision to choose YOLO family models over other alternatives, such as DETR and SSD mobilenetv2, was based on their consistently superior performance.

Furthermore, a thorough examination of different model sizes concluded that the 'n' models are better suited for this work approach. Running a real-time deep learning process without a dedicated GPU can be challenging, necessitating the selection of a model size that balances performance and resource constraints on mobile devices. The 'n' models were found to be nearly four times smaller than the 's' models without compromising detection performance significantly.

In addition to model selection, the availability of a comprehensive dataset with diverse images is crucial for achieving optimal model performance. After raising the dataset size with additional images, the mean average precision (mAP) results improved by 7% when comparing the 'v8s' model before and after dataset expansion.

In summary, the results were promising in a controlled test environment, with the successful detection of grape bunches in both images and video captures. The primary challenge lies in

striking the right balance between FPS and detection performance while considering the cost-benefit trade-offs.

Chapter 5

Precision Sprayer Hardware Development

This chapter presents the objectives, tools, methodologies, and resources utilized in the hardware development process for conceiving the precision sprayer. Initially, a summary of the main goals of the hardware work is provided. The subsequent section delves into a detailed explanation of the methodology and a brief justification for the selection of tools and materials, highlighting the advantages of the chosen ones.

Furthermore, the subsequent sections present the tests, algorithms, proposed solutions, and results developed during this work. The primary aim of this segment was to design the precision sprayer hardware with two degrees of freedom (DOF). The sprayer should be able to cover a larger area of vegetation with a single spraying nozzle. The idea was to adapt a pan and tilt mechanism, as depicted in Figure 5.1, to achieve the spraying objective.

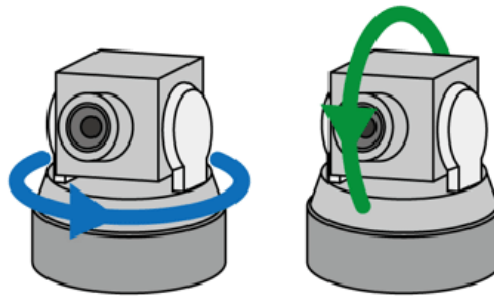


Figure 5.1: Pan and tilt mechanism [63]

5.1 Tools and Methodology

The hardware model described in Section 5.6 was developed using the Fusion360 tool. Additionally, the circuit schematics presented in Subsection 5.3.1 were designed using the Fritzing and KiCad tools. Finally, the control code was developed using the Arduino IDE.

The process began with the study of electromechanical concepts and the selection of the most suitable materials. A homing algorithm was developed to establish the desired initial positions to address the issue of the two motors' starting positions. This algorithm required the use of a hall sensor and a neodymium magnet. Subsection 2.4.5 explains how the hall sensor utilizes the electromagnetic field to trigger its status. In the GraDes solution, the hall sensors are fixed in position while the magnet rotates along the gears. The proposed model is presented in Section 5.6.

The different approaches were tested individually to validate the hypothesis of the solution. The results in Section 5.8 confirm the model's functionality. Once the electrical component was working correctly, the model was assembled with the 3D-printed parts.

After assembling the model, controlling the pan and tilt movements was necessary. Section 5.7 describes the process, from defining the angle amplitude to the execution of the spraying action.

5.2 Camera Device

The camera device used in this project is provided by Luxonis and is called the OAK-1. The Movidius Myriad X VPU powers the Luxonis OAK-1 AI Edge vision system. The system is powered through a USB Type-C port. The OAK-1 incorporates a single built-in RGB vision camera directly connected to the DepthAI Myriad X VPU for depth and AI processing. The data is transmitted to the host computer via USB 3.1 Gen 1. When the OAK-1 is used with a host, the megaAI functionality offloads all computer vision and AI tasks from the host, allowing the host to focus on its business logic. The device can perform deep learning inference processes internally within its hardware. The compact size of the device facilitates easy integration into products. For reference, Figure 5.2 visually compares the camera's size with a penny. Table 5.1 presents the camera parameters, while Figure 5.3 illustrates the camera configuration.

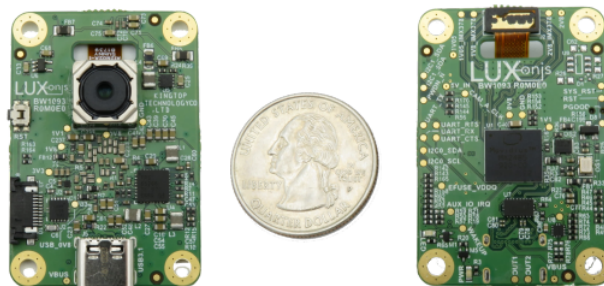


Figure 5.2: OAK-1 size comparison with a quarter coin[64]

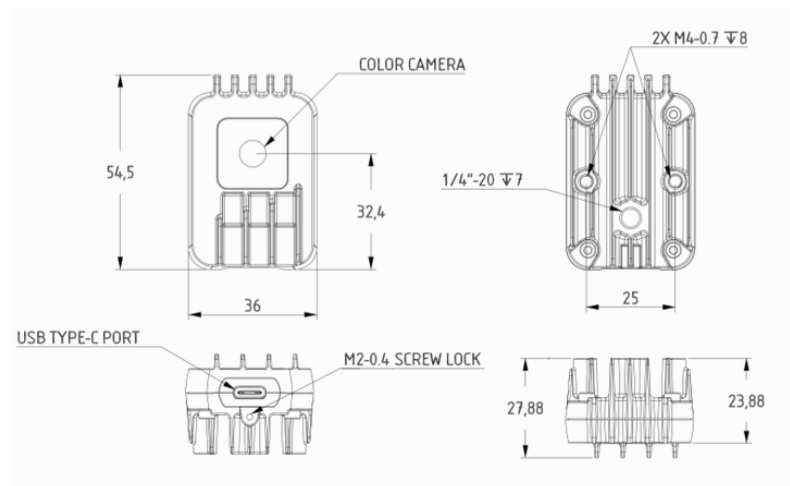


Figure 5.3: OAK-1 mechanical information [64]

Image sensor	Sony IMX378
Active pixels	4056x3030
Output video format	RAW 12/10/8
Focus type	Auto Focus 8cm - Fixed Focus 50cm
FOV	DFOV: 81° / HFOV: 69° / VFOV: 55°
IR sensitive	No
Effective Focal Length	4.81mm
Processing Power	4 TOPS of processing power (1.4 TOPS for AI)

Table 5.1: OAK-1 sensor characteristics [64]

5.3 Material Test

Several materials were used to develop the spray. The following list contains the materials used in the test and assembled model processes.

- Stepper motor 42STH48-2504AC;
- Hall effect sensor SS495;
- 3D printed model;
- Driver DRV8825 carrier board;
- PixelDriver v3 INESC TEC;
- IR Laser;
- Arduino UNO.

5.3.1 Spray Hardware Preliminary Test

Three tests were conducted to validate the different hardware components: one to validate the stepper motor control, another to validate the hall effect sensor, and a final test to integrate both modules and test the homing process of the stepper motors.

Stepper motor

The following components were utilized to test the stepper motor: the 42STH48-2504AC NEMA motor, the DRV8825 driver carrier board, an Arduino UNO, a decoupling capacitor of $240\mu F$, and a 12-volt DC power source. Figure 5.4 presents the pinout diagram of the DRV8825 stepper motor driver [65].

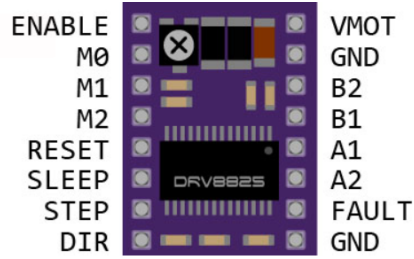


Figure 5.4: Stepper motor driver DVR8825 pin out [65]

Table 5.2 displays the electric parameters of the stepper motor, while Table 5.3 provides the driver's parameters. The DRV8825 carrier board incorporates low-equivalent-series-resistance (ESR) ceramic capacitors, which makes it susceptible to LC voltage spikes that may occur, especially when the power leads are longer than a few inches. It is necessary to connect an electrolytic capacitor between VMOT and GND to protect the driver.

A current limit had to be set to safeguard the driver and the stepper motor. Failure to specify an appropriate current limit can result in the motor drawing more current than it or the driver can handle, potentially causing damage. The reference voltage needs to be measured to set the current limit, which can be done by using a voltmeter with one probe on the potentiometer and the other on the ground (GND). The reference voltage can be adjusted by manipulating the onboard potentiometer. The following formula provides the current limit based on the measured value:

$$I_{Lim} = V_{ref} \cdot 2 \quad (5.1)$$

In this case, the potentiometer was adjusted through observation until there was enough current to move the motor. When the motor began to move, the measured voltage value was $V_{ref} = 0.52V$, resulting in a current limit of $I_{Lim} = 1.04A$.

Parameter	Value
Step angle	1.8 deg/step
Voltage	2.5 V
Current	2.5 A
Resistance	1 Ω
Inductance	1.5mH

Table 5.2: Stepper motor 42STH48-2504AC electric specifications

Drive Parameter	Value
Minimum operating voltage	8.2V
Maximum operating voltage	45 V
Continuous current per phase	1.5 A
Maximum current per phase	2.2 A
Minimum logic voltage	2.5 V
Maximum logic voltage	5.25 V
Microstep resolution	Full, 1/2, 1/4, 1/8, 1/16 and 1/32
Reverse voltage protection?	No

Table 5.3: Driver DRV8825 carrier board specifications

To complete one entire rotation cycle, 200 steps are required. A predetermined number of pulses was assigned to the motor for this purpose. In the testing configuration, the number of pulses, denoted as N_{pulses} , is four times the number of steps. The testing code, outlined in Algorithm 5.5, describes the testing procedure. Initially, it is necessary to configure the Arduino in the output mode. Subsequently, within the control loop, the motor initiates rotation in one direction using the first "for" command, followed by rotation in the opposite direction. The "DelayVel" states control the rotation speed of the motor, while the time interval for changing the operating direction is handled by "DelayDir." The micro-step capacity was not utilized during this preliminary test, as the motor operated at its maximum capacity. The following algorithm was programmed in the Arduino IDE using C++.

With the code and required components, a circuit schematic was developed for the motor testing. Figure 5.5 illustrates the circuit diagram, while Figure 5.6 depicts the experimental setup in the laboratory. The motor connections were made using a complete coil configuration to optimize torque. Conversely, to achieve higher speeds, stepper terminals should be connected in a half-coil configuration.

Algorithm 5.5 Motor testing

```

Step ← OUTPUT
Dir ← OUTPUT
while true do
  Dir ← HIGH
  for  $x \leftarrow 0$  to  $N_{pulses} - 1$  do
    Step ← HIGH
    DelayVel
    Step ← LOW
    DelayVel
  end for
  DelayDir
  Dir ← LOW
  for  $x \leftarrow 0$  to  $N_{pulses} - 1$  do
    Step ← HIGH
    DelayVel
    Step ← LOW
    DelayVel
  end for
  DelayDir
end while

