Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Robotic Systems Lab
Prof. Dr. Marco Hutter

Master Thesis

Autonomous Weed Control Unit for Sustainable Agriculture

Spring Term 2024

Supervised by:
Martin Bauer

Author:
Jan-Marco Haldemann

Declaration of Originality

I hereby submit the written work entitled:

Autonomous Weed Control Unit for Sustainable Agriculture

- ☐ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it[1]. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies[2].
- ☐ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it[1]. Suggestions from the supervisor regarding language and content are excepted. I used and cited generative artificial intelligence technologies[2].
- ■ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it[1]. Suggestions from the supervisor regarding language and content are excepted. I used generative artificial intelligence technologies[2]. In consultation with the supervisor, I did not cite them.

Author(s)

Jan-Marco	Haldemann

Student supervisor(s)

Martin	Bauer

Supervising lecturer

Marco	Hutter

With the signature, I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on The citation conventions usual to the discipline in question here have been respected. I have documented all methods, data, and processes truthfully and fully. Additionally, I have mentioned all persons who were significant facilitators of the work.

The above-written work may be tested electronically for plagiarism.

Place and date	Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

[2] E.g. ChatGPT, DALL-E 2, Google Bard

Contents

Acknowledgements **v**

Abstract **vii**

1 Introduction **1**
1.1 Background and Motivation 1
1.2 Project Overview 2
1.2.1 Objective 2
1.2.2 Workspace 2
1.2.3 Requirements 2
1.2.4 Open Field Automation 4
1.3 Thesis Structure 4

2 Literature Review **5**
2.1 Precision Weed Control 5
2.2 Weed Detection 6

3 Hardware Architecture **9**
3.1 System Overview 9
3.2 Robot Arm 9
3.2.1 Design Considerations 9
3.2.2 Arm Dimensions 12
3.2.3 Revolute Joint Motors 13
3.2.4 Linear Axis 14
3.2.5 Linkages 15
3.3 Single-board computer 15
3.4 RGB-D camera 16
3.5 Mechanical Structure 17
3.6 Illumination 17
3.7 Power Supply 17

4 Software Implementation **21**
4.1 System Overview 21
4.1.1 Robot Operating System 21
4.2 Robot Arm Motion Control 21
4.2.1 ros2_control 23
4.2.2 MoveIt 2 24
4.3 Weed Detection 26
4.3.1 Image Capture 26
4.3.2 Plant Segmentation 26
4.3.3 Finding Connected Components 28
4.3.4 Plant Classification 31

4.4 Network Training 32
4.4.1 Dataset 32
4.4.2 Architecture and Training Process 33
4.4.3 Evaluation Metrics 34
4.4.4 Deployment 34
4.5 System Integration 34

5 Results **37**
5.1 Performance of Weed Classification 37
5.2 Visual Comparison of Weed Detection 38
5.3 Positioning Accuracy of the System 41
5.4 Positioning Speed of the Robot Arm 41

6 Discussion **43**

7 Conclusion and Future Work **45**
7.1 Future Work 45
7.2 Closing Remarks 46

Bibliography **52**

8 Datasheets **53**

A Hyperparameters **58**

Acknowledgements

I would like to extend my sincere thanks to Prof. Dr. Marco Hutter for giving me the opportunity to undertake this thesis at the Robotics System Lab.
My deepest gratitude goes to my supervisor, Martin Bauer, for his invaluable guidance, patience, and constant support throughout this project. His insights and expertise were instrumental in shaping both the direction and quality of this work.
I am also grateful to Manfred Muhr for his exceptional assistance with the construction and assembly of the robot's mechanical components.
Finally, I would like to thank my family and friends for their unwavering support and encouragement, without which this thesis would not have been possible.

Abstract

This thesis presents the development of an autonomous robotic system designed to detect and eliminate weeds in agricultural fields, aiming to promote more sustainable farming practices. The primary objective was to create a cost-effective and efficient solution that reduces reliance on herbicides and minimizes soil disturbances. The system comprises a 3-DOF robot arm and a weed detection algorithm capable of localizing and classifying weeds effectively. The robot arm features a unique design with a linear axis and a two-link planar arm attached to the carriage, allowing it to navigate between crop rows while minimizing interference with crop leaves. Motion control for the robot arm was implemented, demonstrating the ability to transition between weed positions in less than one second. The arm is equipped with an electrode at its end-effector to electrocute detected weeds, enabling autonomous weed elimination.

The weed detection algorithm employs a color-based vegetation segmentation followed by classification using a neural network. To overcome challenges associated with threshold-based segmentation, consistent lighting conditions were maintained within the workspace, and 3D clustering was utilized for connected component analysis. A vision transformer was trained using publicly available datasets of plants and weeds. Through qualitative analysis, we compared our approach to an end-to-end object detection network, demonstrating that our method is more robust and adaptable to unknown field conditions.

The system achieved a positioning accuracy of under 1 cm by integrating the RGB-D camera and robot arm. Initial field tests yielded promising results, suggesting that this autonomous weed control unit has practical potential in real-world agricultural applications. This work advances precision agriculture, laying the groundwork for more sustainable and efficient weed management solutions.

Acronyms and Abbreviations

IR	Infrared
ExG	Excess Green Index
NDVI	Normalized Difference Vegetation Index
DOF	Degrees of freedom
ToF	Time-of-flight
ROS	Robot Operating System
GPU	Graphics Processing Unit
URDF	Unified Robot Description Format
DBSCAN	Density-based spatial clustering of applications with noise

Chapter 1

Introduction

1.1 Background and Motivation

Agriculture is one of the oldest and most essential industries, forming the foundation of the global economy. However, the sector faces numerous challenges, including the need to increase productivity while reducing environmental impact. One of the critical issues in agriculture is weed management. Weeds compete with productive crops for resources such as light, water, and nutrients, making it necessary to control their growth to protect crop yields. Traditionally, farmers have relied heavily on chemical herbicides for weed control.

As the industry increasingly shifts towards more sustainable farming methods, such as organic and conservation agriculture, the limitations of conventional weed management are becoming more apparent. Organic farming avoids herbicides altogether, relying instead on mechanical techniques like tillage to control weeds. While this aligns with the principles of organic agriculture, it brings its own problems, such as the risk of soil erosion and the formation of hardpans, both of which can degrade soil health over time. Tillage also releases CO_2 into the atmosphere, contributing to greenhouse gas emissions [1].

On the other hand, conservation agriculture focuses on minimal soil disturbance through practices like permanent soil cover and no-till farming [2]. While these methods help preserve soil structure and reduce erosion, they rely heavily on herbicides for effective weed suppression. The over-reliance on herbicides has led to the emergence of herbicide-resistant weeds, which are a growing concern in modern agriculture. Moreover, herbicide use can lead to soil and water contamination, raising additional environmental concerns [3].

The advent of precision agriculture has opened up new possibilities for improving farming practices. One promising innovation in this space is the use of autonomous robots for tasks such as weed control. These robots have the potential to significantly reduce the need for chemical inputs, thereby lessening environmental impact and promoting more sustainable agriculture [4]. By integrating technologies like machine learning and robotics, it is possible to design systems that selectively target and eliminate weeds while preserving crops and preserving soil health.

Despite the advancements in agricultural robotics and computer vision, creating an autonomous system that can accurately identify and eliminate weeds without damaging crops remains a significant challenge [5, 6]. Developing a solution that is reliable, efficient, and cost-effective for managing weeds autonomously is the driving force behind this research.

1.2 Project Overview

1.2.1 Objective

The goal of this thesis is to build a fully autonomous weed control unit that applies electric pulses directly to individual weed plants in agricultural fields. This system will enable weed control without the use of herbicides while also minimizing soil disturbances.
A camera system is employed to locate individual weeds, after which a robotic arm positions an electrode at the identified locations. The electrode is connected to a high-voltage impulse generator, capable of producing voltages up to 30 kV. The resulting electric impulses efficiently kill the weeds.
The design of the impulse generator falls outside the scope of this work. Instead, this thesis focuses on the design, development, and programming of the robotic arm, as well as the implementation of the weed detection algorithm. The weed control unit will be mounted on a vehicle designed to navigate autonomously along crop rows. This vehicle is currently being developed at the Eastern Switzerland University of Applied Sciences.

1.2.2 Workspace

The workspace of the weed control unit is illustrated in Figure 1.1 and Figure 1.2. The system should treat two crop rows simultaneously, targeting only intra-row weeds. Inter-row weeds can be effectively managed through simpler techniques such as mulching. Mulching, which involves covering the soil with organic material, is an easy and efficient way to inhibit the growth of unwanted plants. However, this method cannot be applied to intra-row weeds, as it would also prevent the growth of the main crop. Therefore, the weeds growing close to the crops must be eliminated individually.
The autonomous vehicle to which the weed control unit is attached to, will remain stationary during the weed detection and removal process. After each treatment cycle, the vehicle moves a predetermined distance, corresponding to the length of the robot's workspace.
To minimize the risk of interfering with the crop, only one side of each row is treated at a time. As a result, the vehicle has to pass over each row twice, to ensure that weeds on both sides of the crop are removed.

1.2.3 Requirements

The key requirements for the weed control unit are as follows:

- **Autonomous operation**: The system operates without human intervention.
- **Weed detection**: A camera system is used to detect weeds within the workspace.
  - **Detection accuracy**: The system accurately distinguishes between crops and weeds with a low number of false negatives.
  - **3D localization**: The 3D positions of the detected weeds is determined.
  - **Growth stages**: The detection works from the emergence stage of the crops and weeds through to the multi-leaf stage.
  - **Illumination**: The system operates effectively under varying environmental lighting conditions.

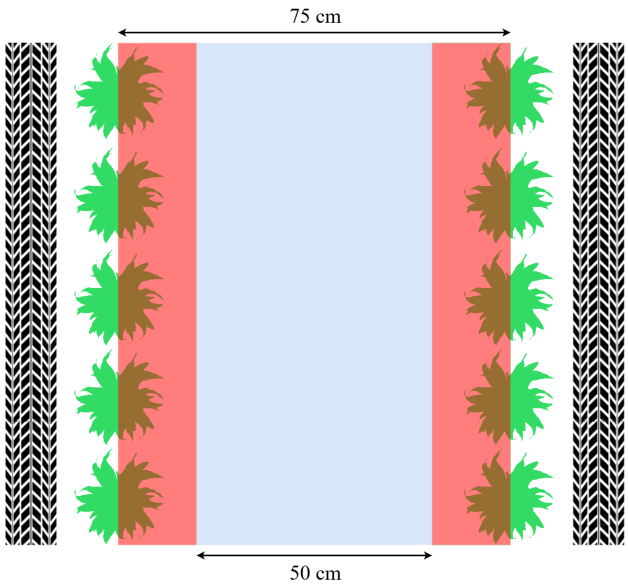


Figure 1.1: Top view of the robot's workspace. The crops are typically planted in rows spaced 75 cm apart. The area marked in red indicates the section that should be targeted by the weed control unit. Only one side of each crop row is treated. The inter-row weeds within the blue-marked area are managed using mulching.

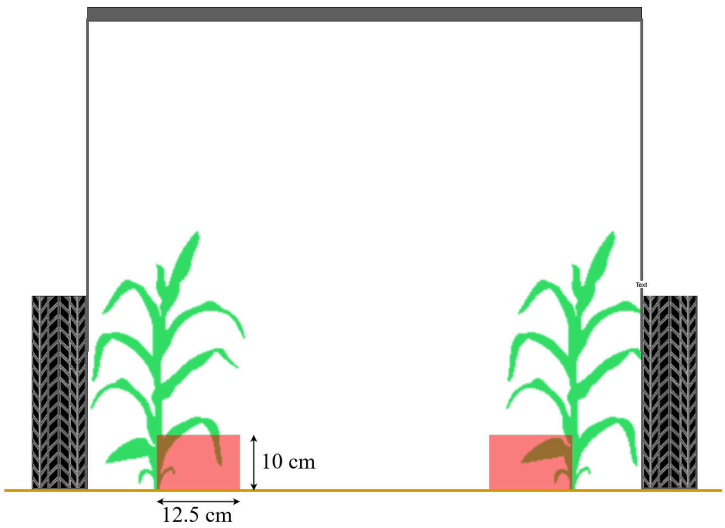


Figure 1.2: Side view of the robot's workspace. The area marked in red shows where weeds have to be killed. We expect to treat weeds up to a height of 10 cm.

- **Robotic arm**: A robotic arm positions the weeding tool (electrode) at the identified locations.
  - **Payload**: The arm supports a payload of up to 1 kg.
  - **Speed**: The arm moves from one weed to the next within one second.
- **Position Accuracy**: The end-effector of the arm is, on average, no more than 5 mm away from the detected weed stem.
- **Plant Interference**: The system avoids damaging plant leaves of the crop.
- **System integration**: The unit is able to communicate with the vehicle by accepting a start command and sending a completion signal once the process is finished.

For this thesis, the primary focus will be on fields with corn (Zea mays) and sugar beet (Beta vulgaris). However, the weed detection algorithm should be adaptable to other crops as well.

1.2.4 Open Field Automation

This work is part of the Open Field Automation (OFA) initiative, which focuses on developing open-source tools for efficient, organic, and regenerative farming. The code and materials developed as part of this thesis are available on the OFA GitHub page at `https://github.com/OpenFieldAutomation-OFA`.