```

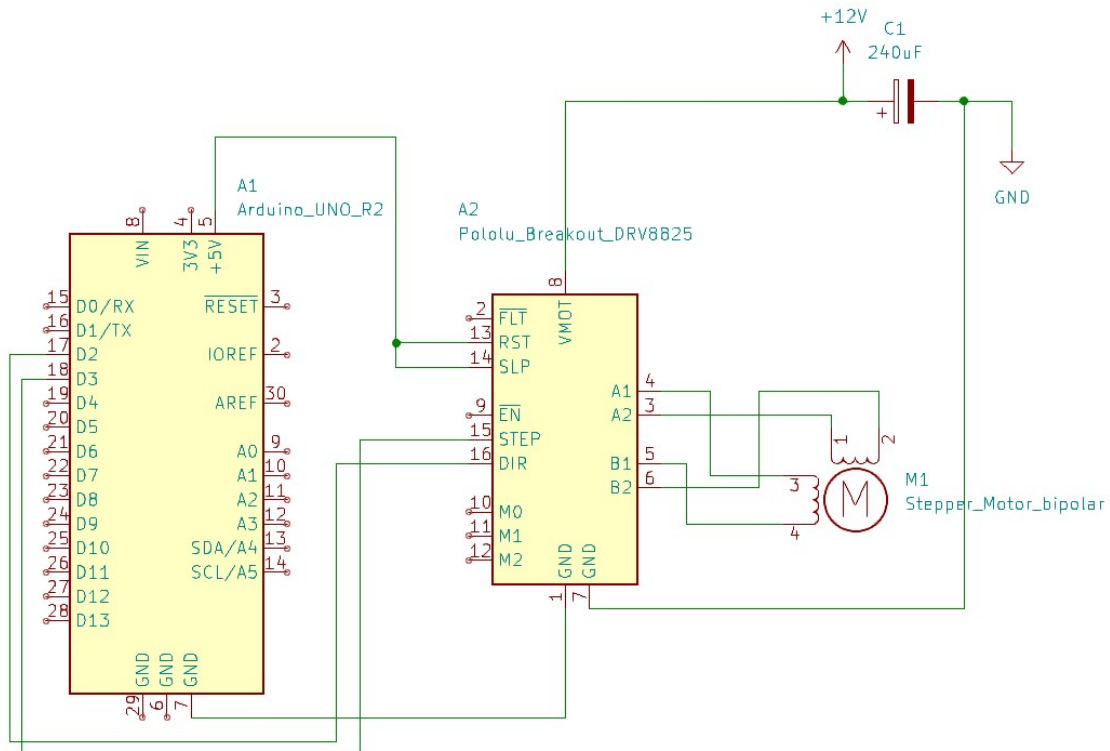


Figure 5.5: Motor experimental setup using an Arduino UNO

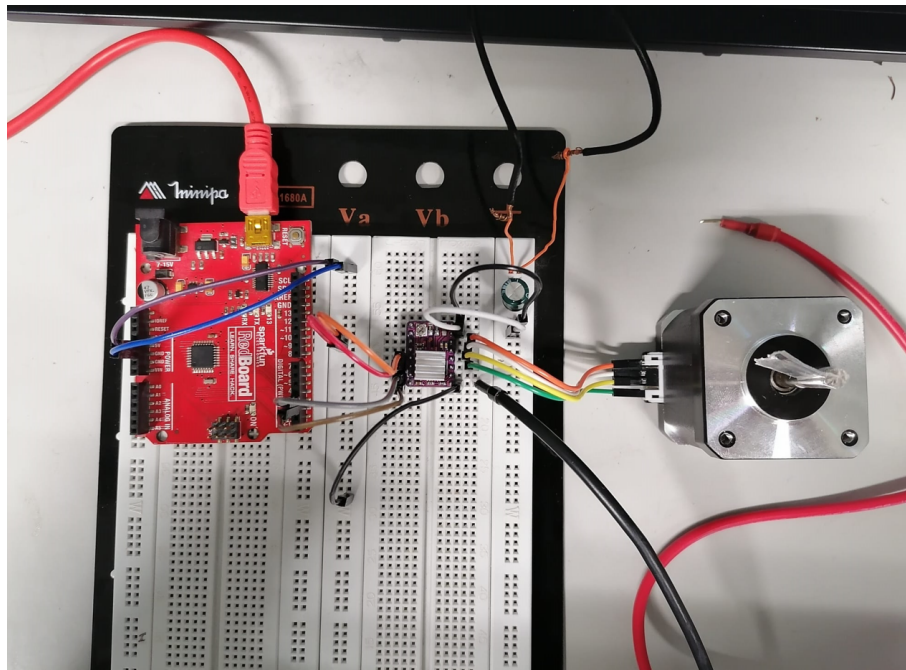


Figure 5.6: Physical motor test setup

Hall effect sensor

In order to test the hall effect sensor, an Arduino UNO was utilized to execute the code and provide a 5-volt supply to the sensor. The specific hall sensor employed for the test was the SS495 model. Upon activation, the sensor's output was set to a HIGH state when a magnet was brought into proximity. Conversely, when the magnet was removed, the output transitioned to a LOW state. Notably, reversing the magnet's polarity did not trigger a HIGH output. The results were examined using the Arduino's built-in LED by employing the algorithm outlined in Algorithm 5.6.

Algorithm 5.6 Hall sensor testing

```

LED ← OUTPUT
HallSensor ← INPUT
while true do
    if HallSensor ← HIGH then
        LED ← HIGH
    end if
end while

```

With the code and the necessary materials, the circuit scheme was developed. Fig. 5.7 shows the circuit schematic, and Fig. 5.8 the experimental setup.

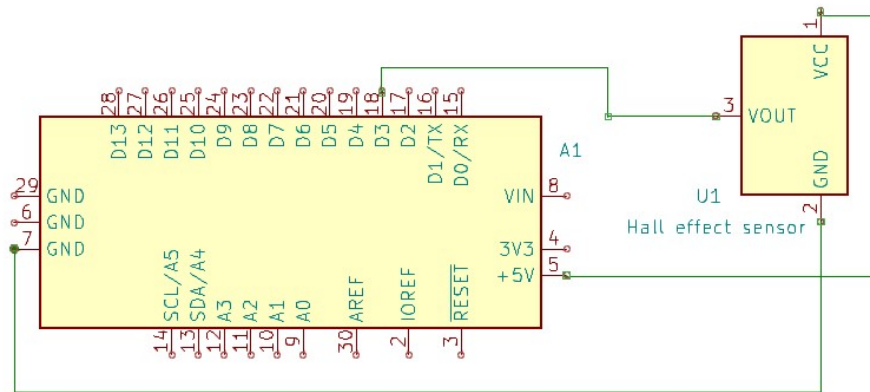


Figure 5.7: Hall sensor experimental setup

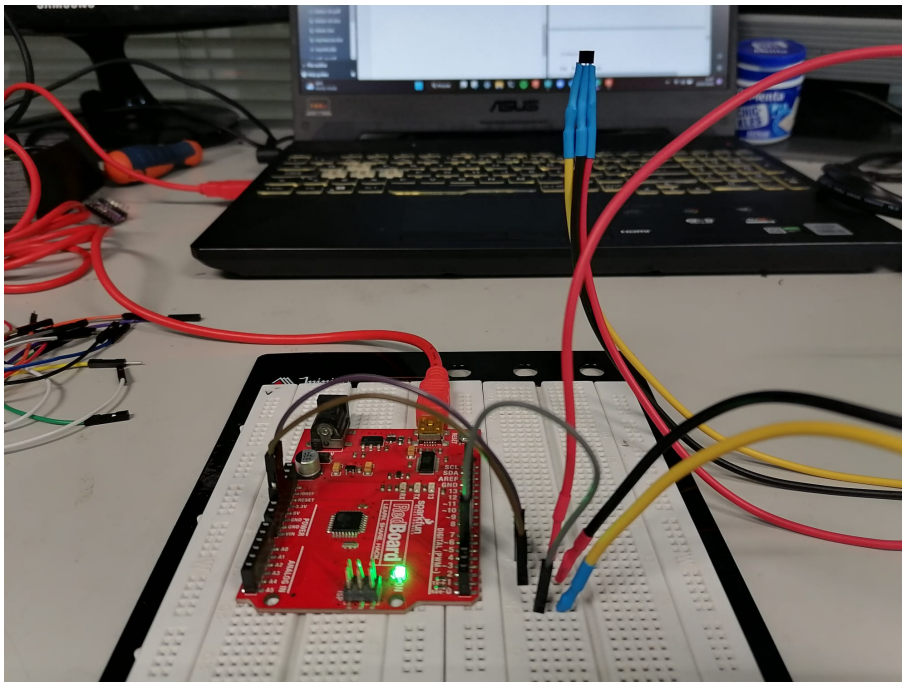


Figure 5.8: Physical hall sensor test setup

5.3.2 Homing process

The homing testing process involved combining the two previous tests. For this test, a magnet was attached to a piece of plastic and assembled onto the motor shaft. All other test conditions, such as current limits and materials, remained the same as in the previous tests. The objective of this test was to move the motor until it reached the home position, which corresponded to the position of

the sensor. The test yielded positive results, as the motor stopped when the sensor was triggered. Upon triggering, the position variable was updated to 0, assuming the current motor position was the initial position.

Algorithm 5.7 presents the pseudo-code that was followed to control the stepper motor.

Algorithm 5.7 Home position testing

```

Step ← OUTPUT
Dir ← OUTPUT
HallSensor ← INPUT
HomePos ← CurrentPos
while true do
    while HallSensor ← LOW do
        Dir ← HIGH
        Step ← HIGH
        DelayVel
        Step ← LOW
        DelayVel
    end while
    HomePos ← 0
end while
  
```

Also, fig. 5.10 and 5.9 shows the circuit schematic and fig. 5.11 the experimental setup.

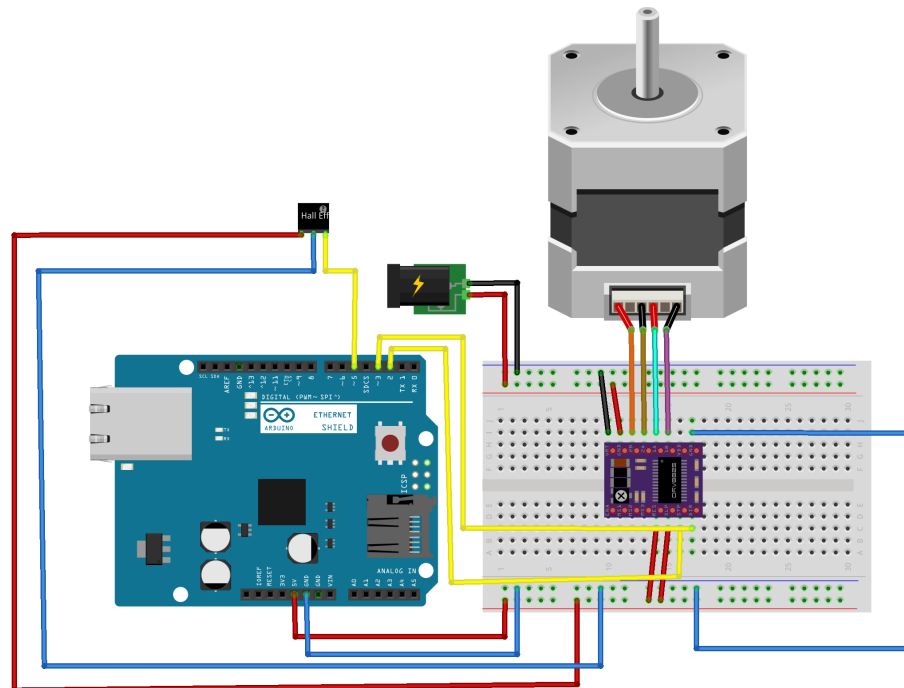


Figure 5.9: Homing mechanism experimental setup

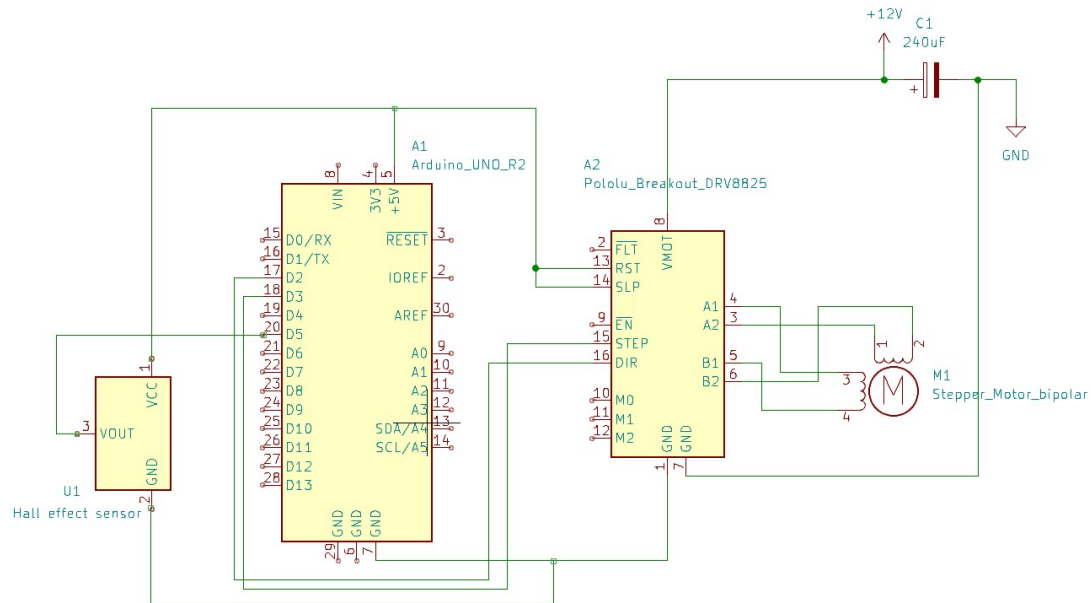


Figure 5.10: Homing mechanism experimental setup

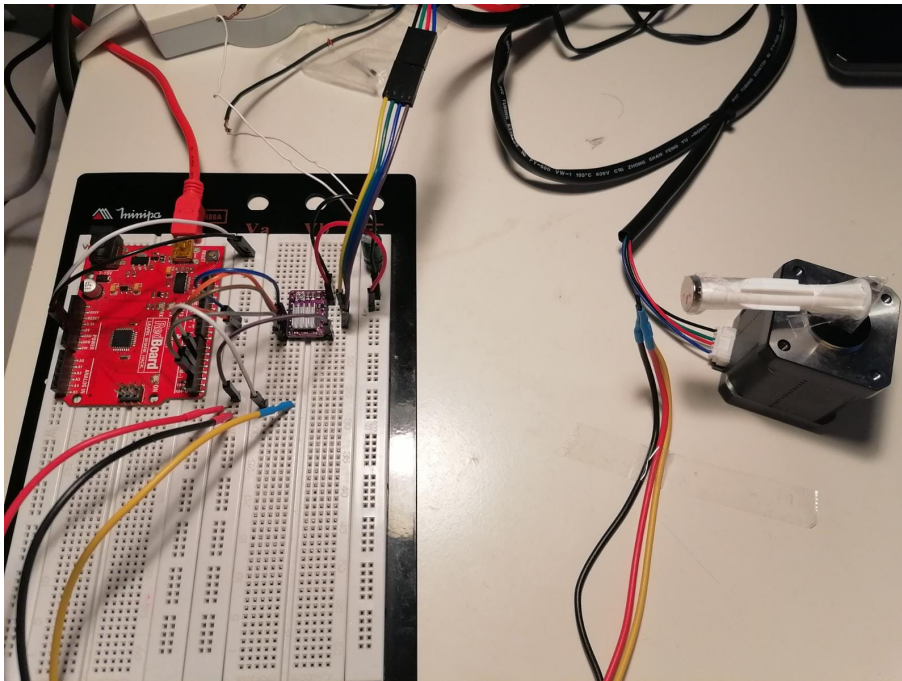


Figure 5.11: Physical homing mechanism experimental setup

5.4 PixelDriver v3

Two stepper motors are required to control the pan and tilt movement. Hence, two DRV8825 driver carrier boards were employed to control these motors. Figure 5.12 displays a PCB project

developed by Francisco Terra at INESC TEC [66]. The schematic of the driver can be found in Appendix B.

The driver incorporates two DRV8825 carrier boards to control the motors, along with an ATmega32U4 microcontroller. The PCB operates on a 12-volt power supply. This well-designed driver simplifies electrical connections by directly integrating resistors, capacitors, and logic power (the 5-volt input from the USB connection) on the PCB. Additionally, the driver features a built-in USB-A port, enabling direct code uploading to the microprocessor using the Arduino IDE.

However, the sensor must operate using analog outputs to achieve more precise results. With the digital configuration, it is impossible to define an arbitrary threshold. The threshold can be adjusted to trigger the sensor at the desired point by changing it to an analog configuration.

A minor modification was made to the board to accommodate the analog outputs of the sensor. The necessary changes were implemented accurately by analyzing the schematic presented in Appendix B. Since the micro-step resolution was not required, the connections between the MS1, MS2, and MS3 pins of the DRV8825 carrier board U1 and the microcontroller were removed. Instead, the associated board pins were directly welded to the H2 connector's trail pins. Consequently, the analog outputs of the microprocessor, PF0, and PF1, were connected to H2 trail pins 55 and 56, respectively. The MS3 pin was connected to H2 trail pin number 14 to send a signal to the DC motor fan in the future. Finally, pin 16 was soldered to GND.

The pin-out mapping used in the control code is specified in the Arduino Leonardo documentation.

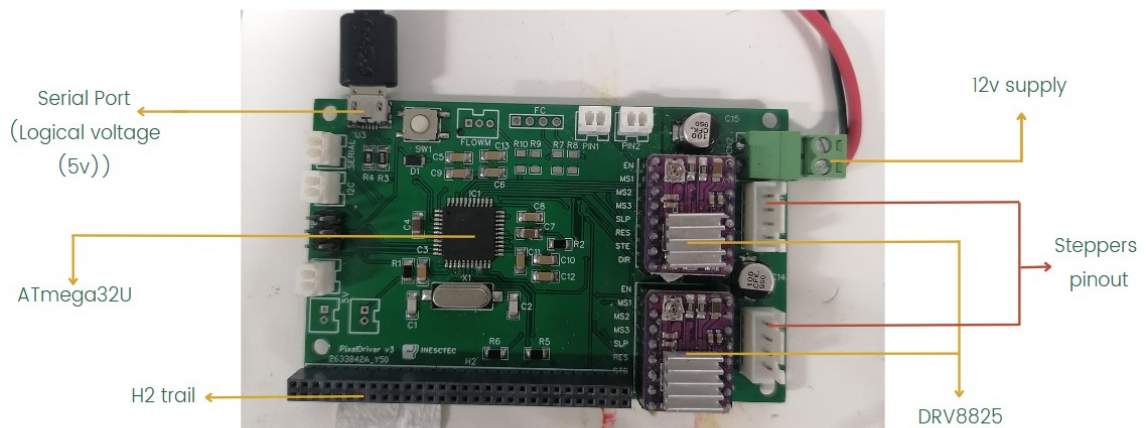


Figure 5.12: PixelDriver v3 INESC-TEC

5.5 Spraying Nozzle and Fan

The spraying module is securely attached to the tube structure of the mechanism. This module comprises a fan, DC motor, and spraying nozzle. The control of the spraying module is straight-

forward, with two states: on and off. When the sprayer is directed toward the grape bunches, the control code will signal the fan to turn on or off. The duration of the fan action has been programmed to be 2.5 seconds. The fan control code has been implemented as a function within the mechanism's control algorithm. Furthermore, a YEP 80A SBEC Brushless Speed Controller driver has been connected to the fan component to ensure proper DC motor control. Figure 5.13 showcases the spraying module and its integration within the mechanism.

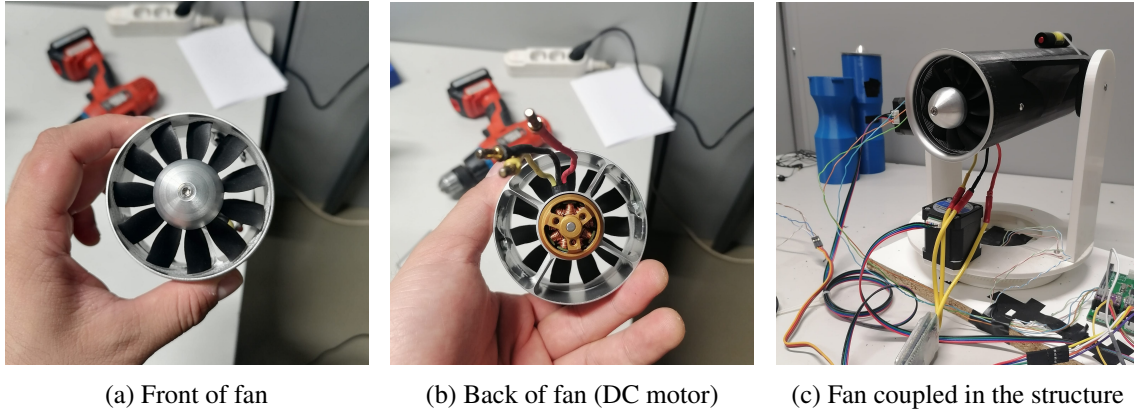


Figure 5.13: Sprayer fan mechanism

5.6 3D Model

The pan and tilt camera device's design inspired the sprayer model [67]. The concept was adapted to replace the camera with a sprayer drum, resulting in a hardware configuration that enables the proposed two degrees of freedom (DOF). The pan movement corresponds to the yaw movement, while the tilt movement is equivalent to the pitch movement, as illustrated in Figures 2.11 and 5.1.

Specific 3D-printed components feature small spaces designed to accommodate magnets and hall sensors. Hall effect sensors were employed to establish an initial position for the sprayer, allowing the mechanism to identify a "home" position. This process is commonly referred to as "homing." In brief, when the sprayer begins its movement, the homing function is activated to align the hardware to its initial position. Figure 5.14a illustrates a designated location for placing the magnet in the tilt movement, along with four screws. Figure 5.14b displays a central position for mounting the hall sensor in the pan movement. Both the pan and tilt structures incorporate a magnet and a hall sensor. Moreover, the model can also be seen in Figure 2.10.

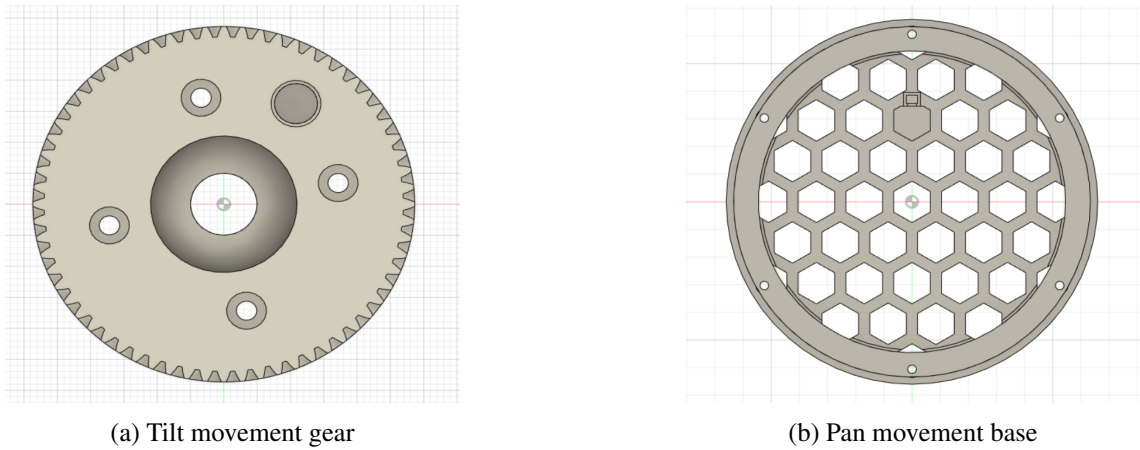


Figure 5.14: Hall sensor and magnet structure examples

As the GraDeS solution builds upon previous works, it was necessary to adapt the model to incorporate the validation IR laser on top of the spray drum. Within the cylinder, the spraying system will be implemented, following a similar configuration to the model presented in Figure 2.2. Lastly, Figure 5.15 showcases the finalized 3D model, which has been developed and is ready for printing.

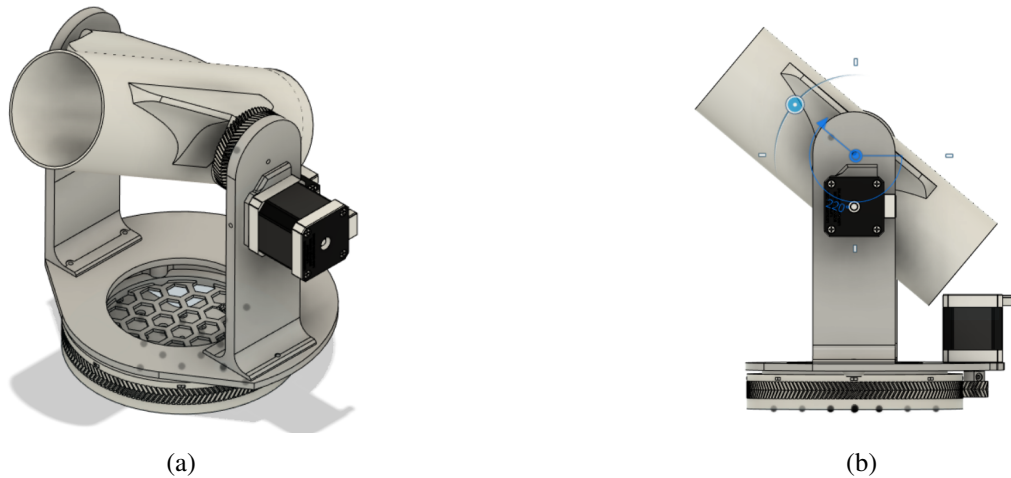


Figure 5.15: Sprayer 3d model

5.7 Sprayer Operational Angle

First and foremost, it is crucial to determine the maximum angle at which the device will operate. Figure 5.16 illustrates a sample problem diagram. The maximum operational angle should be set to capture the highest point within the camera's field of view.