1.3 Thesis Structure

The thesis is structured as follows:

- **Chapter 2** reviews the current state of research in autonomous weed control, with a focus on positioning mechanisms for weeding tools and weed detection techniques.
- **Chapter 3** details the hardware architecture of the weed control unit, including the design and decision-making process behind the selection and integration of its various components.
- **Chapter 4** explores the software implemented on the main computer, covering aspects of motion control and weed detection.
- **Chapter 5** presents the results of the weed classification and detection, as well as the performance evaluation of the robot arm.
- **Chapter 6** discusses the findings, providing an analysis of the results and their implications.
- **Chapter 6** concludes the thesis by summarizing the key findings and suggests possible future improvements to enhance the system's performance and adaptability.

Chapter 2

Literature Review

Autonomous systems for intra-row weed control typically consist of three key components: a vision system for weed detection and identification, a precision weeding mechanism to remove the weeds, and a guidance system to navigate the robot along crop rows. This literature review will focus on the weeding mechanism and the weed detection, as the guidance system is beyond the scope of this thesis.

2.1 Precision Weed Control

Mechanical systems are one common approach for the precise removal of weeds in agricultural fields. Various designs have been developed to balance effective weed elimination with minimal crop damage. Åstrand and Baerveldt [7] introduced a rotating wheel mechanism that is raised and lowered based on real-time plant identification. Tillett et al. [8] advanced this concept by employing a rotating disc with a cut-out section, enabling crop plants to pass through undamaged without having to lift the disc. The disc rotation was synchronized to match the spacing of the plants. Similarly, Quan et al. [9] designed weeding knives with gaps to avoid detected crops. Pérez-Ruíz et al. [10] and Bawden et al. [11] developed systems using pneumatically actuated hoes to uproot weeds between crop plants. In a different approach, Michaels et al. [12] used a cordless nailer attached to a delta robot to stamp weeds into the ground, limiting their access to sunlight and water. Visentin et al. [13] implemented a gantry robot equipped with a gripper to physically grasp and remove weeds.

Precision chemical spraying offers an alternative to mechanical weed removal with the advantage of no soil disturbance. These systems, while advantageous in that respect, pose environmental risks due to herbicide use. However, compared to traditional herbicide use, precision herbicide spraying reduces the amount of chemicals and makes it possible to apply herbicides on crops that lack selective herbicides. These systems typically feature an array of nozzles controlled by solenoid valves, activated only when weeds are detected in a particular area [14, 15, 16]. Despite these advancements, there remains a risk of herbicide splashing onto crops, potentially reducing yields [4].

Laser-based systems represent another approach, offering minimal soil disturbance without relying on herbicides. Multiple studies have demonstrated that lasers can effectively control weeds [17, 18, 19, 20]. However, the high cost [21] and significant energy requirements [22] of laser systems make them less practical for widespread adoption.

A more energy-efficient and cost-effective solution is to use electrical pulses to eliminate weeds. This method involves placing an electrode near the plant and delivering

short, high-voltage pulses, which damage the plant tissue and cause mortality [23]. Blasco et al. [24] utilized a delta robot to deliver an electrical discharge of 15 kV, successfully eliminating weeds in lettuce crops. Building on this concept, Abdelghafour et al. [25] developed a similar system but with higher voltage discharges of 40-60 kV. Their study highlighted that electrocution effectiveness depends on both the weed species and growth stage, with early-stage weeds being particularly susceptible to electrical destruction.

In summary, precision in-row weed management can be categorized into mechanical, chemical, laser-based, or electrical systems. In this thesis, we are using electrical discharges as the weeding mechanism due to their energy efficiency, lower cost compared to lasers, and lack of reliance on herbicides or soil disturbance.

For positioning the weeding tool, which is the focus of this thesis, various approaches have been used. Some methods, such as lasers or herbicide spraying arrays, do not require a manipulator. For mechanical and electric treatments, positioning mechanisms such as XYZ gantry robots [13] and delta robots [24, 12, 25] have been explored. A few studies have also employed off-the-shelf serial manipulators [26, 27]. However, to our knowledge, no research specifically targets robot arm designs for intra-row weed control that aim to minimize plant interference. This work aims to address that gap.

2.2 Weed Detection

All the weed control systems mentioned in the previous section require an effective method to distinguish crops from weeds. While some systems, like the one developed by Pérez-Ruíz et al. [10], rely solely on crop planting patterns and odometry, this approach requires human supervision. For a fully autonomous solution, a camera to capture field images is essential. Processing these images to detect weeds remains one of the most challenging aspects of autonomous weed control, and numerous approaches have been explored over time.

Early weed detection systems primarily employed basic image processing techniques, following a typical three-step workflow: segmentation, feature extraction, and plant classification [28].

The first step involves segmenting the image to differentiate plants from the background. This is typically done by converting the RGB image into a grayscale image using a vegetation index, which highlights plant material against the soil. One of the most widely adopted indices is the Excess Green Index (ExG) introduced by Woebbecke et al. [29], which is derived from RGB images. Other color-based indices, like the Excess Green minus Excess Red Index (ExGR) and the Color Index of Vegetation Extraction (CIVE), were reviewed and compared by Hamuda et al. [30]. If infrared (IR) images are available, the Normalized Difference Vegetation Index (NDVI) is often used for better separation between vegetation and non-vegetation elements [31, 32]. After calculating the vegetation index, either a fixed threshold or Otsu's method is applied to the image turning it into a binary format [28]. In the next step, features are extracted from the connected components of the binary image. These features are typically shape-based, such as the perimeter, area, or moment invariants of the plant masks [33, 34, 35]. The extracted features are then input into traditional machine learning algorithms, like support vector machines or random forests, to classify the plants as either crops or weeds. Additionally, information about the spatial planting pattern can be integrated to enhance classification accuracy [36, 37].

With the advancements in deep learning, more powerful methods for weed detection have emerged. One approach is to modify the traditional workflow by removing the manual feature extraction step and using a neural network to directly classify

segmented plants. This method has been implemented by Milioto et al. [38] and Visentin et al. [13]. Dyrmann et al. [39] demonstrated that convolutional neural networks, given enough training data, outperform traditional classifiers in terms of accuracy.

A more advanced approach involves the use of end-to-end deep learning models. These models remove the need for threshold-based pre-segmentation, which is sensitive to changes in illumination and cannot deal with overlapping plants [40]. Several network architectures have been proposed for this task. Semantic segmentation networks [41, 40, 42] classify every pixel in an image as either crop, weed or soil, while object detection networks [43, 44, 9, 45] locate and classify plants using bounding boxes. Instance segmentation networks [46, 47] go further by providing both a segmentation mask and the location of individual plants.

One major challenge with deep learning models is their sensitivity to domain shifts, where the model's performance drops when there are differences between the training dataset and the target domain. This issue is particularly pronounced in precision agriculture, where large-scale, publicly available training datasets are limited [48] and field conditions vary significantly due to factors such as soil type weed species diversity. To address this, one potential solution is the creation of larger, more diverse datasets that capture a broader range of field conditions. Alternatively, domain adaptation techniques, such as those proposed by [49], can be used to improve model robustness across different environments.

In this work, we aim to develop a weed detection algorithm using only publicly available datasets, without relying on data specific to our field environment. Through this research, we hope to contribute to the development of more robust approaches for handling weed detection across diverse agricultural environments.

Chapter 3

Hardware Architecture

3.1 System Overview

The weed control unit is composed of several key components: a three degrees of freedom (DOF) robot arm, a high-voltage impulse generator connected to an electrode at the tip of the arm, a RGB-D camera, LED strips for illumination, a single-board computer communicating with all components, and batteries that power the entire system. An overview of the hardware is shown in Figure 3.1.
In the following, we will describe the design and hardware decisions for each component in more detail.

3.2 Robot Arm

To position the electrode close to the weeds, a custom-designed robot arm was developed. The main design requirement was that the end effector must be able to reach all areas within the defined workspace (see subsection 1.2.2) while minimizing interference with the crop. Avoiding collisions with the crop is essential to prevent leaf damage, which could reduce yield or even kill the plant in severe cases. This is especially difficult in later growth stages of the crop, as illustrated in Figure 3.2.
The design of the robot arm was limited to three degrees of freedom because the orientation of the electrode is not critical; as long as it is positioned near the weed, the spark discharge will effectively reach the target.

3.2.1 Design Considerations

One potential design option was a Cartesian robot. These robots are easy to control and can cover a large workspace using a gantry system. However, the requirement for a stiff, level frame makes this design unsuitable for mobile robots. Furthermore, approaching the ground vertically increases the risk of collision with crops, as shown in Figure 3.2 (a).
Another option was a delta robot, as illustrated in Figure 3.2 b. The main advantage of a parallel kinematic structure is the low amount of moved masses which minimizes reaction forces on the mobile platform. However, the potential collisions with plants led us to explore alternative designs.
Two more design options were considered, illustrated in Figure 3.2 (c) and (d). Both involve a linear axis positioned in the middle between the two crop rows, parallel to the vehicle's driving direction. The first option has a revolute joint and a linear actuator attached to the axis, while the second one utilizes two revolute

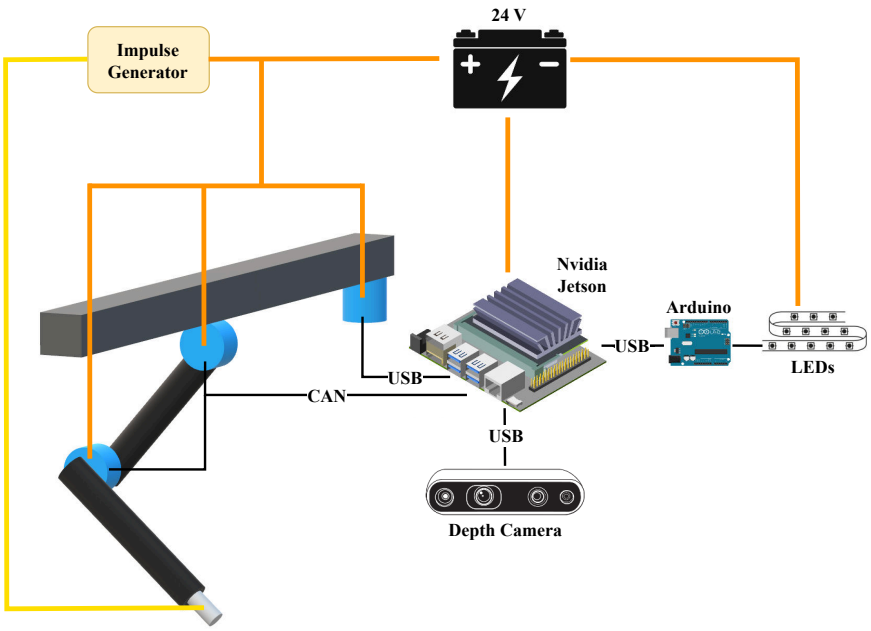
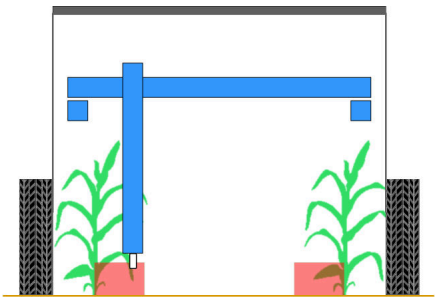
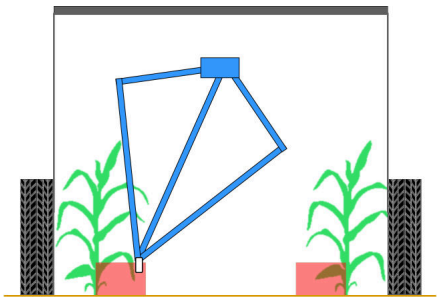


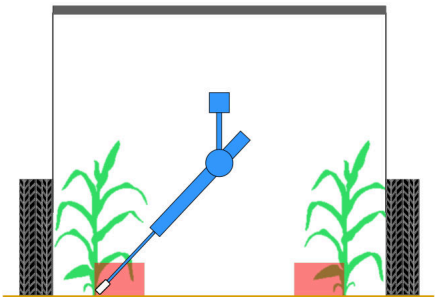
Figure 3.1: Overview of the different components of the weed control unit. The connections to the 24 V battery, which powers the motors, impulse generator, computing device (Nvidia Jetson), and LEDs, are highlighted in orange. The electrode at the tip of the arm is connected directly to the impulse generator. The three motors responsible for the robot arm movement are indicated in blue. The motors on the revolute joints communicate with the Jetson over a CAN bus, while the motor controlling the linear axis is connected via USB. Two more components are connected over USB to the Jetson: the RGB-D camera and the Arduino, which controls the LEDs. The RGB-D camera is powered through its USB connection.



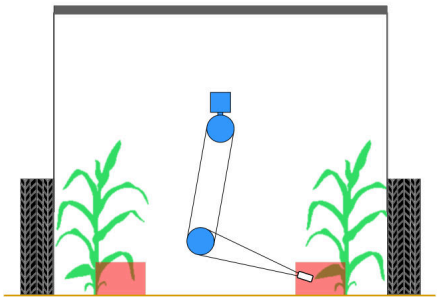
(a) XYZ gantry robot



(b) Delta robot



(c) Rotatable linear actuator



(d) Two-link arm

Figure 3.2: Design options for the robot arm. The red area indicates the target workspace. Designs (c) and (d) are mounted on a linear axis oriented in the drive direction. Designs (a) and (b) collide with the crop, while (c) and (d) avoid most of the plant leaves.