The following assumptions are made: the robot car/spray will be positioned 1 meter from the grapevine, with a height of 0.6 meters. The distance between the camera and the sprayer laser is 0.2 meters. The field of view (FOV) value and the distance between the camera and the spraying

point are known. By employing basic trigonometry and these known values, it is possible to calculate the angle. The first step is to determine the value of $h1$.

$$h1 = \tan\left(\frac{FOV_{angle}}{2}\right)$$

Then, use the $h1$ value to find the maximum operational angle

$$h2 = h1 - 0.20$$

$$angle = \arctan\left(\frac{h2}{1}\right)$$

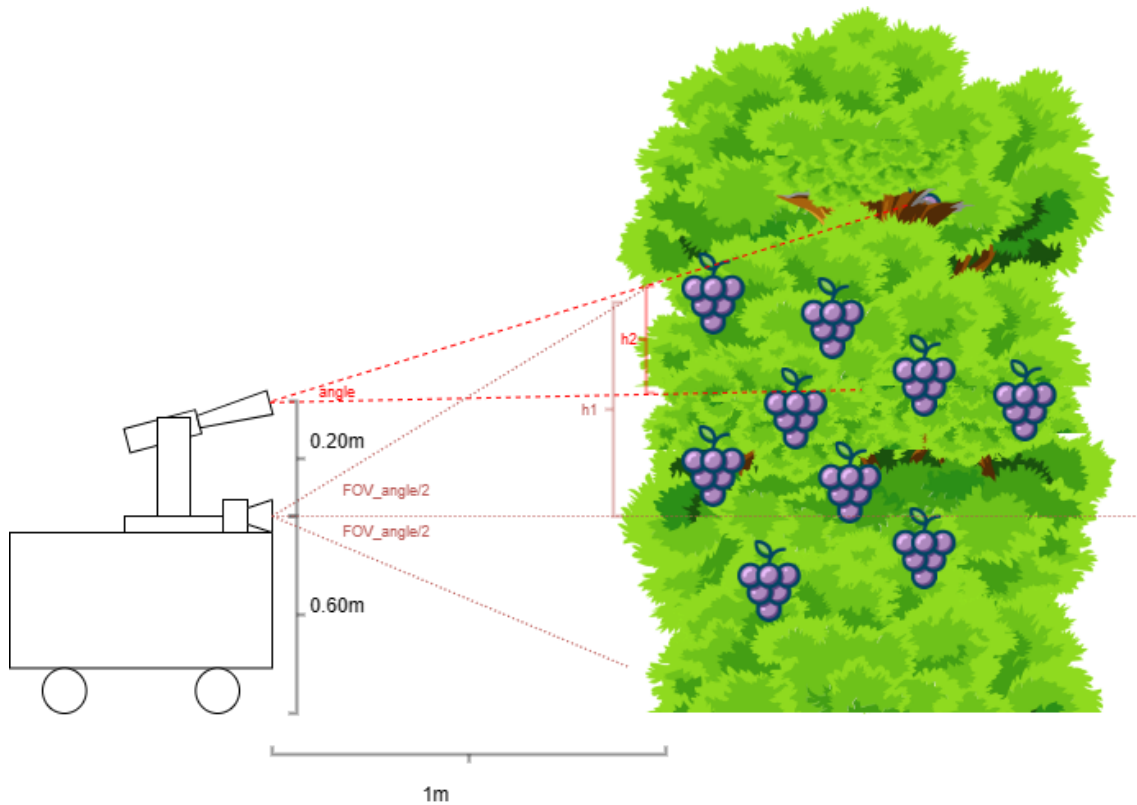


Figure 5.16: Sprayer angle model representation

The fully assembled hardware was subjected to testing by connecting the stepper motors to the structure. The hall sensors were also integrated into the sprayer to facilitate the homing process. One sensor and one magnet were linked to the left gear responsible for the tilt movement, while another sensor and magnet were connected to the gear responsible for the pan movement. In contrast to previous tests, the sensors were now connected to the analog output with a threshold value 465. This adjustment was necessitated by the presence of the sprayer's lateral supports,

which caused interference in the measurements. Switching to analog output improved control precision.

Regarding positioning, the zero position for the tilt movement was defined when the cylinder assumed a parallel position with the base, with the laser pointing forward. Similarly, the zero position was determined for the pan movement when the hall sensor and the pan stepper motor were aligned. The tilt movement from these respective home positions allowed for a total aperture dependent on the camera's field of view. As for the pan movement, a working angle of ± 50 degrees was considered from the home position.

5.8 Results

The mechanism control has been successfully implemented using a combination of algorithms from previous tests in this chapter. The key difference now is the utilization of the "AccelStepper.h" library to control both motors, along with adding an angle value input request on the serial monitor to enable the movement of the machine to the desired positions.

The control system was validated by providing angle inputs to the machine through the serial monitor. The first validation focused on the homing process, whereby upon turning on the device, it was expected to return to the home position. As depicted in Figure 5.17, the cylinder reached the desired position after being powered on.

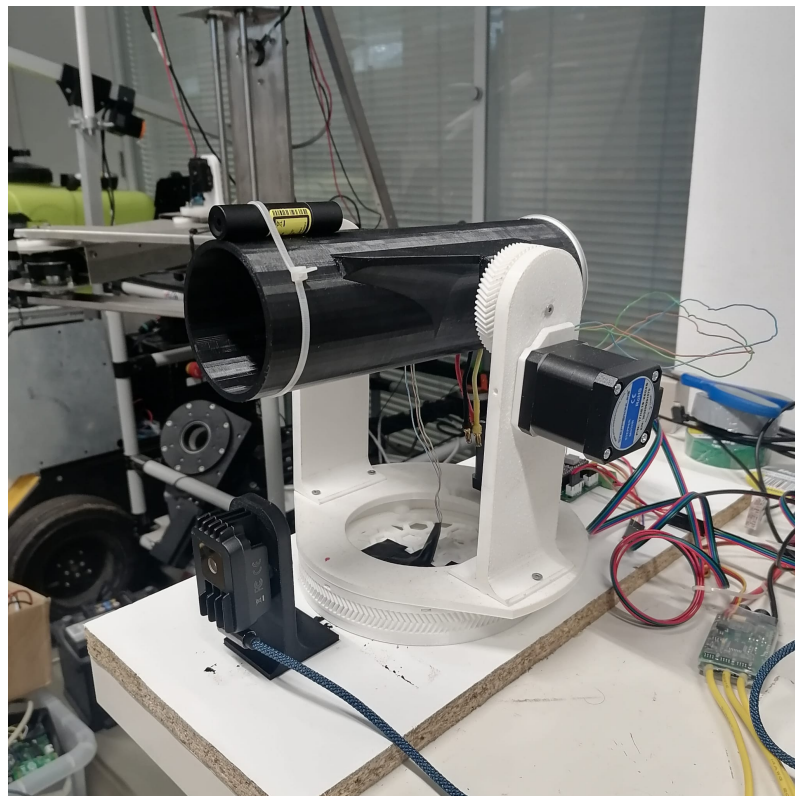


Figure 5.17: Mechanism after homing process

In the second validation, the sprayer tilt mechanism was adjusted to point downward, perpendicular to the base. Subsequently, three consecutive arbitrary inputs were given to test the tilt movement. The first input was set to 30° , followed by 60° , and finally, 51° , resulting in a total of 141° . While there is no hardware limitation for the pan basis, it was assumed that the pan movement would be restricted within $\pm 50^\circ$, as previously mentioned, to accommodate the wire connections. Figure 5.18 depicts the corresponding movement for each input.

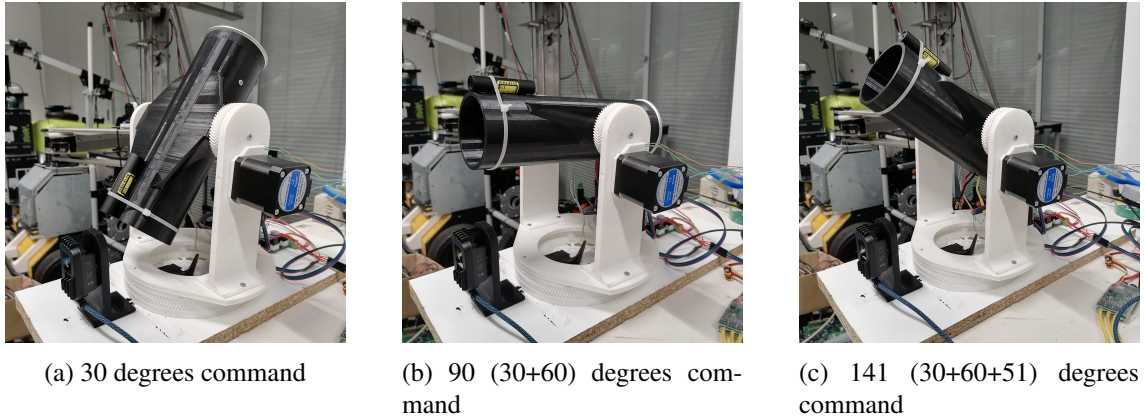


Figure 5.18: Sprayer positions validation

5.9 Conclusion

The utilization of hall sensors proved to be an excellent and cost-effective solution for the homing process. This method offers precision and reliability, requiring only two sensors and two magnets.

Additionally, the OAK-1 camera has proven to be a suitable choice. This compact camera possesses ample processing power, enabling it to perform neural network inference processes without needing a GPU or a separate mini-computer, such as a Raspberry Pi.

Furthermore, the integration of the PixelDriver v3 simplified the implementation process. Thanks to the pre-designed and soldered connections of the DRV8825 carrier board on the PCB, the hardware connections only required the remaining power and stepper motor connections.

In summary, the hardware control has been successful, allowing the sprayer to point accurately in the desired angular direction based on manual input. The following chapter will combine the software and hardware components to automatically receive coordinate points and utilize them to direct the sprayer accordingly.

Chapter 6

Hardware and Software Integration

6.1 Introduction

This chapter presents the objectives, tools, methodologies, and resources for integrating hardware and software components. Firstly, a summary of the primary objectives of the integration work is provided. Subsequent sections elaborate on the methodology, providing a detailed explanation and a brief justification for the chosen tools and materials, emphasizing their advantages. Furthermore, this chapter presents the tests, algorithms, proposed solutions, and results developed during this work.

The primary objective of this phase was to integrate the sprayer hardware, as discussed in Chapter 5, with the detection software based on deep learning techniques from Chapter 4. The sprayer must be able to receive an (x, y) coordinate and align itself in the corresponding direction. It is important to note that the camera and the sprayer will be in different coordinate frames. Therefore, robotic kinematics will convert the (x, y) coordinate from the camera to the sprayer frame. Additionally, the (x, y) coordinates must be converted into angle values to input the pan and tilt mechanism. Serial communication will transmit information between the processing units, enabling communication between the two modules.

6.2 Tools and Methodology

This chapter will use the results and conclusions from the previous chapters to assemble both works into one.

First, finding a way to communicate between the detection algorithm and the sprayer control codes was necessary. Serial port communication was used between the software and the hardware parts. The communication was performed with the aid of a Python script. The script executes a sub-process, that is, runs the inference code. The inference code returned an (x, y) coordinate of the bounding box center, an ID, and an (x, y) coordinate of the laser position. This coordinate was then sent through the serial port to the microprocessor of the driver. The driver then uses the value

as input to the hardware algorithm. The creation of the script simplifies the program execution, requiring only one command to execute the whole system.

Assigning an ID to each bounding box was necessary to spray multiple grape bunches without repeating themselves. These IDs were then read one by one using a buffer. Moreover, another function was developed to control the spraying action.

6.3 Sprayer Input Value

As previously mentioned, the inference software provides two-dimensional coordinates to the mechanism, necessitating the conversion of the (x, y) point into angle values. The mechanism operates with two degrees of freedom, requiring the computation of two angle values: one for pan movement and another for tilt movement.

Initially, it was crucial to determine the camera's effective field of view (FOV). While Table 5.1 provides a constant value for the camera's FOV, resizing the capture window to a resolution of 640x640 can alter the FOV angles. A measuring tape was used to determine the total size in meters from one edge to another in the captured image. By defining the distance to the capture point as one meter, the total FOV angle could be easily determined. The following equation utilizes trigonometry to compute the necessary angles.

$$VFOV/HFOV = \arctan\left(\frac{size_meas}{dist_to_capture}\right) \quad (6.1)$$

Where *size_meas* represents the measured vertical/horizontal amplitude captured by the camera, and *dist_to_capture* was defined as one meter, both angles returned a value of 33 degrees, confirming the change. Additionally, the camera lens was designated as the global static reference. The calculation was simplified since the laser is consistently fixed and translated 20 cm from the zero reference point along the x-axis. Figure 6.1 illustrates the coordinate system.

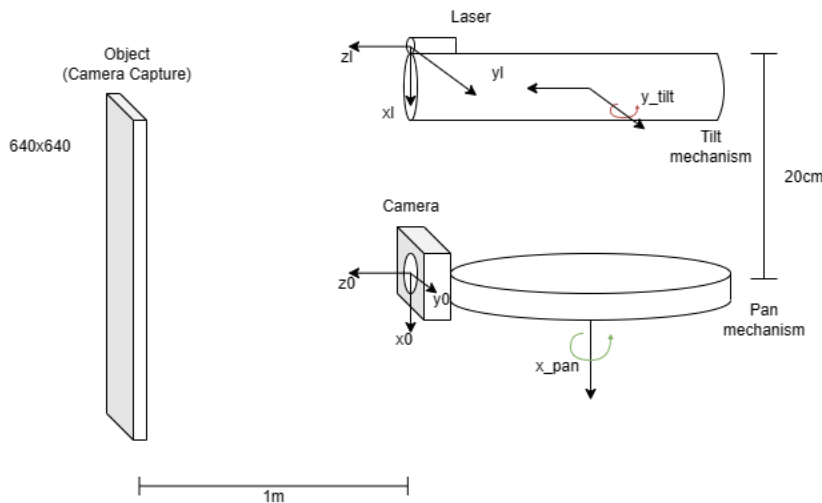


Figure 6.1: Coordinate vector referential

The translation was applied to the x-axis, and the angles were calculated with the following equations:

$$pan_angle = \arctan\left(\frac{y_c - y_m}{dist_to_capture}\right) \quad (6.2)$$

$$tilt_angle = -\arctan\left(\frac{x_c - x_m + 0.20}{dist_to_capture}\right) \quad (6.3)$$

Where x_c and y_c are the image center and x_m and y_m are the grape bunch center coordinate. The angle values are then sent to the driver through the serial port as input to the stepper motors.

6.4 Code Integration

The integration of the code was achieved by creating a Python script that allows the simultaneous execution of both modules. The script initiates by establishing a serial communication with the driver microprocessor, enabling the software to send information to the hardware. Once the communication is established, the script sends the homing command to the mechanism, positioning the motors at the starting position. Subsequently, the camera inference process begins, tracking the grape bunches in the video capture.

The primary challenge in this phase pertained to the IDs assigned to the grape bunches. Each bunch received an ID when the code was executed; however, the IDs became rearranged when a bunch went out of the capture area. For instance, if three grape bunches were in the capture, the left bunch would have ID=0, the middle bunch ID=1, and the right bunch ID=2. However, when the mechanism moved to the right, and the left bunch went out of the image, the ID of the middle bunch was rearranged to 0 and the right bunch to 1. Consequently, the mechanism could not determine which grape bunches had already been sprayed.

The solution involved modifying the software code from a detection-based approach to a tracking-based approach. In other words, the inference not only detects the objects but also tracks their movement and associates a unique ID number with each bunch in the capture.

Two tracking algorithms were tested: zero-term tracking and short-term tracking. Zero-term tracking focuses on object association rather than prediction and monitoring based on previous tracking data. Object association entails linking the detected items from an external detector to the tracked objects identified and tracked in previous frames. On the other hand, short-term tracking enables the tracking of objects between frames, reducing the need for object detection in each frame.

Initially, a short-term Kernelized Correlation Filter (KCF) tracking algorithm was implemented. KCF utilizes circulant matrix features to enhance processing speed [68]. However, it was observed that this approach significantly reduced the frame rate. Consequently, an alternative short-term, imageless tracker was utilized. This tracker enables object tracking in frames where object detection was skipped, achieved by extrapolating object trajectories from prior detections. Another

tracker tested was the zero-term color histogram tracker, which tracks an object based on its position, shape, and input image information, such as the RGB histogram. However, this tracking mode exhibited some inconsistencies.

Moreover, two main threads were implemented to prevent unnecessary or undesirable movements. One thread executes the primary process, which involves processing the grape bunches. The other thread runs an algorithm to check if the inference code on the computer side has stopped. If the inference code has stopped, the script sends a signal to the microprocessor and halts the primary process. When the inference code restarts, the script sends an alert to the microprocessor, resuming the main task. Prior to the implementation of the threads, the inference and movement codes were not synchronized. Consequently, when the inference stopped, the movement would continue processing the bunches and moving. Running the two tasks in parallel enabled a comprehensive check on the process status, ensuring that the code inside the driver only executes its actions when the inference is running.

Finally, a function was implemented to monitor the buffer's capacity. Due to memory limitations, the buffer has a size of 600, allowing it to process 600 grape bunches. However, in large-scale vineyards, the number of bunches can exceed thousands. To address a significant amount of grape bunches, a method was implemented to check the remaining capacity of the buffer. When ten free spaces remain in the buffer, the algorithm flushes the buffer until 20 positions before reaching its total capacity. Subsequently, the data is moved to the initial positions of the buffer, and writing begins again from the next empty position. For instance, in a buffer with a size of 600 positions, when it reaches position 590, the algorithm flushes the data from position 0 to 579. The remaining data from positions 580 to 600 is then moved to positions 0 to 20. Following this process, the algorithm resumes writing from position 10. This approach ensures that the data from the bunches in the video capture is recovered.

6.5 Control Process

The GraDeS solution comprises two principal codes: the inference and control codes. The control process is implemented in the code transmitted to the microprocessor. A solution was devised to effectively handle the data transmitted by the software, including the bunch position, laser position, and bunch ID for multiple bunch detections in the video capture.

To address this challenge, a class named "Grapes" was created to receive and store these variables. When the software sends the values, the hardware code assimilates them within the "Grapes" class. Each detected grape bunch is then inserted into a vector, enabling the processing of all the detections. Additionally, a buffer was implemented to manage the grape bunches already processed by the mechanism, ensuring that each grape bunch is sprayed only once.

In summary, the serial port connection serves as the means of communication between the software and hardware components. The inference code returns the (x, y) coordinate of the bounding box center, the ID, and the (x, y) coordinate of the laser location. This coordinate is subsequently transmitted to the driver via the serial port. The driver utilizes this value as input for the hardware

algorithm. The algorithm creates instances of the "Grapes" class and inserts them into a vector. Then, a function processes each grape bunch once. The control code applies the equations derived in subsection 6.3 to manipulate the mechanism.

6.6 Results

First, the main modules of the sprayer were assembled: the OAK-1 camera, the pan and tilt mechanism, and the sprayer fan. The process starts with the execution of a central Python script that manages the inference code (on the computer side) and the sprayer movement code (on the micro-processor side).

The test was conducted inside the laboratory using a wood pallet with leaves and plastic grape bunches attached. The objective is to simulate a vineyard. So the camera was put one meter from the pallet to recognize the multiple bunches in a leaf background. Fig. 6.2 shows the test environment.



Figure 6.2: Testing leafed background

Three main tests were planned. The first test consisted of putting the grape bunches in the leafed background and observing if the machine detects the bunch and if it moves in the grape bunch direction. The second test involved reordering the grapes bunches in the leafed background and observing if the machine performed the expected movement. Finally, the following test consisted of running the mechanism and adding some environmental noise. The implemented noise was a black object passing in front of the camera and covering the grape bunch briefly. The system should have the same behavior as in the previous test, keeping track of the lost grape bunches and aiming at the detected ones.

Six grape bunches were placed randomly in the leafed background for the first test. The mechanism was set one meter away from the detection point. In fig. 6.3, it is possible to see the

screen capture of the inference code running on the computer. The "Obs" point, given by the green square in the image, represents the laser position.

The laser position was retrieved using a simple color segmentation algorithm with a red mask (laser color). The upper left image represents the first bunch processed by the code, and the bottom right image represents the last. The results were positive for this test. The sprayer successfully tracked and aimed toward the six grape bunches. The following results were obtained and presented in fig. 6.3.

For the second test, the grape bunches were rearranged in other random positions to validate the sprayer again. A higher "level" of difficulty was inserted this time. Two bunches were put side by side to confuse the inference process. The algorithm must aim the sprayer in the fruit direction independent of the bunch position in the image. Again, the results were all positive. The sprayer likewise successfully tracked and aimed toward the six grape bunches. Fig. 6.4 shows the respective results.

Finally, the previous grape bunches' positions were maintained for the last test. During the spraying process, the noise was applied in the detection inference. The objective was to guarantee that after the camera capture interfered with any noise, the inference saved a bunch of information and kept track of the fruits. Also, the noise in the image must not affect the spraying process; that is, the spraying must continue to aim at the respective positions.

The test was mostly successful. After passing an object in front of the camera and leaving briefly, the inference keeps track of the bunch's position and assigns a "LOST" status. After removing the object, the tracking and spraying process were unaffected, returning the "TRACKED" status. However, to periods of noise bigger than three/four seconds, the algorithm losses the fruit position and assign a new ID to the bunch as shown in subfigures 6.5e and 6.5f. Fig. 6.5 shows the results for the last test.

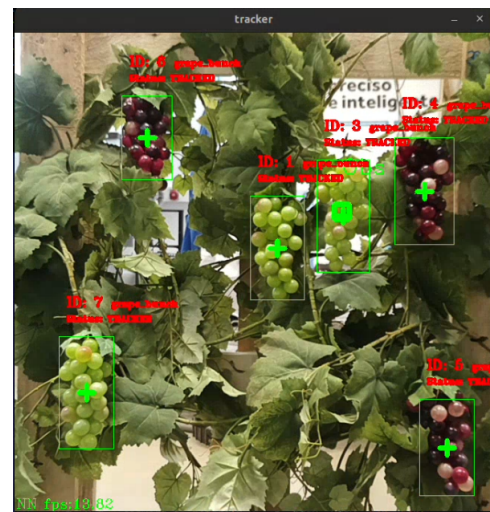
6.7 Conclusion

The structure successfully interpreted the information sent by the inference algorithm. With the detection center information, the microprocessor sent the correct angle values to the sprayer. With these values, the mechanism aimed with success in the grape bunch direction.

Moreover, implementing threads to handle the execution of the different modules of the sprayer proved effective. This functionality allows us to run and stop the inference and movement actions simultaneously. Also, it unified both processes in a single Python script. This unification permitted control of the restart process inside the script rather than manually uploading the code in the driver.