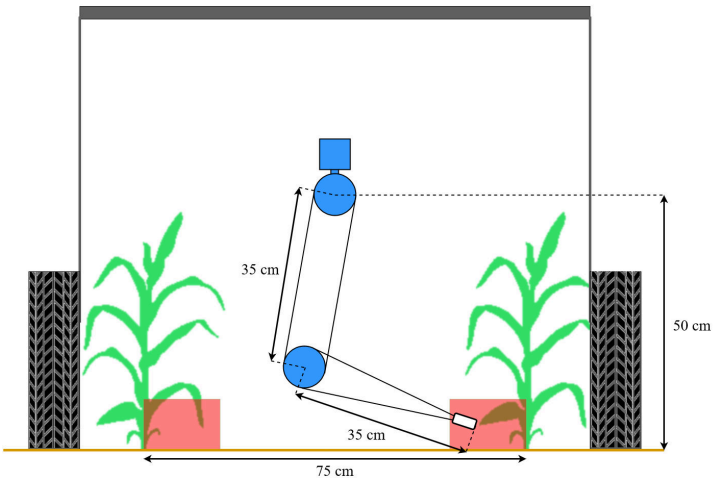


Figure 3.3: Final design of the robot arm with lengths marked.

joints. Both designs minimize crop interference by approaching the crop rows from the side.
After evaluating available hardware options, we opted for the design with the two revolute joints. The alternative design posed challenges, primarily due to the difficulty in finding a linear actuator that met all of our requirements. To cover the entire workspace, a stroke length of at least 150 mm was necessary. Additionally, the linear actuator needed to be both fast and accurate. While some electric cylinders satisfied these criteria, they were often heavy, expensive, and poorly suited to handling high radial loads.

3.2.2 Arm Dimensions

Determining the dimensions of the robot arm involved three key decisions.
The first decision concerned the length ratio between the two arm segments shown in Figure 3.3. Making the second arm longer than the first would introduce complexity when switching between crop rows and increase the likelihood of interference with plants. By ensuring the second arm is at least as long as the first, the lower arm can pass smoothly beneath the linear axis when switching sides. However, making the second arm significantly longer would reduce the clearance between the lower rotary joint and the ground, increasing the risk of collisions. For these reasons, we decided to make both arm segments equal in length.
The second consideration was the height of the upper revolute joint (mounted to the linear axis) relative to the ground. The objective was to minimize the overall arm length while ensuring it could still reach all points within the workspace. A shorter arm requires less torque to move the end effector, making the system more efficient. Ideally, the arm should be mounted as close to the ground as possible to reduce its length. However, as mentioned before, maintaining sufficient clearance between the lower joint and the ground was also critical to avoid collisions.
After careful evaluation, we selected a height of 50 cm between the upper revolute joint and the ground. To cover the entire workspace, the arm needs to extend to at least $\sqrt{50^2 + 37.5^2}\,\text{cm} = 62.5\,\text{cm}$. To account for uneven terrain and potential misalignment between the vehicle and crop rows we extended the arm segments to 35 cm each, providing a total reach of 70 cm.

The final design consideration was the length of the linear axis. While a longer axis reduces the number of stops the vehicle carrying the weed control unit needs to make, it also increases the overall size of the system. To keep the prototype compact, we opted for a linear axis with a working stroke of 1 m.

3.2.3 Revolute Joint Motors

To choose suitable actuators for the two revolute joints of the robot arm, we first calculated the torque requirements of each motor.
Ideally, both actuators would be mounted at the top, just beneath the linear axis, with the lower joint driven by either a roller chain or a toothed belt. This configuration would reduce the torque on the first joint. However, to simplify the mechanical design for the initial prototype, we decided to place the motor directly on the joint.
To estimate the required torque, we used the equations of motion for a two-link planar arm. The detailed derivation can be found in standard robotics textbooks [50]. For simplicity, we made several assumptions: the inertia of the motors was neglected, the linkages between the joints were considered massless, and we used a worst-case scenario where the arm is fully extended, parallel to the ground, and accelerating. The dynamic model for this configuration is expressed by the following equations:

$$\begin{aligned} \tau_1 &= \left(m_2(l_1+l_2)^2 + m_1 l_1^2\right)\alpha_1 + m_2 l_2 (l_1+l_2)\alpha_2 \\ &\quad + \left(m_2(l_1+l_2) + m_1 l_1\right) g \\ \tau_2 &= m_2 l_2 (l_1+l_2)\alpha_1 + m_2 l_2^2 \alpha_2 + m_2 l_2 g \end{aligned} \tag{3.1}$$

where τ_1 and τ_2 represent the torques at the two joints, l_1 and l_2 are the lengths of the arm segments, α_1 and α_2 are the accelerations at the actuators, m_1 is the mass of the motor on the second joint, and m_2 is the mass of the end-effector.
Based on the requirements outlined in subsection 1.2.3, we set $m_2 = 1\,\mathrm{kg}$, and estimated $m_1 = 1\,\mathrm{kg}$ for the motor mass. The lengths of the arm segments were given as $l_1 = l_2 = 0.35\,\mathrm{m}$. For acceleration, we assumed $\alpha_1 = 5\,{}^{\mathrm{rad}}/_{\mathrm{s}^2}$ and $\alpha_2 = 10\,{}^{\mathrm{rad}}/_{\mathrm{s}^2}$. We verify that these accelerations are sufficient to meet our performance requirements in section 5.4.
Substituting the values into Equation 3.1, we obtained the following peak torque requirements:

$$\begin{aligned} \tau_1 &= 15.81\,\mathrm{Nm} \\ \tau_2 &= 5.88\,\mathrm{Nm} \end{aligned}$$

Although these calculations included several simplifying assumptions, they offered an estimate of the maximum torque the system would encounter during operation. This estimate was then used to guide our selection of an appropriate motor.
Beyond the torque requirements, we considered other factors such as positional accuracy, maximum velocity, weight, size, communication interface, cost, and availability. After evaluating various options, we selected the CubeMars AK70-10 motor modules. These brushless DC motors come with an integrated 10:1 planetary gearbox and an onboard motor controller, offering a compact and efficient solution. The specifications for this actuator are provided in chapter 8.
Weighing only 521 g, the AK70-10 is well-suited for our application, minimizing the additional load on the arm. The motor provides a peak torque of 24.8 Nm, which comfortably exceeds our calculated torque requirements. During our tests, the motor's rated torque of 8.3 Nm was rarely exceeded, even for the upper joint,

indicating that the motor operates well within its optimal performance range under normal conditions.

In terms of positional accuracy, the motor has a backlash of 0.12 °. Actuators with harmonic drives can achieve much higher accuracy, often below 0.02 ° but they are significantly more expensive. Given that our application does not require sub-millimeter precision, the planetary gear solution provided by the AK70-10 was a more cost-effective choice.

Communication with the motors is facilitated through a CAN bus. This simplifies cable management by allowing both motors to share the same communication interface.

3.2.4 Linear Axis

For the choice of the linear axis, the primary requirement was its ability to handle the torques exerted by the attached two-link arm on the carriage. Additionally, it needed to be capable of achieving an acceleration of $1\,\mathrm{m/s^2}$ and maximum speeds of $0.5\,\mathrm{m/s}$. These specifications would allow the system to perform a point-to-point movement of 0.25 m within one second, which is sufficient to meet our operational requirements, as detailed in section 5.4.

When selecting a drive mechanism for the linear axis, there are two primary options: toothed belt and ball screw. Toothed belt axes have lower positional accuracy and load capacity compared to screw ball screw axes. However, they are cheaper, can achieve higher speeds and accelerations, and can cover longer travel distances. After evaluating the available hardware, we opted for a toothed belt axis due to its cost-effectiveness and its ability to meet our specific requirements. The datasheet for the selected Festo axis is included in chapter 8.

The chosen axis features a carriage that can withstand a maximum moment of 29.1 Nm on the roll axis, which is nearly double the expected torque on the upper revolute joint. Additionally, it offers a repetition accuracy of 0.1 mm, can accelerate up to $15\,\mathrm{m/s^2}$, and has a maximum speed of $1.2\,\mathrm{m/s}$, exceeding our initial requirements.

Next, we calculated the torque requirement for the motor driving the linear axis. The axis needs to move a total mass of approximately $m = 3\,\mathrm{kg}$ (including the end-effector, motors, and links). Given an acceleration $a = 1\,\mathrm{m/s^2}$ and a pulley radius of $r = 12.4\,\mathrm{mm}$, the required torque can be calculated using the equation:

$$\tau = m \cdot a \cdot r = 0.037\,\mathrm{Nm} \tag{3.2}$$

This calculation neglects the the friction of the linear axis, meaning the actual required torque will be slightly higher.

In selecting the motor for the linear axis, we also considered positional accuracy, maximum speed, size, cost, availability, and ease of integration. Ensuring compatibility with the chosen linear axis was also crucial. We selected an integrated servo motor from Teknic, whose detailed specifications are provided in chapter 8. It is a brushless DC motor with a rated torque of 0.7 Nm at 24 V which is more than sufficient for our application. The built-in encoder offers a resolution of 0.057 ° which translates to a linear resolution of 0.04 mm on the axis. This ensures accurate positioning during operation.

A significant advantage of this motor is the manufacturer-provided C++ library, which includes functions for positioning and homing. This simplified the software integration process. The motor can be connected to a computer via a USB communication hub.

Figure 3.4: CAD model of the robot arm. The parts highlighted in blue are custom-made metal sheet parts.

3.2.5 Linkages

In addition to the hardware decisions of the three motors and the linear axis, we also had to design links between the motors. For the arm segments, we chose off-the-shelf carbon fiber tubes because they are stiff, lightweight, and allow us to route motor cables through them.
To ensure robust connections between the carbon fiber tubes and the motors, as well as between the motors and the linear axis carriage, we designed custom metal sheet parts. Additionally, we selected an energy chain to manage the movement of cables along the linear axis. The energy chain is positioned on top of the linear axis and carries the CAN bus and power cables.
The CAD model of the final design is shown in Figure 3.4. Along the drive direction, the arm can travel 90 cm. While the linear axis has a working stroke of 100 cm, we had to limit the range slightly to prevent the carbon fiber tubes from colliding with the motor on the linear axis.

3.3 Single-board computer

For the main computing unit, we selected the reComputer Industrial J4011, which features an Nvidia Jetson Orin NX module with 8 GB of memory. The Jetson platform was chosen specifically to be able to utilize the latest generation graphics processing unit (GPU), which speeds up the inference of our weed classification network.
The reComputer Industrial is a fanless design, making it well-suited for our applica-

Figure 3.5: Front view of the Orbbec Femto Bolt. The camera is equipped with an RGB sensor, a depth camera, and an IR emitter.

tion, as it reduces the risk of dirt and debris from the field infiltrating the computer. The computer is mounted directly at the end of the linear axis, next to the Teknic motor, to simplify cable management.
The computer is responsible for handling the motion control of the robot arm and is connected directly to all three motors. It has an onboard CAN transceiver that is connected to the CAN bus of the CubeMars motors. The Teknic motor is connected to the Jetson through the USB communication hub.
Additionally, the computer is connected to a 4G mobile travel router via Ethernet. This setup enables easy remote access to the system from any device by connecting to the router's WLAN and using SSH or Remote Desktop Protocol (RDP). Although the software of the weed control unit operates completely offline, the internet connection is useful for tasks such as updating the operating system.

3.4 RGB-D camera

To accurately detect weeds, we require a camera that captures images of the ground and provides the 3D location of detected weeds. For this purpose, we utilize an RGB-D camera, which provides depth estimation for each pixel in the image.
Consumer RGB-D cameras employ various technologies such as stereoscopy, structured light, and time-of-flight (ToF). According to Neupane et al. [51], who measured the root mean square error of different depth cameras in both indoor and outdoor settings, ToF cameras exhibit higher accuracy than stereo cameras. Based on their findings, they recommend the use of the Azure Kinect camera. Although the Azure Kinect was discontinued in 2023, its camera module is still available as part of the Orbbec Femto Bolt camera. Consequently, we selected the Orbbec Femto Bolt, shown in Figure 3.5, as our RGB-D camera.
The Femto Bolt is equipped with a color camera offering a 4K resolution, alongside a depth sensor with a 1 megapixel resolution. In our setup, we utilize the unbinned wide field of view depth mode, which provides a depth range from 0.25 m to 2.21 m, aligning well with our operational requirements.
To have an unobstructed view of the ground, we mount the camera beneath the linear axis. The color camera has a field of view of $80\,^\circ \times 51\,^\circ$. At a distance of 50 cm from the ground, this covers an area of 84 cm × 48 cm. Given the dimensions of the robot arm's workspace, a single image is insufficient to cover the entire area. Therefore, we capture two images: one at the front and another at the back of the

workspace. To accomplish this, the camera is mounted on the carriage, as shown in Figure 3.4. The two images are captured with some overlap, resulting in a total photographed area of 84 cm × 92 cm. This is slightly larger than the robot's arm operational workspace, which is constrained to 75 cm × 90 cm. The weed detection algorithm is designed to filter out any weeds that fall outside of the arm's reachable area.
We also considered mounting the camera closer to the ground to capture a side view of a crop row. This configuration could potentially detect weeds growing below the crop leaves, which are often occluded when viewed from above. However, the dataset we used for training our classifier (see subsection 4.3.4) only contained images taken from an overhead perspective. Therefore, we opted for the same setup. Additionally, mounting the camera at the top offers the benefit of being further away from the electric field generated by the electrode, reducing the risk of electromagnetic interference. It also helps keep the camera cleaner by minimizing exposure to dirt and debris.
The Femto Bolt is connected to the Nvidia Jetson via USB 3.2, which provides both power and data transmission.

3.5 Mechanical Structure

The robot arm is mounted to a stainless steel frame, which is securely fastened to the vehicle. In the future, this vehicle will navigate autonomously along crop rows. For this thesis, we utilized a dummy vehicle that is moved around manually for testing and development. An image of the fully assembled hardware is shown in Figure 3.6.

3.6 Illumination

For our weed detection algorithm, consistent illumination is crucial. To achieve this, we shield the entire workspace from direct sunlight using plywood and tarpaulins, as shown in Figure 3.1. To provide controlled and uniform lighting across the workspace, we use WS2811 LED strips attached below the wooden board. These LEDs are distributed evenly and provide controlled and uniform lighting for the entire workspace.
The LED strip is controlled by an Arduino microcontroller, which is connected to the main computer via USB and communicates through serial communication.

3.7 Power Supply

To power the entire system, we use two 12 V lead acid batteries connected in series, providing a total output of 24 V. All components can operate directly from the 24 V supply or from a USB port of the computer. We do not need any DC-to-DC converters.
The motors are capable of running at 48 V, which would increase their maximum speed and power output. However, this was deemed unnecessary for our application, as the current configuration adequately meets the operational requirements.

Figure 3.6: View of the weed control unit from below. The linear axis is attached to a metal frame that is covered with a plywood panel. Attached to the wooden board are LED strips that illuminate the ground. The camera and the two-link arm are attached to the carriage of the linear axis. The arm consists of two motors connected by carbon fiber tubes and an electrode mounted at the tip. The lower arm can rotate below the linear axis without interfering with the camera. The third motor is visible at the end of the linear axis.

Figure 3.7: Tarpaulins shield the workspace from external light, ensuring consistent illumination for weed detection. The batteries that power the system are attached at the front, while the high-voltage impulse generator is located at the back.

Chapter 4

Software Implementation

4.1 System Overview

The software for the weed control unit is executed on the Nvidia Jetson Orin NX board, which is responsible for both the motion control of the robot arm and the weed detection. An overview of the software architecture running on the Jetson board is illustrated in Figure 4.1 and will be discussed in detail in the following sections.
The Jetson board was set up using Nvidia's JetPack, a software development kit that includes drivers for the on-board GPU as well as software libraries such as CUDA and TensorRT, which facilitate GPU-accelerated computing. For this thesis, JetPack 6.0 was used, the latest version available at the time of writing, which comes with an Ubuntu 22.04-based root file system.

4.1.1 Robot Operating System

The Robot Operating System (ROS) is an open-source middleware framework widely adopted in the robotics industry for its flexibility and modularity. It simplifies the development, deployment, and operation of complex robotic systems by offering a collection of tools, libraries, and conventions aimed at simplifying the task of creating robust robot behavior across a wide variety of robotic platforms.
We chose to implement the software using ROS 2, the second generation of ROS [52]. ROS 2 offers several advantages over its predecessor, including enhanced real-time capabilities, improved communication protocols, and better support for multi-robot systems. This choice not only made our codebase more modular and reusable but also aligned with the objectives of the Open Field Automation initiative to make the system accessible and adaptable for other researchers and developers. Additionally, ROS 2 allowed us to leverage existing frameworks such as ros2_control for hardware abstraction and MoveIt 2 for advanced motion planning, significantly accelerating the development process.
For this project, we installed ROS 2 Iron, the latest compatible version available for Ubuntu 22.04.

4.2 Robot Arm Motion Control

There are two primary approaches to controlling the position of a robot arm: decentralized control and centralized control [50].
The decentralized approach treats each joint as an independent single-input/single-output (SISO) system. In this method, the coupling effects between the joints are

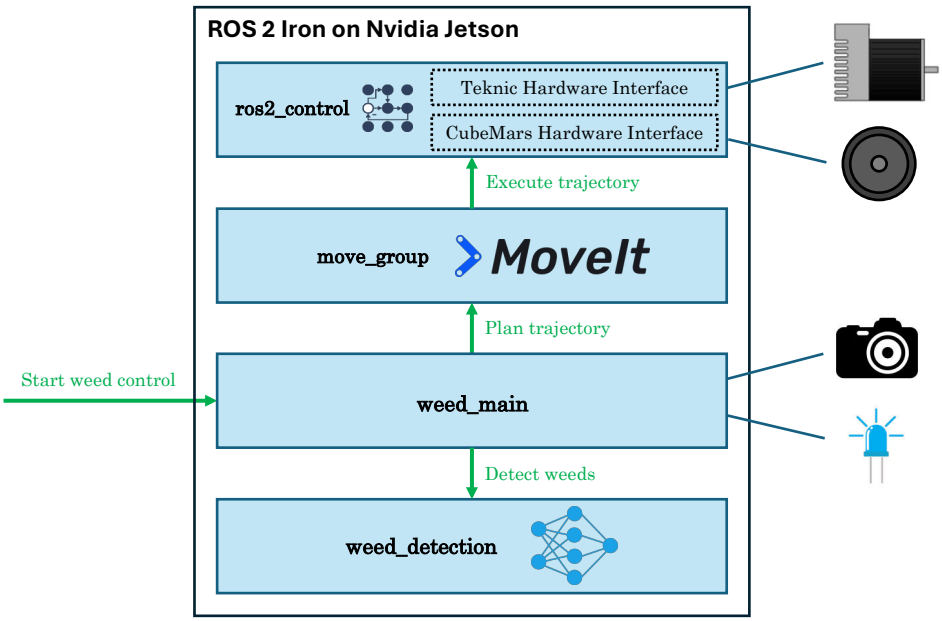


Figure 4.1: Software architecture overview of the weed control unit running ROS 2 Iron on the Nvidia Jetson Orin NX board. The ROS nodes are depicted as blue blocks, while the green arrows indicate ROS actions. The `weed_main` node interacts with the depth camera and LED strips, and the `ros2_control` node facilitates communication with the robot arm's motors.

considered disturbances and are not explicitly accounted for in the control design. Each joint has its own feedback controller, typically consisting of a proportional (P) position controller as the outer loop and a proportional-integral (PI) velocity controller as the inner loop.
While this method may not provide the best performance due to the lack of compensation for inter-joint coupling, it is widely used in industrial robot applications due to its simplicity and robustness [53].
In contrast, the centralized approach accounts for the nonlinear coupling between the joints. This method involves a central controller that calculates the required torques for each joint based on a dynamic model of the entire robot arm. By considering the full system dynamics, including the coupling effects, centralized control can significantly improve the robot's performance [50]. However, this approach requires an accurate physical model of the robot to solve the inverse dynamics problem effectively.
For this project, we opted for the simpler decentralized control method. The primary reason for this choice was that our application does not require high operational speeds, allowing us to prioritize simplicity and reliability over optimal performance. Additionally, the CubeMars and Teknic motors used in the robot arm have integrated position controllers. This integration makes decentralized control advantageous, as it reduces the reliance on continuous communication with the main computer. The control loop is closed directly on the motors, making the system more resilient to potential communication delays or interruptions.
We manually tuned the velocity and position loops of all motors using standard heuristic methods [54], performing the tuning with the full load connected to ensure realistic conditions. High proportional gains were employed to make the arm's movements as responsive as possible.

4.2.1 ros2_control

ros2_control is a key framework within the ROS ecosystem, providing a standardized interface for communicating with hardware components such as motors and sensors. It abstracts the hardware layer, allowing seamless integration of new components while keeping the hardware communication isolated from the rest of the software implementation. This modularity simplifies the process of adding or replacing hardware without the need for significant changes to the overall system.
To integrate the Teknic and CubeMars motors into ros2_control, we developed two custom hardware interfaces [55]. A hardware interface in ros2_control is a concrete implementation of an abstract hardware class defined by the framework. This implementation is compiled as a library and loaded as a plugin by the ros2_control controller manager, enabling communication with the specific hardware.
The primary functions we implemented in the hardware interface include:

- `on_configure` and `on_cleanup`: Initialize and terminate communication with the motors.
- `read`: Fetches the current position, velocity, and torque values from the motors.
- `write`: Sends position, velocity, or torque commands to the motors, depending on the selected control mode.
- `perform_command_switch`: Switches the motor's command mode, such as transitioning from position control to velocity control.

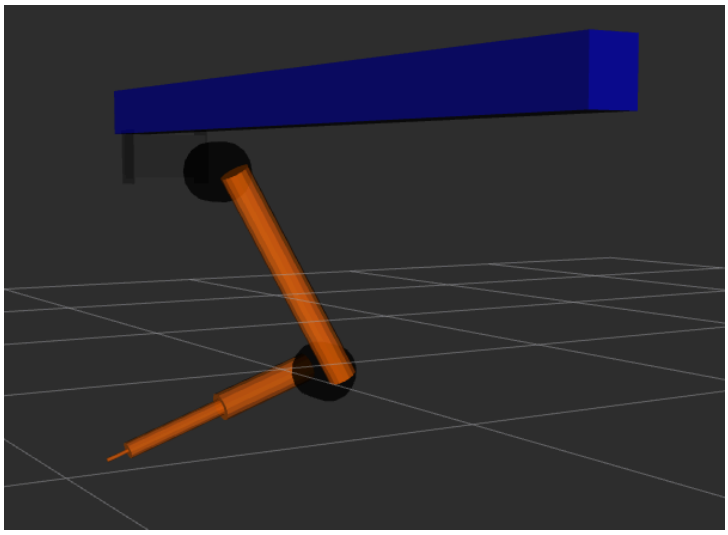


Figure 4.2: Visualization of the robot's Unified Robot Description Format (URDF) model in RViz.

For the CubeMars motors, which communicate over a CAN bus, we utilized the SocketCAN library to handle CAN communication. The message formats for sending commands to and receiving feedback from the motors were derived from the CubeMars manual, ensuring compliance with the manufacturer's communication protocol.
In addition, we implemented a safety feature where the motors disengage if the measured torque exceeds a predefined limit. This precaution prevents potential damage to the robot arm or motors in case of unintended collisions, such as accidentally crashing into the ground.
The Teknic motors were integrated using the manufacturer-provided C++ library. Since the library already implements the core communication functions, we were able to directly call these within our hardware interface. Although creating a hardware interface for a motor with an existing library might seem redundant, it offers the significant advantage of hardware abstraction through a standardized interface. This ensures consistency across different hardware components and simplifies future upgrades or replacements.
The `ros2_control` node, depicted as the top node in Figure 4.1, is the only part of our codebase that directly interacts with the robot arm's actuators. Its primary control loop runs at a frequency of 100 Hz, sending position commands to the motors approximately every 10 ms to the motors. This frequency, in conjunction with the motors' integrated position controllers, enables smooth and responsive movement of the robot arm. During testing, we found that lowering the control loop frequency resulted in choppy motion while increasing it unnecessarily added computational load without perceptible improvements in smoothness.

4.2.2 MoveIt 2

One of the key advantages of using ros2_control is its seamless integration with other frameworks, such as MoveIt 2. MoveIt 2 is a comprehensive platform for robotic manipulation, providing developers with the tools to implement complex tasks. It includes features for forward and inverse kinematics, motion planning algorithms, and real-time collision detection and avoidance capabilities.
To utilize MoveIt 2, the kinematic structure of the robot must be defined using

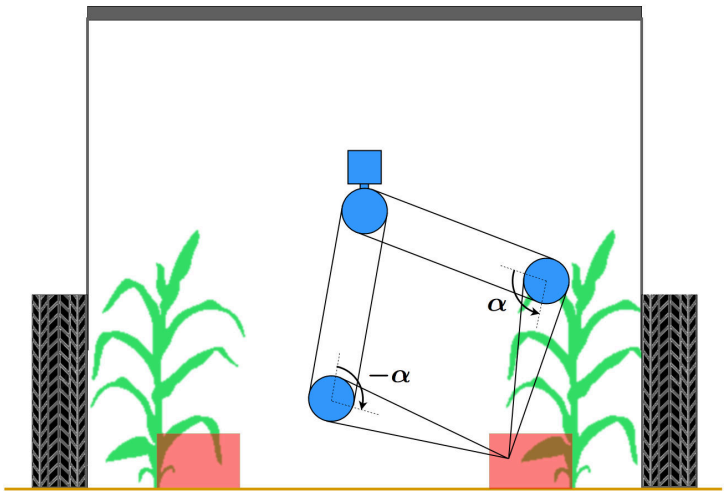


Figure 4.3: Illustration of two possible configurations for the robot arm to reach the same end-effector position. In this example, the program would choose the solution with a negative angle at the second joint. The alternative configuration, which would result in interference with the plant, is not used. For positions on the opposite side of the workspace, the program selects the configuration with a positive angle on the second joint.

the Unified Robot Description Format (URDF). The URDF file serves as a digital representation of the robot's physical structure, including the links and joints. We created this file based on the CAD model of our robot arm, using simple geometric shapes like rectangles and cylinders to represent each component. This simplification facilitates collision detection, as simpler shapes reduce computational complexity. Additionally, we ensured that the collision boxes in the URDF were the same size or slightly larger than the actual hardware to provide a safety margin. Figure 4.2 shows a visual representation of the robot as defined by the URDF.

MoveIt 2 uses a numerical solver for inverse kinematics by default. However, for robots with simpler kinematics, such as ours, it is possible to use IKFast, an analytical kinematics solver. IKFast is part of the OpenRAVE motion planning software [56] and can automatically generate C++ code for the robot's analytic solutions. This results in faster and more stable inverse kinematics calculations.