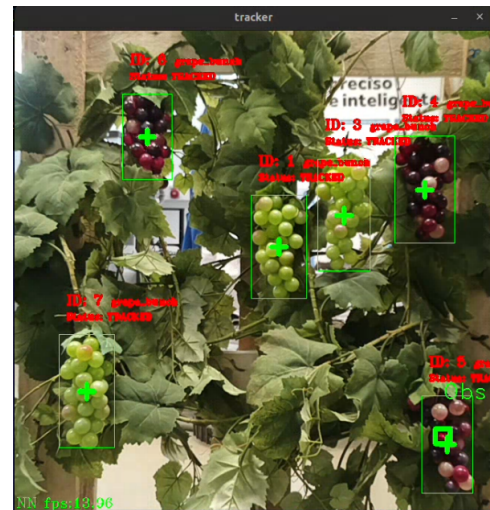
(a) First grape bunch aiming



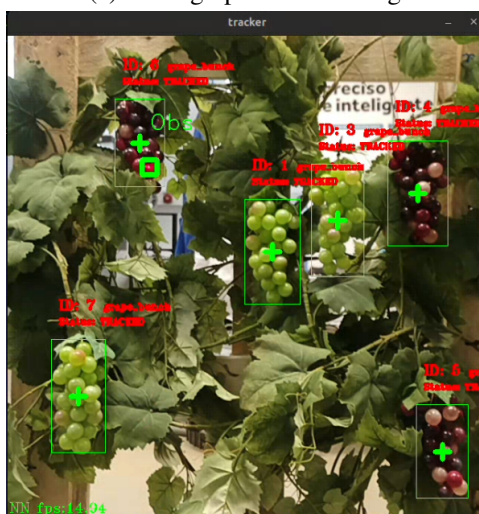
(b) Second grape bunch aiming



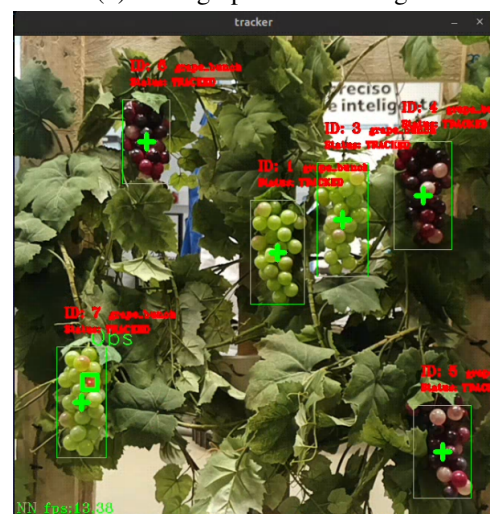
(c) Third grape bunch aiming



(d) Forth grape bunch aiming



(e) Fifth grape bunch aiming

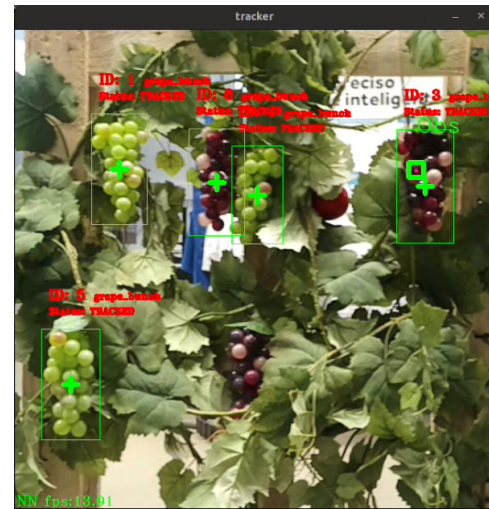


(f) Sixth grape bunch aiming

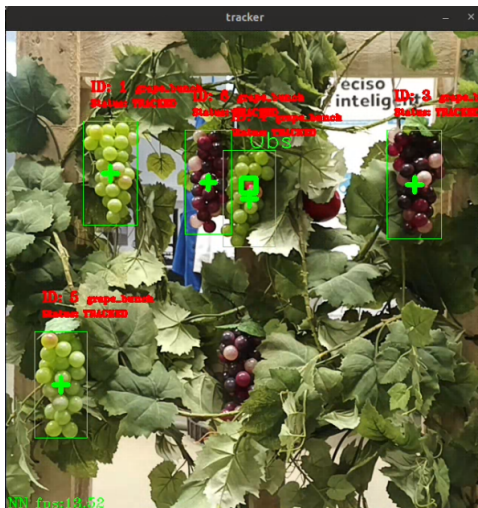
Figure 6.3: First test results



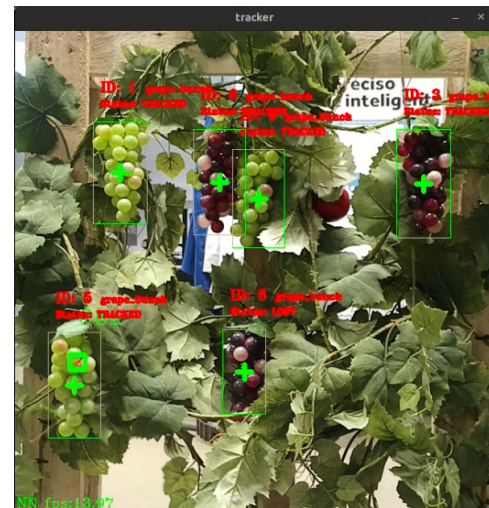
(a) First grape bunch aiming



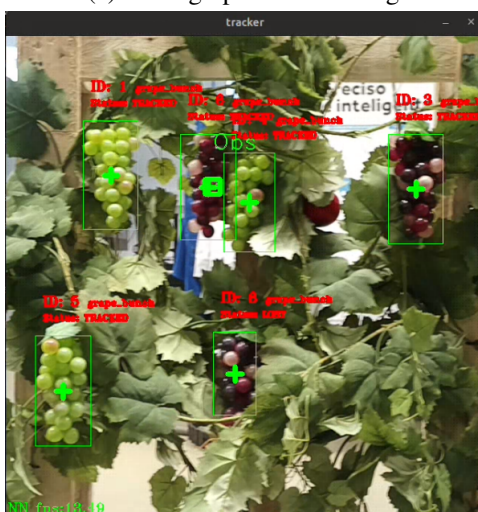
(b) Second grape bunch aiming



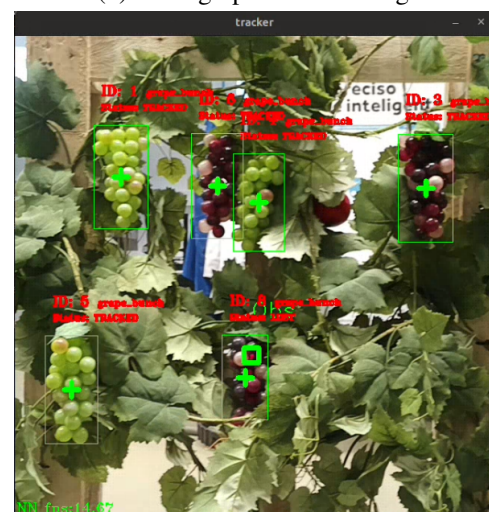
(c) Third grape bunch aiming



(d) Forth grape bunch aiming

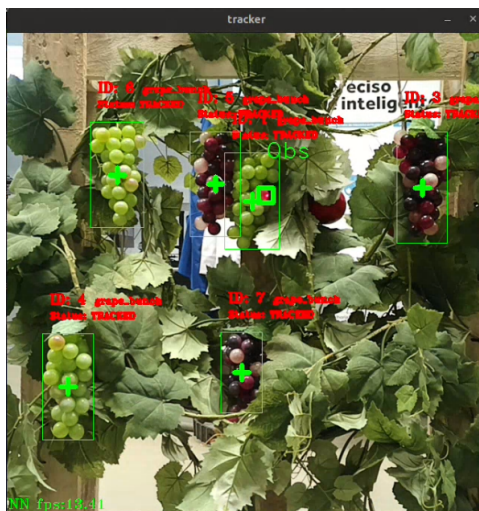


(e) Fifth grape bunch aiming



(f) Sixth grape bunch aiming

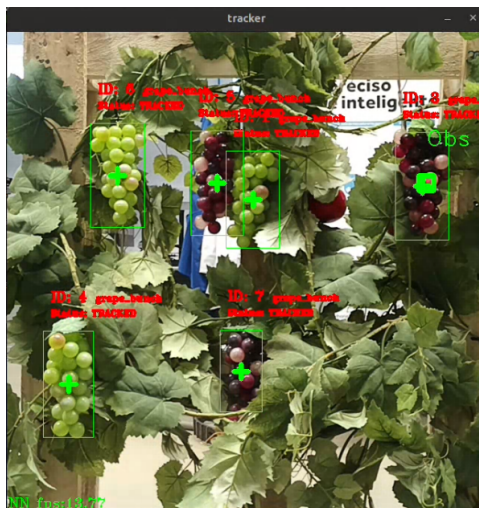
Figure 6.4: Second test results



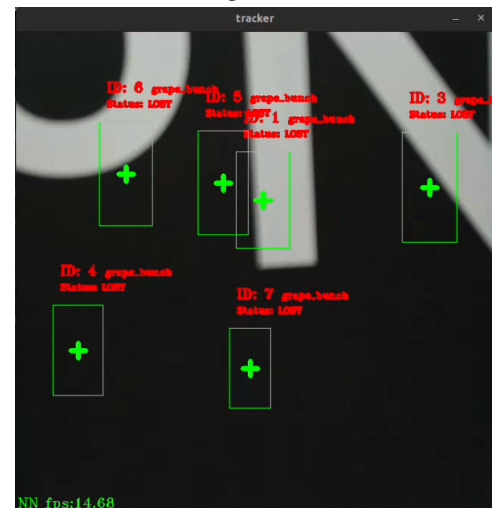
(a) Tracked bunches



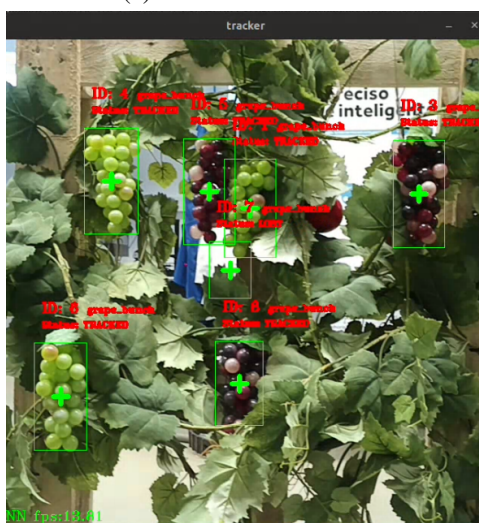
(b) Image noise one



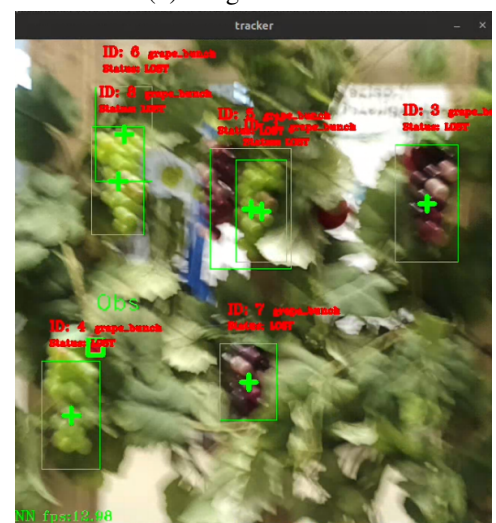
(c) Re-tracked bunches



(d) Image noise two



(e) Bottom grape bunch lost (ID:7 and ID:8)



(f) Upper grape bunch lost (ID:6 and ID:8)

Figure 6.5: Third test results

Chapter 7

Conclusions and Future Work

A vision-based intelligent sprayer for precision farming was developed in this work. The GraDeS solution was designed to incorporate a robotic platform capable of working autonomously on rugged terrain with steep slopes and various ground conditions. The system was tested in a pseudo-real scenario inside the laboratory using a structure with leaves and grape bunches to simulate a vineyard. The central core of this work was the development of a DL algorithm to track the grape bunches. Then, with the bunch position information, redirect the sprayer structure in the bunch direction with the pan and tilt movement.

The tests conducted with the designed sprayer proved two main points. First, the deep learning algorithm successfully identified and tracked grape bunches. Even with the structure movement and covering the bunches, the algorithm uses some techniques to keep track of the detected objects. Second, the robotic sprayer has considerable potential to increase spraying precision in specific regions, such as only spraying grape bunches. The compact size of the structure allows the application of the system in medium and small robots.

Nevertheless, several improvements for future work were identified. These improvements result from certain aspects which were noticed during this work.

First, producing a proper PCB for the mechanism from the hardware side would be interesting. Besides, using the PixelDriver3 has saved a tremendous amount of time in the electric connections of the driver and the microprocessor; however, the driver was not natively developed to control the mechanism of this dissertation work. The driver was modified manually to handle adding hall sensors and a DC motor to control the fan.

Moreover, updating the control code to perform the pan and tilt movements simultaneously can save precious milliseconds when aiming at a grape bunch. As the mechanism can be coupled in a robot with constant velocity, and the camera captures images accordingly to the robot's movement, the tool must be fast and precise not to lose any grape bunch in the spraying process. Also, the prototype developed in this dissertation uses the serial port to communicate between the software and the hardware. Using a communication protocol such as CAN can bring several advantages. It is a protocol that delivers reliability, robustness, high-speed transmission, scalability, simplicity, and low cost.