The IKFast solver will usually find two solutions for our robot arm, as illustrated in Figure 4.3.To ensure minimal interference with the crop's plant leaves, we customized the C++ code generated by IKFast to select the most suitable configuration. Specifically, for points to the right of the robot's linear axis, we choose the solution with a positive angle at the second joint. Conversely, for points on the left side, we select the solution with a negative angle at the second joint. We defined the zero position of the second joint as the lower arm pointing toward the first joint, rather than the fully extended position.

For motion planning, MoveIt 2 uses the RRT-Connect algorithm [57] by default. Once the motion planner generates a path, it is converted into a trajectory using Time Optimal Trajectory Generation (TOTG) [58]. The trajectory generation is based on predefined velocity and acceleration limits. The specific limits used for our robot are described in section 5.4.

To interact with MoveIt 2, we utilize the `move_group` node, as shown in Figure 4.1.

When motion planning to a target position is required, a ROS action is sent to this node. After planning the motion, the `move_group` node sends the generated joint trajectories to the `ros2_control` node. The `ros2_control` node then utilizes the hardware interfaces to send the corresponding position commands to the motors, executing the planned movement.

4.3 Weed Detection

The weed detection algorithm implemented in this project comprises three main steps: segmenting plants from the ground, identifying connected components, and classifying individual plants.
Unlike the increasingly popular end-to-end learning models [59, 45, 60], our approach employs a deep neural network solely for plant classification after segmentation. The primary reason for choosing this method over an end-to-end approach was the lack of a dataset tailored to our specific camera setup and field conditions. End-to-end models tend to suffer from performance degradation when applied to environments that differ from their training data [49]. By incorporating a pre-segmentation step, we aimed to mitigate this issue and enhance the model's robustness under varying field conditions.
Additionally, using a classification network instead of an object detection or semantic segmentation network allowed us to leverage larger, more diverse datasets such as PlantCLEF Goëau et al. [61]. By training on a more extensive dataset, we hoped to develop a model that could generalize better and handle the complexities of our specific agricultural environment more effectively.
The detailed implementation of our weed detection algorithm is outlined in the following sections. While the majority of the processing is conducted within the `weed_detection` node (see Figure 4.1), certain aspects, such as camera communication and LED control, are managed by the `weed_main` node.

4.3.1 Image Capture

To set up the camera and capture images, we used the Azure Kinect Software Development Kit, which is compatible with the Orbbec Femto Bolt camera. The camera provides three types of images for each frame: a color image, a depth image, and an infrared image. An example of these three images is shown in Figure 4.4.
Due to the spatial separation of the sensors on the camera, the depth and infrared images have a different viewpoint than the color camera (see Figure 3.5). To align these images with the color camera frame and ensure pixel correspondence across all images, a transformation is necessary. The Azure Kinect SDK offers built-in functions to perform this transformation. While this process can sometimes lead to occlusions, we found that this was not an issue in our setup, as the scene in our application is mostly flat.

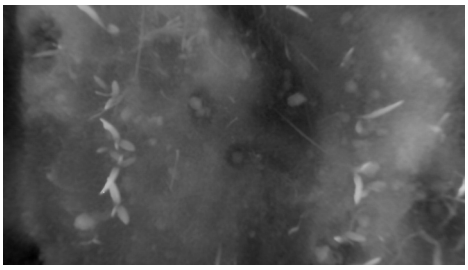
4.3.2 Plant Segmentation

The first step of the weed detection algorithm involves segmenting the plants from the ground. This process creates a binary image where each pixel is classified as either background or plant.
We utilized the popular Excess Green Index (ExG) for segmentation, which is calculated as:

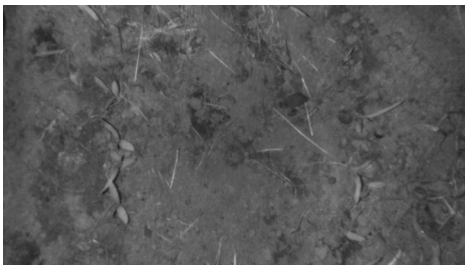
$$ExG = 2G - R - B \tag{4.1}$$

where G, R, and B represent the pixel intensities of the green, red, and blue channels, respectively [62]. In the resulting image (see Figure 4.5b), plant pixels appear

(a) Color image



(b) Depth image



(c) Infrared image

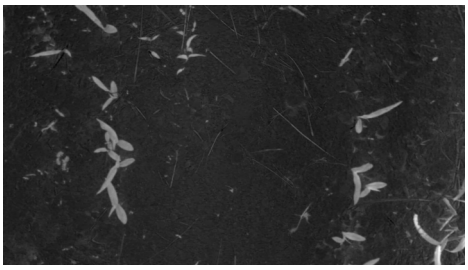
Figure 4.4: The three images captured by the RGB-D camera. The depth and infrared images have been transformed to align with the viewpoint of the color camera, ensuring pixel correspondence across all three images.

significantly brighter than the background due to their disproportionately high green values. By applying a simple threshold, $ExG > 0$, we were able to generate a segmentation mask with minimal noise, as shown in Figure 4.5c.
We experimented with other color-based vegetation indices such as the Excess Red Index (ExR) and Excess Green minus Excess Red (ExGR). However, in our setup, the ExG index provided the best contrast between plants and soil.
The ToF camera provides an image in the near-infrared (NIR) spectrum, which allows us to calculate the Normalized Difference Vegetation Index (NDVI). This index is defined as:

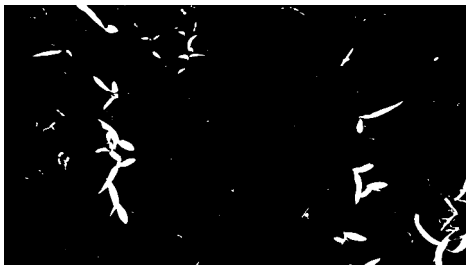
$$NDVI = \frac{NIR - R}{NIR + R} \tag{4.2}$$

where NIR represents the near-infrared channel and R the red color channel. NDVI is based on the fact that healthy vegetation typically has high reflectivity in the NIR spectrum and low reflectance in the visible spectrum [63]. Figure 4.6c and Figure 4.6d show the resulting NDVI image and the binary mask after thresholding..
Despite the success reported by other studies in using NDVI for plant-soil segmentation [32], our results were suboptimal regardless of the thresholding method used. This discrepancy could be due to differences in the sensor or variations in lighting and field conditions. We observed that the red channel (see Figure 4.6b) not only had lower values for plant pixels but also for certain soil patches where the soil was darker. This resulted in parts of the ground appearing bright in the NDVI image, making it challenging to threshold the image without introducing significant noise.
We also experimented with thresholding the NIR image directly, without computing the NDVI first. However, as shown in Figure 4.6e and Figure 4.6f, the contrast between plants and soil was generally low. Some dead plants and parts of the soil reflected similar amounts of infrared light as healthy plants, making it difficult to achieve a clean binary segmentation.
Our initial plan was to use a combination of ExG and NDVI for segmentation,

(a) Color image



(b) ExG image



(c) Thresholded ExG

Figure 4.5: Plant segmentation with ExG. For illustration purposes (a) was normalized between the values 0 and 255.

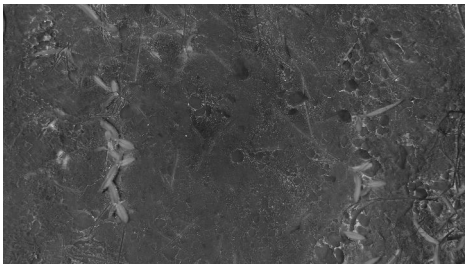
similar to the approach by Lottes et al. [32]. However, in our experiments, incorporating the NDVI image only introduced additional noise without improving the segmentation results achieved with ExG alone. Consequently, we decided to rely solely on the ExG index for plant segmentation.
One of the main drawbacks of using a threshold-based segmentation method is its sensitivity to lighting conditions. To mitigate this, we ensured consistent lighting during image capture. As described in section 3.6, the workspace was shielded from direct sunlight and illuminated using LED strips. Additionally, we disabled the camera's auto white balance and auto exposure features to maintain consistent camera parameters, further enhancing the reliability of the segmentation process.

4.3.3 Finding Connected Components

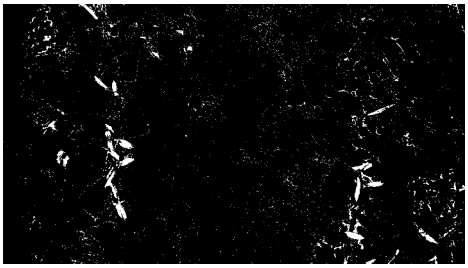
After obtaining a binary vegetation mask, the next step is to identify and separate the connected components within the image [38]. Since the vegetation mask may contain noise and misclassified pixels, we use morphological operations to clean up the segmentation [28].
To begin, we apply a morphological opening operation to the binary mask. This process helps remove small noise elements by eroding the edges of regions and then dilating them back, effectively smoothing out the segmented areas. Following this, we use connected-component labeling to identify clusters of neighboring pixels, treating each cluster as a distinct blob. Regions with an area below a certain threshold are considered noise and filtered out. The resulting connected regions are shown in Figure 4.7b.
A common issue we observed was that the leaves of a single plant were often split into multiple components. To address this, we perform an additional dilation operation on the vegetation mask before executing the connected-component analysis. This dilation increases the size of the plant regions, merging fragmented leaves into a single component. The dilated vegetation mask is illustrated in Figure 4.7c, and

(a) Color image

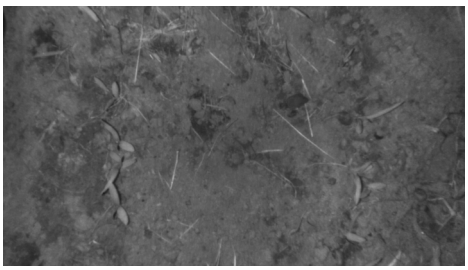
(b) Red channel



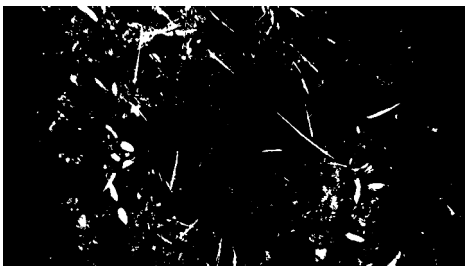
(c) NDVI image



(d) Thresholded NDVI

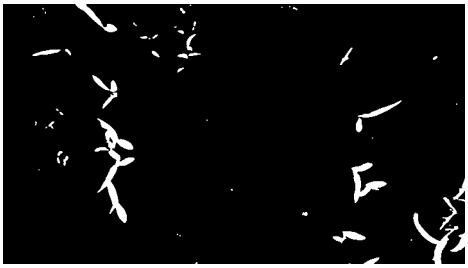


(e) Infrared image

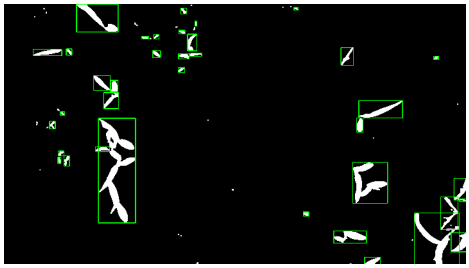


(f) Thresholded infrared

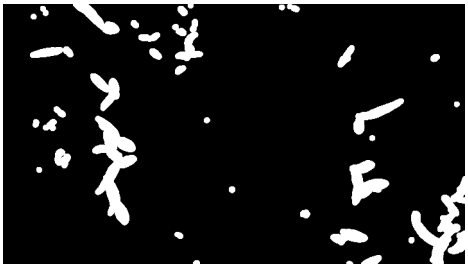
Figure 4.6: Plant segmentation using the infrared channel. For illustration purposes (c) was normalized between the values 0 and 255.



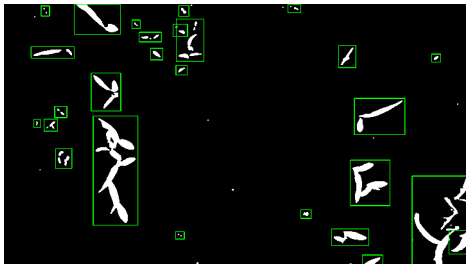
(a) Vegetation mask



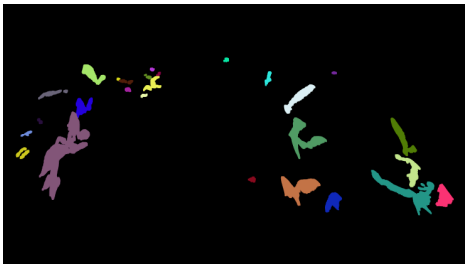
(b) Connected components without dilation



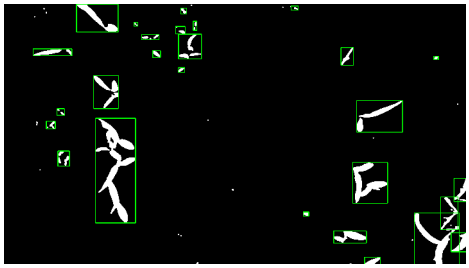
(c) Dilated vegetation mask



(d) Connected components with dilation



(e) Clustered 3D point cloud of vegetation mask



(f) Connected components with 3D clustering

Figure 4.7: Comparison of three methods for finding connected components in the vegetation mask. (b) shows the result of 2D connected-component analysis without dilation (d) applies 2D connected-component analysis after performing a dilation operation, and (e) demonstrates 3D clustering using DBSCAN. The point cloud in (c) is displayed from an angled perspective, different dataset it to appear different from the other images.

the connected regions after applying dilation are shown in Figure 4.7d.
While this dilation helps group leaves of the same plant together, it can also cause adjacent plants that were previously separated to become connected. Therefore, it is challenging to achieve perfect segmentation where each plant is correctly identified as a single region. This limitation is inherent in pre-segmentation methods, as opposed to learning-based approaches Milioto et al. [40].
Given that our depth camera provides an estimation of the 3D location for each point, we explored an alternative approach for identifying connected components using 3D clustering. This approach consists of the following four steps:

- **Projection to 3D**: We project all points from the vegetation mask into 3D space, forming a point cloud representing the detected vegetation.
- **Downsampling**: To improve computational efficiency, the point cloud is downsampled using voxel downsampling. This reduces the number of points while preserving the overall structure of the vegetation.
- textbfClustering: We apply the Density-based spatial clustering of applications with noise (DBSCAN) algorithm to the downsampled point cloud. DBSCAN forms clusters based on the density of neighboring points, similar to how connected-component analysis works in 2D. The resulting clusters are displayed in Figure 4.7e.
- **Back-Projection**: Finally, we project each cluster back onto the image plane and draw bounding boxes around the identified components, as shown in Figure 4.7f.

The use of DBSCAN is advantageous because it allows for clustering based on the spatial distance between points, which is analogous to connected-component analysis in 2D. By downsampling the point cloud before applying DBSCAN, we significantly speed up the computation without losing critical information. However, this method is still slower than traditional 2D connected-component analysis.
The primary advantage of performing clustering in 3D space is that it can differentiate plants that overlap in the 2D image but are separated along the third dimension. As a result, plants that appear connected in 2D but are spatially distinct in 3D will be correctly identified as separate components. Comparing Figure 4.7b, Figure 4.7d, and Figure 4.7f we can see that the 3D clustering approach tends to provide better segmentation results. In Figure 4.7f, both plants in the bottom right corner are separated, while the leaves of the plants in the upper left corner are successfully connected. Still, it is not a perfect solution and cannot separate plants whose leaves physically touch each other.

4.3.4 Plant Classification

After identifying the bounding boxes of the plants, the algorithm crops these regions from the original color image, scales them to match the input dimensions required by the network, and passes them through the classification network. The details of the network training process are described in section 4.4. An example of the classified plants is shown in Figure 4.8.
The final step in the weed detection algorithm is to determine the exact 3D positions that the robot arm should move to. Our objective is to target the stem positions of the weeds, as outlined in subsection 1.2.3. Some researchers have employed learning-based methods to detect stem positions directly [60, 64, 45]. However, implementing an additional neural network for this purpose was beyond the scope of this work. A simpler approach is to assume that the stem is located near the plant's centroid Xiong et al. [65], Visentin et al. [13].

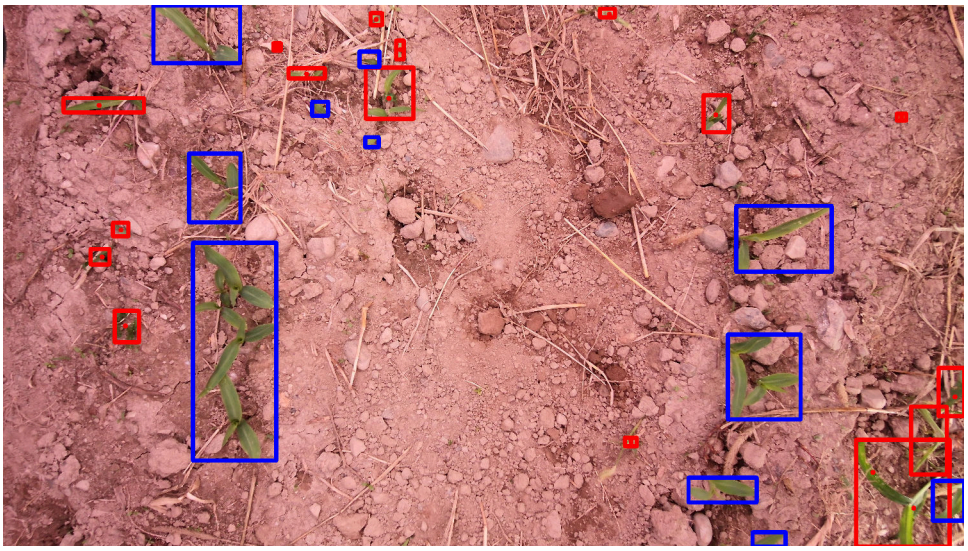


Figure 4.8: Output of the weed classification process. Plants classified as crops are highlighted in blue, while those identified as weeds are marked in red. The target positions for the weeds, where the robot arm will apply treatment, are indicated by red points.

From our previous calculations in subsection 4.3.3, we already have the 3D point cloud for each plant. Using this data, we calculate the 3D centroid of each plant to approximate the stem position. For plants exceeding a certain size threshold, we apply an additional k-means clustering, where the number of clusters k is determined based on the plant's size. This approach offers two advantages:

- **Multiple Treatments for Large Weeds**: Large weeds receive multiple electric impulses, which is necessary to ensure they are effectively killed.
- **Handling Over-Segmentation**: Plants that were mistakenly connected during pre-segmentation are likely to receive multiple treatment positions due to their size, effectively allowing us to treat them as separate entities.

It is important to note that the calculation of centroids and the application of k-means clustering are performed only for plants that have been classified as weeds. The resulting treatment positions are marked in Figure 4.8, where one weed located in the bottom right has been assigned two treatment positions due to its size.

4.4 Network Training

4.4.1 Dataset

For training the network, we utilized two datasets: the PlantCLEF 2024 dataset [66] and the CropAndWeed (CAW) dataset [67].
PlantCLEF is an annual competition aimed at developing state-of-the-art plant classification models. Each year, extensive datasets for plant classification are released. The 2024 competition dataset comprises 1.4 million images of 7.8k plant species, with a focus on the flora of southwestern Europe. The images are categorized into types such as flower, leaf, fruit, bark, and habit.
Despite its large size, PlantCLEF contains many images that are not directly relevant to our task of classifying weeds in agricultural fields. It also has fewer images

of plants in their early growth stages. However, accurately classifying seedlings is crucial for our application, as crops are most vulnerable at the early growth stage. To address this limitation, we also trained our network on the CropAndWeed (CAW) dataset, which is specifically designed for identifying weeds in agricultural fields. This dataset is one of the most extensive and diverse collections for weed detection [48], covering several hundred cultivation areas in Austria. It contains 112k instances of 16 crops, including maize and sugar beet, along with 58 weed classes. The dataset was annotated for object detection and image segmentation tasks. To adapt it for image classification training, we extracted the annotated bounding boxes from the images.
While the PlantCLEF dataset is fairly balanced across classes, the CAW dataset is imbalanced, with some classes, like corn and sugar beet, having significantly more instances than others. However, this imbalance reflects the expected distribution of plants in our target fields, making the dataset's inherent bias advantageous for our application.
To identify the optimal dataset combination for training, we experimented with the following configurations:

1. Training on the CAW dataset alone.
2. Training on the PlantCLEF dataset alone.
3. Training on both the CAW and PlantCLEF datasets.
4. Training on the CAW dataset and a subset of the PlantCLEF dataset, excluding all image types except leaf.

The last configuration was included because we hypothesized that images of flowers or fruits would not contribute to weed detection performance, given that crops and weeds are primarily in the leaf stage during early growth.
Additionally, since our goal was to distinguish between crops and weeds, we explored whether it was necessary to use the full set of class labels during training. We tested two different class configurations:

1. Using all 7.8k classes available in the datasets.
2. Using only two classes: crop and weed

The second option was tried for both sugar beet and corn, the crops of primary interest in this thesis.

4.4.2 Architecture and Training Process

While Convolutional Neural Networks (CNNs) have long been the standard for image classification tasks, recent advances in Vision Transformers (ViTs) [68], pre-trained using self-supervised learning, have demonstrated superior performance on a variety of tasks, including plant classification [69]. These foundation models can achieve high performance on tasks such as classification or segmentation with minimal fine-tuning.
We opted to use a vision transformer pre-trained with the DINOv2 method [70], a state-of-the-art foundation model. For training, we employed MMPreTrain, a toolbox designed primarily for image classification tasks [71], built on the PyTorch framework. MMPreTrain uses configuration files to define all hyperparameters and training settings, allowing us to train our network without writing additional code, simply by creating a config file. This approach enhances reproducibility and makes the training process more transparent. An example configuration file is provided in Appendix A. The same hyperparameters, including learning rate, optimizer, and

scheduler, were used across all training configurations, with only the dataset and classification head varying.
For evaluating the different datasets and class combinations, we fine-tuned only the classification head of the vision transformer to reduce training time. Each model was trained for 12 epochs before evaluation. The final model was fine-tuned for an additional five epochs using a smaller learning rate. Most of our training was conducted on one or two A100 GPUs.

4.4.3 Evaluation Metrics

To evaluate the model's performance, we created a test set consisting solely of images from the CAW dataset. We excluded images from the PlantCLEF dataset in the test set because the CAW dataset more accurately represents our target domain.
For assessing the different dataset combinations, we used top-1 accuracy as the primary metric. For the comparison between binary class labels and the full set of class labels, we employed the F2-score. We chose not to use accuracy for binary classification due to the high class imbalance, with the weed class encompassing a far greater number of images than the crop class. While the F1-score is a common metric, we opted for the F2-score because it places more emphasis on recall, which aligns with our goal of minimizing false negatives. A false negative in this context means that a crop is mistakenly killed, directly affecting yield, whereas a false positive means a weed is left untreated. While the latter can still impact crop health, it is less critical and can be rectified in subsequent treatments at different growth stages.
The results of our evaluation are presented in section 5.1.

4.4.4 Deployment

For deployment, we exported the final PyTorch model to the ONNX format and then converted it to a TensorRT engine on the Nvidia Jetson. This allowed us to achieve the highest possible inference speed. Processing a single input takes the network on average 110 ms.

4.5 System Integration

The `weed_main` node serves as the central integrator for all the software components, managing the interaction between the `weed_detection` node and the `move_group` node. This node effectively bridges the weed detection process with the motion control of the robot arm, ensuring seamless operation of the entire weed control unit.
The workflow of the software is structured as follows:

1. **Wait for Action**
   The `weed_main` node awaits a ROS action to initiate the main program. This is a custom-defined action with an empty request and result message. Once the request is received, the workspace treatment process begins.

2. **Capture Camera Images**
   The linear axis moves the camera to two predetermined positions, where images of the ground are captured. During this process, the robot arm is positioned to the side to avoid obstructing the camera's view.

3. **Weed Detection**
   The captured images are processed by the weed detection algorithm, which returns a list of 3D coordinates for the detected weeds.

4. **Filter and Sort Positions**
   The list of detected weed positions is filtered to remove any weeds outside the robot arm's workspace. The remaining positions are then sorted so that all weeds in the left crop row are treated first, followed by those in the right crop row. This sorting minimizes the need for the robot arm to change orientation multiple times, enhancing efficiency.

5. **Move to Positions**
   The sorted positions are sequentially passed to the `move_group` node, which plans and executes the arm's trajectory to each weed location. At each position, the high-voltage impulse generator is activated to shock the weed. To avoid collisions with the ground, the end effector is lifted after each treatment before moving to the next position.

6. **Report Success**
   Once the entire workspace has been treated, the ROS action goal is marked as successful.

The design of this system allows for easy integration with other ROS-based systems. Since ROS actions can be sent between different computers connected to the same network, the weed control unit can seamlessly interact with the autonomous vehicle navigating along the crop rows. The vehicle, also running ROS 2, can send a ROS action to the `weed_main` node to start the weed treatment process, wait for it to complete, and then move to the next location.
At the time of this thesis, the autonomous vehicle was not yet available for integration. Therefore, for our field tests, we initiated the start ROS action via the command line.

Chapter 5

Results

5.1 Performance of Weed Classification

Table 5.1: Accuracy of plant classification with various training dataset combinations.

Training Datasets	Accuracy
Entire PlantCLEF & CAW	74.69 %
Leaf PlantCLEF & CAW	**75.41 %**
CAW	72.47 %
Entire PlantCLEF	9.94 %

The performance of the plant classification network using different training datasets is summarized in Table 5.1. It is important to note that the test dataset consisted solely of images from the CAW dataset, which represents our target domain.
Training with the PlantCLEF dataset alone resulted in poor performance. However, combining PlantCLEF with CAW improved the classification accuracy. The highest accuracy was obtained by restricting the PlantCLEF dataset to leaf images and combining it with the CAW dataset. This combination was used for further evaluations.
Table 5.2 and Table 5.3 show a comparison between training with all 7.8k plant species as class labels and binary labeling for corn and sugar beet, respectively. For the evaluation of the network trained with all plant species, any class different from the main crop was categorized as a weed.
For both crops, the binary training approach resulted in significantly lower recall compared to the network trained with all class labels, leading to a lower F2-score despite higher precision. Adjusting the threshold boundary for the binary classifier

Table 5.2: Classification performance for corn using different class label configurations.

Training	Precision	Recall	F2-Score
All classes	91.00 %	92.55 %	**92.24 %**
Binary, threshold = 0.5	95.19 %	82.02 %	84.35 %
Binary, threshold = 0.3	87.04 %	87.87 %	87.70 %
Binary, threshold = 0.1	63.49 %	94.36 %	86.00 %

Table 5.3: Classification performance for sugar beet using different class label configurations.