Furthermore, using a better camera, with more processing power and depth sensor, can be the leading improvement to be a functional product in the market. The control code depends on the robot's distance to the detection point. This distance directly affects the movement angle computation. A slight deviation in the measurements can accumulate giant errors in the results. In a vineyard, the terrain is not ideal and flat, so while moving, the camera distance will suffer from noise, affecting the movement of the sprayer. So, a camera with a depth sensor could use a variable distance value to perform the correct computation. Also, a camera with more processing power allows heavier CNN processing without losing the FPS rate, being more precise and accurate in tracking/detection.

Besides PETG being resistant and durable, it is very susceptible to UV damage. The material turns yellow and brittle after constant and prolonged exposure to UV radiation. Functionally, it does not impact the structure; however, it is interesting to use an aesthetically appealing material such as ASA for commercial purposes. This material is also durable and resistant to heat, impact, and weather degradation.

Finally, it is interesting to append more images to the dataset from the software part. Raising the dataset size with more grape bunch varieties and noise (such as leaves or trunks covering the bunch) could significantly improve the model accuracy. Also, try a new approach to compute the movement angles. Implementing an extended Kalman filter would be interesting to estimate and precisely predict the pan and tilt movement to the center of the detection.

This work validated an electric-based pan, tilt sprayer, and grape bunch tracking system. The results were positive, and the main objective of the work was surpassed. Applying the mentioned upgrades is necessary to deliver a fully functional product in the future.

Appendix A

Filament Material

A.1 Introduction

The sprayer will be susceptible to external conditions such as heat, sunbeam, dust, and rain. So choosing the best material that can handle these conditions is vital. In short, the material must be durable and resistant to moisture and temperature.

With brief research, the best materials in the market today for outdoor applications are PETG and ASA. This conclusion is due to the excellent resistance in external conditions, besides being durable.

A.1.1 PETG

PETG (Glycol-modified Polyethylene Terephthalate) is a variant of Polyethylene Terephthalate (PET), one of the world's most widely used plastics. While PET is valued for its high mechanical strength and moisture resistance, adding glycol during the polymerization process results in a more durable plastic, clearer and easier to handle [69].

PETG is one of the most durable filament materials on the market today. As previously stated, its durability and impact resistance outperform standard PET. It is heat-stable enough to withstand steam sterilization without deforming, making it a popular material for medical containers [69].

The main properties of this material are high strength, high impact resistance, heat stability, chemical stability, clarity, and susceptibility to UV damage. The main applications are in protective components, mechanical parts, and objects that will come into contact with food.

A.1.2 ASA

Acrylonitrile Styrene Acrylate (ASA) is a plastic remarkably similar to ABS. Instead of the butadiene component that gives ABS (Acetonitrile Butadiene Styrene) its more excellent impact resistance, ASA has acrylate integrated into its acrylonitrile-styrene matrix. ASA has approximately ten times the resilience to UV radiation and weathering compared to ABS due to acrylate [69].

In contrast to ABS, which turns yellow and brittle after prolonged UV exposure, ASA can preserve its glossy surface and mechanical strength. It is also resistant to heat and chemical destruction. Because of this distinguishing feature, ASA has become the material of choice for 3D-printed things intended for outdoor use. They could include road signs, outdoor adventure equipment, garden fittings, and vehicle parts [69].

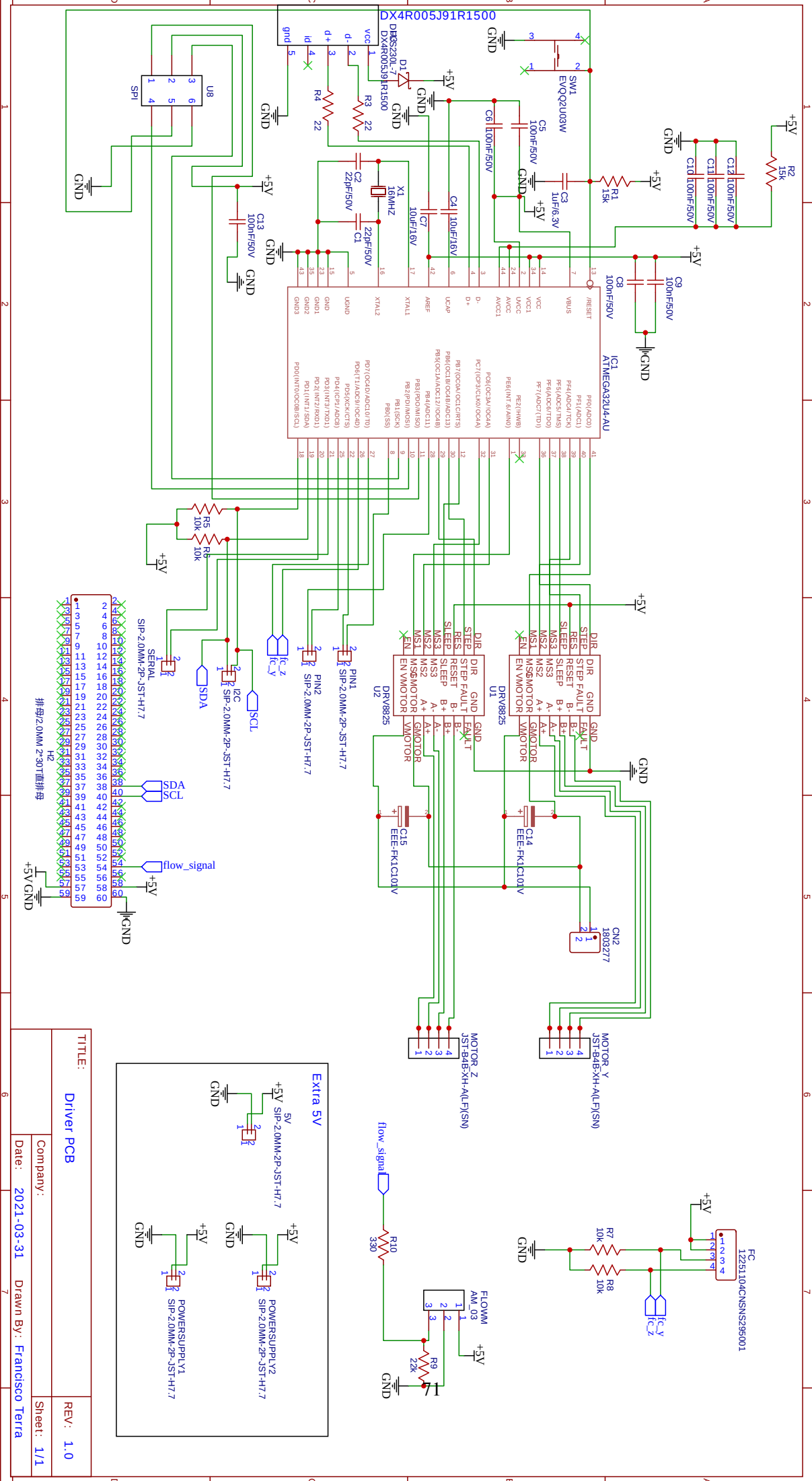
The main properties of this material are incredibly long-lasting, has good resistance to UV degradation and weathering, is less prone to stress cracking, has good impact resistance, is heat-stable, chemically stable, and is prone to stringing and warping. The main applications are garden fixtures, outdoor sports equipment, road signage, and automotive parts.

A.1.3 Material Choice

Taking into account both materials, the material chosen was PETG. This choice is due to the availability of the material in the laboratory, reducing the logistic problems with order and delivery time. Moreover, ASA filament is more expensive than PETG. The main advantage of ASA over PETG is having more resistance to UV light. However, this advantage is more aesthetic mainly than functional, keeping the material resistant to discoloration. That advantage does not justify the higher price. Furthermore, PETG is more recommended for use in mechanical parts. With all the mentioned factors, the model was printed with PETG filament.

Appendix B

PixelDriver3 Schematic



TITLE: Driver PCB	
Company:	REV: 1.0
Date: 2021-03-31	Drawn By: Francisco Terra
Sheet: 1/1	

References

- [1] André Rodrigues Baltazar, Filipe Neves dos Santos, António Paulo Moreira, António Valente, and José Boaventura Cunha. Smarter robotic sprayer system for precision agriculture. *Electronics*, 10(17), 2021. URL: <https://www.mdpi.com/2079-9292/10/17/2061>, doi:10.3390/electronics10172061.
- [2] Johnny D. Top 10 best backpack sprayers in 2022 reviews. Available at <https://www.10guider.com/best-backpack-sprayers/> (2022).
- [3] Sharon O’Keeffe. Amazone release auxiliary tank. Available at <https://www.farmonline.com.au/story/6521342/turning-your-tractor-into-a-self-propelled-sprayer/> (2019/12/06).
- [4] Flavia Geiger, Jan Bengtsson, Frank Berendse, Wolfgang W. Weisser, Mark Emerson, Manuel B. Morales, Piotr Ceryngier, Jaan Liira, Teja Tscharntke, Camilla Winqvist, Sönke Eggers, Riccardo Bommarco, Tomas Pärt, Vincent Bretagnolle, Manuel Plantegenest, Lars W. Clement, Christopher Dennis, Catherine Palmer, Juan J. Oñate, Irene Guerrero, Violetta Hawro, Tsipe Aavik, Carsten Thies, Andreas Flohre, Sebastian Hänke, Christina Fischer, Paul W. Goedhart, and Pablo Inchausti. Persistent negative effects of pesticides on biodiversity and biological control potential on european farmland. *Basic and Applied Ecology*, 11(2):97–105, 2010. URL: <https://www.sciencedirect.com/science/article/pii/S1439179109001388>, doi:<https://doi.org/10.1016/j.baae.2009.12.001>.
- [5] M.E.R. Paice, P.C.H. Miller, and J.D. Bodle. An experimental sprayer for the spatially selective application of herbicides. *Journal of Agricultural Engineering Research*, 60(2):107–116, 1995. URL: <https://www.sciencedirect.com/science/article/pii/S0021863485710050>, doi:<https://doi.org/10.1006/jaer.1995.1005>.
- [6] Hossein Maghsoudi, Saeid Minaei, Barat Ghobadian, and Hassan Masoudi. Ultrasonic sensing of pistachio canopy for low-volume precision spraying. *Computers and Electronics in Agriculture*, 112:149–160, 2015. Precision Agriculture. URL: <https://www.sciencedirect.com/science/article/pii/S0168169914003251>, doi:<https://doi.org/10.1016/j.compag.2014.12.015>.
- [7] Roberto Oberti, Massimo Marchi, Paolo Tirelli, Aldo Calcante, Marcello Iriti, Emanuele Tona, Marko Hočevár, Joerg Baur, Julian Pfaff, Christoph Schütz, and Heinz Ulbrich. Selective spraying of grapevines for disease control using a modular agricultural robot. *Biosystems Engineering*, 146:203–215, 2016. Special Issue: Advances in Robotic Agriculture for Crops. URL: <https://www.sciencedirect.com/science/article/pii/S1537511015001865>, doi:<https://doi.org/10.1016/j.biosystemseng.2015.12.004>.

- [8] R. Oberti, M. Marchi, P. Tirelli, A. Calcante, M. Iriti, M. Hoaevan, J. Baur, J. Pfaff, C. Schütz, and H. Ulbrich. Selective spraying of grapevineâ€™s diseases by a modular agricultural robot. *Journal of Agricultural Engineering*, 44(s2), Sep. 2013. URL: <https://www.agroengineering.org/index.php/jae/article/view/jae.2013.s2.e29>, doi:10.4081/jae.2013.271.
- [9] Datagy. Support vector machines (svm) in python with sklearn. Available at <https://datagy.io/python-support-vector-machines/> (2022/02/25).
- [10] Mayank Chandra and Sarabjeet Bedi. Survey on svm and their application in image classification. *International Journal of Information Technology*, 13, 01 2018. doi:10.1007/s41870-017-0080-1.
- [11] Theodoros Evgeniou and Massimiliano Pontil. Support vector machines: Theory and applications. volume 2049, pages 249–257, 09 2001. doi:10.1007/3-540-44673-7_12.
- [12] Ahmed Imam. Introduction to deep-learning. Available at <https://medium.com/analytics-vidhya/introduction-to-deep-learning-e3a7899a04a3> (2020/06/19).
- [13] Ksenia Sorokina. Image classification with convolutional neural networks. Available at <https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8> (2017/11/20).
- [14] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing.
- [15] Roboflow. What is mobilenet ssd v2? Available at <https://roboflow.com/model/mobilenet-ssd-v2> last accessed at (2023/05/26).
- [16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi:10.1109/CVPR.2016.91.
- [17] Ultralytics. YOLOv5. Available at <https://github.com/ultralytics/yolov5> last accessed at (2023/03/02).
- [18] DataCamp. YOLO object detection explained. Available at <https://www.datacamp.com/blog/yolo-object-detection-explained> last accessed at (2023/03/02).
- [19] Boaz Shmueli. Multi-class metrics made simple, part ii: the f1-score. Available at <https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-i-the-f1-score-ebe8b2c2ca1> last accessed at (2023/01/07).
- [20] Stephen Allwright. What is a good f1 score and how do i interpret it? Available at <https://stephenallwright.com/good-f1-score/> last accessed at (2023/01/07).
- [21] Harikrishnan N B. Confusion matrix, accuracy, precision, recall, f1 score. Available at <https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd> last accessed at (2023/01/07).