Training	Precision	Recall	F2-Score
All classes	82.16 %	82.82 %	**82.69 %**
Binary, threshold = 0.5	89.84 %	65.86 %	69.57 %
Binary, threshold = 0.3	78.86 %	73.08 %	74.17 %
Binary, threshold = 0.1	46.11 %	82.82 %	71.44 %

Table 5.4: Performance of the fine-tuned model with binary metrics for corn and sugar beet. Additional evaluation was conducted on image samples with bounding boxes smaller than 64^2 pixels.

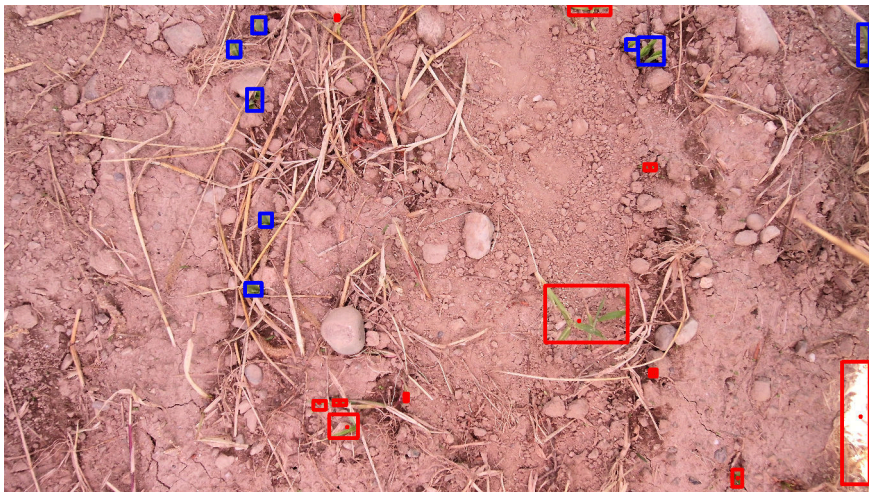
	Crop	Precision	Recall	F2-Score
All	Corn	95.28 %	94.57 %	94.71 %
	Sugar Beet	87.31 %	92.21 %	91.19 %
area $< 64^2$	Corn	84.75 %	83.68 %	83.89 %
	Sugar Beet	72.82 %	81.42 %	79.54 %

did not improve performance to match that of the model trained using all class labels. Consequently, the final model was trained using the complete set of classes. Fine-tuning the entire network resulted in a noticeable performance improvement as shown in Table 5.4.
A separate test set was created consisting of plants from the CAW dataset with an area below 64^2 pixels to evaluate performance on smaller samples. The network's performance on this subset was lower for both sugar beet and corn.

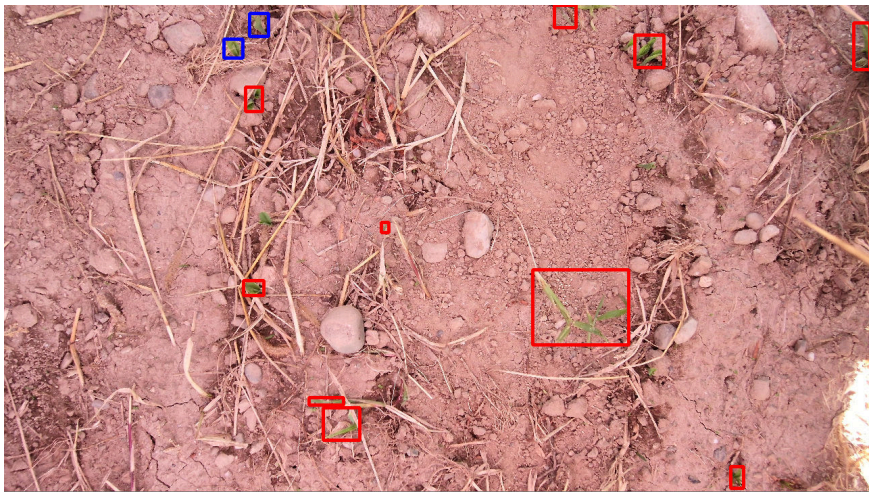
5.2 Visual Comparison of Weed Detection

Since we lack labeled data from our field tests, we conducted a qualitative analysis of the algorithm's performance. Two different scenarios are presented for evaluation: a cornfield with crops just emerging from the ground and a cornfield where the crops are in the multi-leaf stage.
In addition to our segmentation-based weed detection algorithm, we tested a pre-trained YOLOv7-E6 [72] object detection network on the same test images. We used a publically available checkpoint [73] of the model pre-trained on the CropAndWeed dataset. The aim of this comparison was to determine if an end-to-end model, trained on the same dataset, could outperform our segmentation-based approach.
In Figure 5.1b, we observe that all plants were assigned bounding boxes by our algorithm. The weed detection successfully identified all the weeds in the left row as corn plants. It is worth noting that the corn had just emerged from the ground making it challenging to distinguish. In the right crop row, the algorithm correctly identified the corn plant at the top of the image but misclassified the two other emerging corn plants as weeds.
The YOLOv7 model had more difficulty in this scnario, as shown in Figure 5.1c. It failed to detect several plants and frequently misclassified those it did identify. In the left crop row, YOLOv7 identified only two corn plants, despite there being five visible. For the remaining weeds, the performance of both approaches was similar.
An interesting observation in Figure 5.1 is that sunlight entered the right side of the workspace, illustrating a limitation of the threshold-based ExG segmentation used in our algorithm. This lighting change caused the algorithm to erroneously

(a) Color image



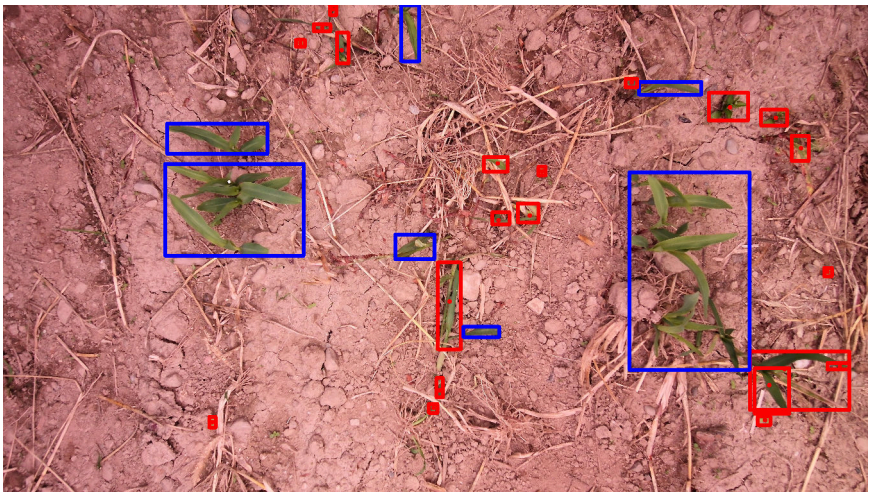
(b) Our weed detection



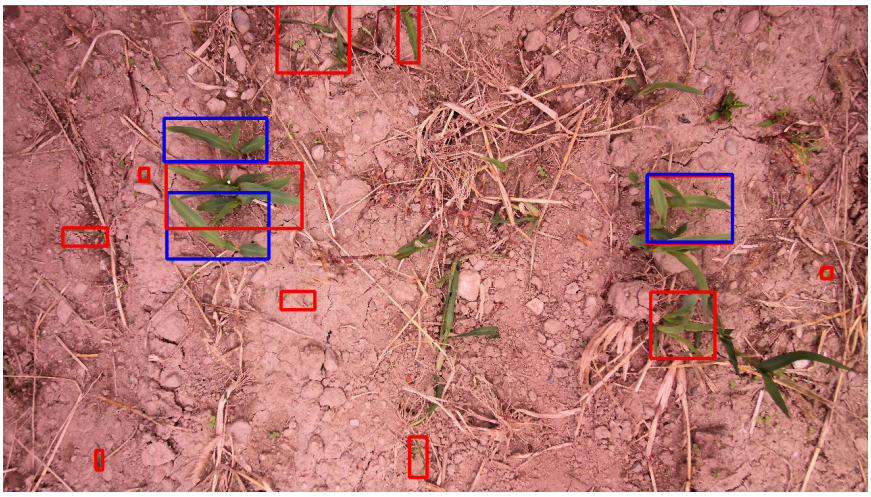
(c) YOLOv7 weed detection

Figure 5.1: Cornfield with crops at the early emergence stage. Red boxes indicate weeds, while blue boxes represent plants identified as corn.

(a) Color image



(b) Our weed detection



(c) YOLOv7 weed detection

Figure 5.2: Cornfield with crops in the multi-leaf growth stage. Red boxes indicate weeds, while blue boxes represent plants identified as corn.

Table 5.5: Positioning accuracy errors of the weed control unit. Measurements were taken from points captured by the depth camera at the back and the front of the workspace.

	Position (mm)		Errors (mm)			Euclidean Distance (mm)
	x	y	Δx	Δy	Δz	$\sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}$
Back	-40	-40	-6	-1	-3	6.78
	-40	0	-2	3	3	4.69
	-40	40	1	1	6	6.16
	0	-40	-8	0	-2	8.25
	0	0	-2	2	4	4.90
	0	40	2	0	5	5.39
Front	0	-40	2	-4	-7	8.31
	0	0	5	0	0	5.00
	0	40	8	-4	4	9.80
	40	-40	2	-3	-7	7.87
	40	0	4	-3	0	5
	40	40	7	-2	2	7.55
Average						**6.64**

segment that portion of the image as a plant.
For larger plants, as shown in Figure 5.2, the YOLOv7 model again encountered difficulties in detecting all plants. While it was capable of separating some plants that were in close proximity—something our algorithm struggles with—it had issues correctly classifying the plants and providing accurate bounding boxes.
Our algorithm was unable to separate the corn plants in the right crop row due to their close proximity. However, the network still correctly classified the cluster of all three plants as corn. This result indicates that the network is somewhat robust to incorrect segmentation of connected plant segments.

5.3 Positioning Accuracy of the System

To evaluate the positioning accuracy of the system, we placed a 3×3 grid of green dots on the ground, covering an area of 40 cm × 40 cm. The weed detection algorithm was run without the classification step and the offset between the center of each green dot and the actual position of the end-effector was measured.
Out of the nine dots on the grid, three were visible in both camera images, resulting in a total of 12 points for measurement. The positioning errors are reported in Table 5.5. The average Euclidean error was 6.64 mm. The measurements were repeated twice. However, repeatability data is not included as the deviation was minimal, approximately 1 mm, which is around the resolution of our measurement capability.

5.4 Positioning Speed of the Robot Arm

The velocity and acceleration limits configured for MoveIt 2 are shown in Table 5.6. The velocity limits were set to half of the acceleration limits. Assuming a trapezoidal velocity profile, this configuration ensures that movements taking less than one second are not velocity-limited, allowing for quicker transitions.
A higher acceleration was assigned to the lower revolute joint since it typically travels a longer distance than the upper revolute joint due to the additional upward

Table 5.6: Velocity and acceleration limits of the robot arm.

Joint	Velocity Limit	Acceleration Limit
Linear axis	$0.5\,{}^{\mathrm{m}}/_{\mathrm{s}}$	$1\,{}^{\mathrm{m}}/_{s^2}$
Upper revolute	$2.5\,{}^{\mathrm{rad}}/_{\mathrm{s}}$	$5\,{}^{\mathrm{rad}}/_{s^2}$
Lower revolute	$5\,{}^{\mathrm{rad}}/_{\mathrm{s}}$	$10\,{}^{\mathrm{rad}}/_{s^2}$

movements detailed in section 4.5. Various acceleration values were tested on all joints to ensure that no single axis limited the overall speed of the trajectory.
The time taken for the robot arm to move from one weed to another is influenced by the spacing between weeds. Manual measurements indicated that the average time to move between weeds was approximately 0.9 s, including the additional upward movement. This duration is within our target movement time of 1 s, as specified in subsection 1.2.3. The transition from one crop row to the other took slightly longer, around 1.5 s. However, this switch only needs to be performed once per workspace position, minimizing its impact on the overall operation time.

Chapter 6

Discussion

A primary objective of this thesis was to develop a weed detection algorithm capable of effectively distinguishing crops, such as corn and sugar beet, from other plants without the need to create a highly specific dataset tailored to our field conditions. Our initial field tests have yielded promising results for our weed detection approach. The algorithm successfully localized and classified both corn and weeds, even at an early growth stage. In contrast, the YOLOv7 model frequently failed to localize all plants and often misclassified them. Although we have yet to conduct a quantitative analysis, the images from our field tests consistently showed similar outcomes to those depicted in Figure 5.2 and Figure 5.1.

Several factors could explain why our approach outperformed YOLOv7. Our primary hypothesis is that end-to-end models are less adaptable to domain shifts compared to classification models, as they tackle the more complex problem of simultaneously localizing and classifying plants. Additionally, other factors may contribute to this difference in performance. For instance, our classification network was trained on a larger dataset, combining PlantCLEF and CAW, while the YOLOv7 model was trained solely on CAW. Resolution could also play a role. YOLOv7-E6 processes 1280×1280 images, while our classification network uses crops from the full 3840×2160 image, scaled to the network's input size of 518×518 pixels.

Further investigation is needed to determine which weed detection algorithm performs better and the reasons behind it. Our current evaluation is based on only two tests, both conducted in cornfields at a relatively early growth stage. To substantiate the claim that our method is more robust, additional testing across various conditions is essential.