- [22] Luqman Ali, Fady Alnajjar, Medha Mohan Ambali Parambil, Mohammad Issam Younes, Ziad Ismail Abdelhalim, and Hamad Aljassmi. Development of yolov5-based real-time smart monitoring system for increasing lab safety awareness in educational institutions. *Sensors*, 22(22), 2022. URL: <https://www.mdpi.com/1424-8220/22/22/8820>, doi: 10.3390/s22228820.
- [23] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006. doi:10.1109/TPAMI.2006.244.
- [24] Scikit-image. Local binary pattern for texture classification. Available at https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_local_binary_pattern.html last accessed at (2022/12/23).
- [25] Vijay Trivedi, Piyush Shukla, and Anjana Pandey. Automatic segmentation of plant leaves disease using min-max hue histogram and k-mean clustering. *Multimedia Tools and Applications*, 81, 06 2022. doi:10.1007/s11042-022-12518-7.
- [26] Fillipe de Souza, Eduardo Valle, Guillermo Chávez, and Arnaldo Araújo. Hue histograms to spatiotemporal local features for action recognition. *CoRR*, abs/1104.3742, 01 2011.
- [27] Lukas Stratmann. Hsv. Available at <https://color.lukas-stratmann.com/color-systems/hsv.html> last accessed at (2022/12/23).
- [28] Mounir Louhaichi, Michael Borman, and Douglas Johnson. Spatially located platform and aerial photography for documentation of grazing impacts on wheat. *Geocarto International*, 16, 04 2001. doi:10.1080/10106040108542184.
- [29] Phenospex. Greenness (green leaf index). Available at <https://phenospex.helpdocs.com/plant-parameters/greenness#:~:text=Value%20Range%3A%20-1%E2%80%933%20The%20greenness%20index%2C%20also, and%20blue%29.%20It%20is%20computed%20as%20%282%2AG-R-B%29%2F%20%282%2AR%2BG%2BB%29.> last accessed at (2022/12/24).
- [30] Jadson Silva, Yenê Paz, Pedro Silva, João Pereira, and Ana Candeias. Índices de vegetação do sensoriamento remoto para processamento de imagens na faixa do visível (rgb). 9:228–239, 12 2019. doi:10.29150/jhrs.v9.4.p228-239.
- [31] Juliane Bendig, Kang Yu, Helge Aasen, Andreas Bolten, Simon Bennertz, Janis Broscheit, Martin L. Gnyp, and Georg Bareth. Combining uav-based plant height from crop surface models, visible, and near infrared vegetation indices for biomass monitoring in barley. *International Journal of Applied Earth Observation and Geoinformation*, 39:79–87, 2015. URL: <https://www.sciencedirect.com/science/article/pii/S0303243415000446>, doi:<https://doi.org/10.1016/j.jag.2015.02.012>.
- [32] Compton J. Tucker. Red and photographic infrared linear combinations for monitoring vegetation. *Remote Sensing of Environment*, 8(2):127–150, 1979. URL: <https://www.sciencedirect.com/science/article/pii/0034425779900130>, doi:[https://doi.org/10.1016/0034-4257\(79\)90013-0](https://doi.org/10.1016/0034-4257(79)90013-0).
- [33] Esmael Hamuda, Martin Glavin, and Edward Jones. A survey of image processing techniques for plant extraction and segmentation in the field. *Computers and Electronics in Agriculture*,

- 125:184–199, 2016. URL: <https://www.sciencedirect.com/science/article/pii/S0168169916301557>, doi:<https://doi.org/10.1016/j.compag.2016.04.024>.
- [34] SoftFarm. Vegetation indices ndvi, evi, gndvi, cvi, true color. Available at <https://www.soft.farm/en/blog/vegetation-indices-ndvi-evi-gndvi-cvi-true-color-140#:~:text=GNDVI%20-%20green%20normalized%20difference%20vegetation%20index.%20It,in%20the%20range%20from%200.54%20to%200.57%20microns>. last accessed at (2022/12/24).
- [35] Richard Escadafal and Alfredo Huete. Etude des propriétés spectrales des sols arides appliquée à l'amélioration des indices de végétation obtenus par télédétection. 01 1991.
- [36] A. Bannari, D. Morin, F. Bonn, and A. R. Huete. A review of vegetation indices. *Remote Sensing Reviews*, 13(1-2):95–120, 1995. URL: <https://doi.org/10.1080/02757259509532298>, arXiv:<https://doi.org/10.1080/02757259509532298>, doi:[10.1080/02757259509532298](https://doi.org/10.1080/02757259509532298).
- [37] Color indices for weed identification under various soil, residue, and lighting conditions. *Transactions of the ASABE*, 38(1):259–269, 1995. doi:[10.13031/2013.27838](https://doi.org/10.13031/2013.27838).
- [38] George E. Meyer, Timothy W. Hindman, and Koppolu Laksmi. Machine vision detection parameters for plant species identification. In *Other Conferences*, 1999.
- [39] George.E Meyer, Joao Camargo Neto, David D Jones, and Timothy W Hindman. Intensified fuzzy clusters for classifying plant, soil, and residue regions of interest from color images. *Computers and Electronics in Agriculture*, 42(3):161–180, 2004. URL: <https://www.sciencedirect.com/science/article/pii/S0168169903001224>, doi:<https://doi.org/10.1016/j.compag.2003.08.002>.
- [40] Xavier P. Burgos-Artizzu, Angela Ribeiro, Maria Guijarro, and Gonzalo Pajares. Real-time image processing for crop/weed discrimination in maize fields. *Computers and Electronics in Agriculture*, 75(2):337–346, 2011. URL: <https://www.sciencedirect.com/science/article/pii/S0168169910002620>, doi:<https://doi.org/10.1016/j.compag.2010.12.011>.
- [41] T. Kataoka, T. Kaneko, H. Okamoto, and S. Hata. Crop growth estimation system using machine vision. In *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, volume 2, pages b1079–b1083 vol.2, 2003. doi:[10.1109/AIM.2003.1225492](https://doi.org/10.1109/AIM.2003.1225492).
- [42] Tony Hague, N. D. Tillett, and Helen C. Wheeler. Automated crop and weed monitoring in widely spaced cereals. *Precision Agriculture*, 7:21–32, 2006.
- [43] Kasian Myagila and Hassan Kilavo. A comparative study on performance of svm and cnn in tanzania sign language translation using image recognition. *Applied Artificial Intelligence*, 36(1):2005297, 2022. URL: <https://doi.org/10.1080/08839514.2021.2005297>, arXiv:<https://doi.org/10.1080/08839514.2021.2005297>, doi:[10.1080/08839514.2021.2005297](https://doi.org/10.1080/08839514.2021.2005297).
- [44] Özgür Kaplan and Ediz Saykol. Comparison of support vector machines and deep learning for vehicle detection. 11 2018.

- [45] Chaoqun Shan. Vehicle-detection. Available at <https://github.com/uranus4ever/Vehicle-Detection> last accessed at (2023/03/02).
- [46] Huipeng Li, Changyong Li, Guibin Li, and Lixin Chen. A real-time table grape detection method based on improved yolov4-tiny network in complex background. *Biosystems Engineering*, 212:347–359, 2021. URL: <https://www.sciencedirect.com/science/article/pii/S1537511021002786>, doi:<https://doi.org/10.1016/j.biosystemseng.2021.11.011>.
- [47] Smlease. What is degree of freedom (dof) in mechanics. Available at <https://www.smlease.com/entries/mechanism/what-is-degree-of-freedom-dof-in-mechanics/> last accessed at (2022/12/29).
- [48] Mecharithm. Everything about the degrees of freedom of a robot. Available at <https://www.mecharithm.com/degrees-of-freedom-of-a-robot/> last accessed at (2022/12/29).
- [49] John Clarke. Six-degrees of freedom. Available at <https://johnclarkeonline.com/2011/12/18/six-degrees-of-freedom/> last accessed at (2022/12/29).
- [50] Niall Henry. Understanding robot coordinate frames and points. Available at <https://developercenter.robotstudio.com/api/robotstudio/articles/Concepts/Math/Coordinate-Systems.html> last accessed at (2023/223/06).
- [51] ABB. Coordinate systems. Available at <https://www.solisplc.com/tutorials/robot-coordinate-frames-and-points> last accessed at (2023/223/06).
- [52] Jan Bodez. Robotics kinematics and dynamics/description of position and orientation. Available at https://en.wikibooks.org/wiki/Robotics_Kinematics_and_Dynamics/Description_of_Position_and_Orientation last accessed at (2023/04/04).
- [53] Reza Jazar. *Theory of Applied Robotics*. 01 2007. doi:10.1007/978-0-387-68964-7.
- [54] Howie Choset. An introduction to robot kinematics. Available at https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/16311/www/current/lecture/Kinematics_final.pdf last accessed at (2023/04/04).
- [55] Peter Corke. *Robotics, Vision and Control - Fundamental Algorithms in MATLAB®*, volume 73. 01 2011. doi:10.1007/978-3-642-20144-8.
- [56] R Nave. Hall effect. Available at <http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/Hall.html> last accessed at (2023/03/23).
- [57] Electronics Tutorials. Hall effect sensor. Available at <https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html> last accessed at (2023/03/23).
- [58] Elta Europe. Cam and crank sensors. Available at <https://eltaeurope.com/engine-management/crankshaft-sensors-camshaft/> last accessed at (2023/03/23).
- [59] Luigi Uccello. Grape bunch detector 2 dataset. https://universe.roboflow.com/luigi-uccello-mpjt9/grape-bunch-detector_2, feb 2023. visited on 2023-03-23. URL: https://universe.roboflow.com/luigi-uccello-mpjt9/grape-bunch-detector_2.

- [60] Grape detection. label map dataset. <https://universe.roboflow.com/grape-detection/label-map>, mar 2022. visited on 2023-03-23. URL: <https://universe.roboflow.com/grape-detection/label-map>.
- [61] Glenn Jocher. Train custom data. Available at <https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data> last accessed at (2023/03/06).
- [62] Ultralytics. YOLOv8. Available at <https://github.com/ultralytics/ultralytics> last accessed at (2023/03/17).
- [63] Valentin Belissen. *From Sign Recognition to Automatic Sign Language Understanding : Addressing the Non-Conventionalized Units*. PhD thesis, 11 2020.
- [64] Luxonis. depthai-hardware. Available at https://github.com/luxonis/depthai-hardware/blob/master/BW1093_USB3C/Datasheet/OAK-1_Datasheet.pdf last accessed at (2023/03/22).
- [65] Benne. How to control a stepper motor with drv8825 driver and arduino. Available at <https://www.makerguides.com/drv8825-stepper-motor-driver-arduino-tutorial/> last accessed at (2023/03/29).
- [66] Francisco Terra, Leandro Rodrigues, Sandro Magalhães, Filipe Santos, Pedro Moura, and Mário Cunha. Pixelcrobot, a cartesian multitask platform for microfarms automation. In *2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA)*, pages 382–387, 2021. doi:10.1109/IRIA53009.2021.9588786.
- [67] ARDUINO TEAM. Capture cinematic shots with this motorized slider and pan-tilt camera mount. Available at <https://blog.arduino.cc/2020/07/06/capture-cinematic-shots-with-this-slider-and-pan-tilt-camera-assembly/> last accessed at (2023/223/06).
- [68] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015. doi:10.1109/TPAMI.2014.2345390.
- [69] Joseph Flynt. All 3d printing filaments types explained: Detailed guide. Available at <https://3dinsider.com/3d-printing-filaments-types/> last accessed at (2023/03/30).