The analysis of our weed classification training delivered a few interesting insights. Firstly, training on larger datasets seems to help, which is something that can be observed in many deep learning tasks. However, in our case, excluding all PlantCLEF images except those of leaves led to better performance on our test set. This confirmed our hypothesis that focusing on leaf images would enhance detection performance and allow us to significantly reduce dataset size and training time.

Table 5.3 and Table 5.2 demonstrate that providing the network with information on all plant species, rather than just distinguishing between crops and weeds, resulted in better performance on the binary classification task. While we only evaluated two extreme cases, it is possible that an optimal combination of classes exists that could further improve performance.

An additional benefit of using all classes during training is that our classifier can not only differentiate between crops and weeds but also identify each plant species, which could be valuable for monitoring weed populations. Furthermore, this approach allows the same pre-trained network to be used in fields with different crops,

requiring only an adjustment of the primary class.
Both our qualitative analysis and the performance metrics in Table 5.4 indicate that classifying smaller plants is more challenging. This makes intuitive sense, as plants are harder to distinguish during early growth stages, even for humans. Additionally, smaller plants have a lower resolution in the images, further complicating classification.
Regarding the positioning accuracy of our system, we measured an average error of 6.64 mm (see Table 5.5), which is slightly higher than the target accuracy of 5 mm. However, we observed that the error appears systematic. For example, on the left side of the workspace (negative y-axis), the z-coordinate was consistently too low, while on the other side, it was too high. This could be due to a physical misalignment between the camera and the robot arm or inaccuracies in the camera sensor's depth estimations. Addressing this issue could involve identifying and correcting the source of the misalignment or implementing a software-based mapping between commanded and actual positions. It should be feasible to reduce inaccuracies to below 5 mm. If not, an alternative approach would be to use visual servoing, as demonstrated by Michaels et al. [12].
In addition to the measured positioning error of the robot arm, there is also an inherent error because the plant's centroid and the weed stem are not always aligned. The original goal was to position the electrode 5 mm next to the weed stem, not the centroid. While this error was not quantified in this thesis, in most tests, the centroid appeared close to the stem. If future tests indicate that this approximation is inadequate, a learning-based approach for stem detection may be required [60].

Chapter 7

Conclusion and Future Work

In this thesis, an autonomous weed control unit was developed and tested, focusing on two main aspects: the design of a robotic arm capable of positioning an electrode near intra-row weeds without damaging the crops, and a weed detection algorithm robust to varying field conditions.
The 3 DOF robot arm features a cost-effective and unique design that is fast, accurate, and capable of navigating around crop leaves with minimal interference. The weed detection algorithm combines color-based vegetation segmentation with a classification network using a vision transformer. We mitigated common challenges associated with threshold-based segmentation by maintaining consistent lighting conditions within the workspace and leveraging 3D clustering for connected component analysis. Our trained classifier achieved an F2-score of 94.71 % for corn and 91.19 % for sugar beet on a representative dataset.
Initial field tests indicate promising results for the weed detection system. However, further testing under diverse field conditions is necessary to ensure the algorithm's robustness and accuracy in practical applications. A comprehensive analysis of the system's impact on weed and crop survival in typical field conditions will be crucial to determine its overall effectiveness.

7.1 Future Work

There are several avenues for future improvement of the weed control unit:

- **Field Testing and Validation**: To fully validate the robustness and practicality of the weed detection algorithm, extensive field testing is required in a variety of conditions. This should include different crop types, growth stages, and environmental factors to assess the system's adaptability and reliability. Additionally, quantitative analysis of weed and crop mortality rates will help evaluate the system's practical impact.

- **Robotic Arm Design Enhancements**: The robot arm's design could be further optimized, for instance, by implementing a toothed belt to drive the second joint, improving the arm's efficiency and reliability. During testing, it was observed that applying electric impulses to the weeds could disrupt the camera or motor operation due to the electric field. Future iterations should include shielding these components from electromagnetic interference to ensure consistent functionality. Additionally, the current use of tarps to shield the workspace from light is not ideal due to susceptibility to wind. Developing a more robust method to maintain consistent lighting would improve the system's performance.

- **Advanced Weed Detection Techniques**: Further exploration into enhancing the weed detection algorithm is warranted. For example, incorporating the spatial arrangement of crop rows into the detection process could improve accuracy [36]. Another approach would be utilizing a camera system closer to the ground to reduce occlusions caused by plant leaves. Moreover, there is a need for developing large-scale public datasets for weed detection to support the training of more robust models [48].
- **Ground Mesh for Motion Planning**: Currently, MoveIt 2 does not have information about the ground's position, resulting in a need to lift the end-effector after each move to avoid collisions. By using the depth image from the camera to create a mesh of the ground and plants, this mesh could be integrated into MoveIt 2 for collision checking. Providing this spatial information to the planner would allow for more efficient movement, eliminating unnecessary lifting and further minimizing interference with the plants.
- **Optimizing Plant Traversal**: Improving the traversal algorithm for plant treatment could significantly reduce operation time. Rather than simply sorting plants in the direction of the linear axis, a more efficient path-planning algorithm could be implemented to optimize the order in which weeds are treated [65].

7.2 Closing Remarks

This thesis has demonstrated the feasibility and potential of using weed detection combined with precise motion control to target individual weeds in agricultural fields. The system's development represents a step towards more sustainable and efficient agricultural practices. While there are areas for further enhancement, the groundwork laid by this research has the potential to drive future advancements in precision agriculture, ultimately contributing to reduced herbicide use and healthier crop yields.

Bibliography

[1] D. Reicosky, "Tillage-induced CO_2 emission from soil," *Nutrient Cycling in Agroecosystems*, vol. 49, no. 1, pp. 273–285, 1997.

[2] P. R. Hobbs, K. Sayre, and R. Gupta, "The role of conservation agriculture in sustainable agriculture," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 363, no. 1491, pp. 543–555, 2007.

[3] P. Kudsk and J. C. Streibig, "Herbicides – a two-edged sword," *Weed Research*, vol. 43, no. 2, pp. 90–102, 2003.

[4] D. C. Slaughter, D. K. Giles, and D. Downey, "Autonomous robotic weed control systems: A review," *Computers and Electronics in Agriculture*, vol. 61, no. 1, pp. 63–78, 2008.

[5] Z. Wu, Y. Chen, B. Zhao, X. Kang, and Y. Ding, "Review of Weed Detection Methods Based on Computer Vision," *Sensors*, vol. 21, no. 11, p. 3647, 2021.

[6] A. S. M. M. Hasan, F. Sohel, D. Diepeveen, H. Laga, and M. G. K. Jones, "A survey of deep learning techniques for weed detection from images," *Computers and Electronics in Agriculture*, vol. 184, p. 106067, 2021.

[7] B. Åstrand and A.-J. Baerveldt, "An Agricultural Mobile Robot with Vision-Based Perception for Mechanical Weed Control," *Autonomous Robots*, vol. 13, no. 1, pp. 21–35, 2002.

[8] N. D. Tillett, T. Hague, A. C. Grundy, and A. P. Dedousis, "Mechanical within-row weed control for transplanted crops using computer vision," *Biosystems Engineering*, vol. 99, no. 2, pp. 171–178, 2008.

[9] L. Quan, W. Jiang, H. Li, H. Li, Q. Wang, and L. Chen, "Intelligent intra-row robotic weeding system combining deep learning technology with a targeted weeding mode," *Biosystems Engineering*, vol. 216, pp. 13–31, 2022.

[10] M. Pérez-Ruíz, D. C. Slaughter, F. A. Fathallah, C. J. Gliever, and B. J. Miller, "Co-robotic intra-row weed control system," *Biosystems Engineering*, vol. 126, pp. 45–55, 2014.

[11] O. Bawden, J. Kulk, R. Russell, C. McCool, A. English, F. Dayoub, C. Lehnert, and T. Perez, "Robot for weed species plant-specific management," *Journal of Field Robotics*, vol. 34, no. 6, pp. 1179–1199, 2017.

[12] A. Michaels, S. Haug, and A. Albert, "Vision-based high-speed manipulation for robotic ultra-precise weed control," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5498–5505.

[13] F. Visentin, S. Cremasco, M. Sozzi, L. Signorini, M. Signorini, F. Marinello, and R. Muradore, "A mixed-autonomous robotic platform for intra-row and inter-row weed removal for precision agriculture," *Computers and Electronics in Agriculture*, vol. 214, p. 108270, 2023.

[14] W. S. Lee, D. C. Slaughter, and D. K. Giles, "Robotic Weed Control System for Tomatoes," *Precision Agriculture*, vol. 1, no. 1, pp. 95–113, 1999.

[15] T. Utstumo, F. Urdal, A. Brevik, J. Dørum, J. Netland, Ø. Overskeid, T. W. Berge, and J. T. Gravdahl, "Robotic in-row weed control in vegetables," *Computers and Electronics in Agriculture*, vol. 154, pp. 36–45, 2018.

[16] V. Partel, S. Charan Kakarla, and Y. Ampatzidis, "Development and evaluation of a low-cost and smart technology for precision weed management utilizing artificial intelligence," *Computers and Electronics in Agriculture*, vol. 157, pp. 339–350, 2019.

[17] S. K. Mathiassen, T. Bak, S. Christensen, and P. Kudsk, "The Effect of Laser Treatment as a Weed Control Method," *Biosystems Engineering*, vol. 95, no. 4, pp. 497–505, 2006.

[18] C. Wöltjen, H. Haferkamp, T. Rath, and D. Herzog, "Plant growth depression by selective irradiation of the meristem with CO_2 and diode lasers," *Biosystems Engineering*, vol. 101, no. 3, pp. 316–324, 2008.

[19] C. Marx, S. Barcikowski, M. Hustedt, H. Haferkamp, and T. Rath, "Design and application of a weed damage model for laser-based weed control," *Biosystems Engineering*, vol. 113, no. 2, pp. 148–157, 2012.

[20] S. Kaierle, C. Marx, T. Rath, and M. Hustedt, "Find and Irradiate — Lasers Used for Weed Control," *Laser Technik Journal*, vol. 10, no. 3, pp. 44–47, 2013.

[21] D. Tran, J. J. Schouteten, M. Degieter, J. Krupanek, W. Jarosz, A. Areta, L. Emmi, H. De Steur, and X. Gellynck, "European stakeholders' perspectives on implementation potential of precision weed control: the case of autonomous vehicles with laser treatment," *Precision Agriculture*, vol. 24, no. 6, pp. 2200–2222, 2023.

[22] G. R. Y. Coleman, A. Stead, M. P. Rigter, Z. Xu, D. Johnson, G. M. Brooker, S. Sukkarieh, and M. J. Walsh, "Using energy requirements to compare the suitability of alternative methods for broadcast and site-specific weed control," *Weed Technology*, vol. 33, no. 4, pp. 633–650, 2019.

[23] M. F. Diprose and F. A. Benson, "Electrical methods of killing plants," *Journal of Agricultural Engineering Research*, vol. 30, pp. 197–209, 1984.

[24] J. Blasco, N. Aleixos, J. M. Roger, G. Rabatel, and E. Moltó, "AE—Automation and Emerging Technologies: Robotic Weed Control using Machine Vision," *Biosystems Engineering*, vol. 83, no. 2, pp. 149–157, 2002.

[25] F. Abdelghafour, F. Rançon, S. Liu, J. Champ, V. De Rudnicki, C. Guizard, H. Goëau, C. Doussan, A. Joly, P. Bonnet, and G. Rabatel, "WeedElec: a robotic research platform for individual weed detection and selective electrical weeding," in *Precision agriculture '21*. Wageningen Academic Publishers, 2021, pp. 695–702.

[26] J. Underwood, M. Calleija, Z. Taylor, C. Hung, J. I. Nieto, R. Fitch, and S. Sukkarieh, "Real-time target detection and steerable spray for vegetable crops," in *Proceedings of the International Conference on Robotics and Automation: Robotics in Agriculture Workshop, Seattle, WA, USA*, 2015.

[27] H. Y. Jeon and L. F. Tian, "Direct application end effector for a precise weed control robot," *Biosystems Engineering*, vol. 104, no. 4, pp. 458–464, Dec. 2009.

[28] A. Wang, W. Zhang, and X. Wei, "A review on weed detection using ground-based machine vision and image processing techniques," *Computers and Electronics in Agriculture*, vol. 158, pp. 226–240, 2019.

[29] D. M. Woebbecke, G. E. Meyer, K. Von Bargen, and D. A. Mortensen, "Color indices for weed identification under various soil, residue, and lighting conditions," *Transactions of the American Society of Agricultural Engineers*, vol. 38, no. 1, pp. 259–269, 1995.

[30] E. Hamuda, M. Glavin, and E. Jones, "A survey of image processing techniques for plant extraction and segmentation in the field," *Computers and Electronics in Agriculture*, vol. 125, pp. 184–199, Jul. 2016.

[31] S. Haug, A. Michaels, P. Biber, and J. Ostermann, "Plant classification system for crop /weed discrimination without segmentation," in *IEEE Winter Conference on Applications of Computer Vision*, 2014, pp. 1142–1149.

[32] P. Lottes, M. Hoeferlin, S. Sander, M. Müter, P. Schulze, and L. C. Stachniss, "An effective classification system for separating sugar beets and weeds for precision farming applications," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5157–5163.

[33] F. Ahmed, H. A. Al-Mamun, A. S. M. H. Bari, E. Hossain, and P. Kwan, "Classification of crops and weeds from digital images: A support vector machine approach," *Crop Protection*, vol. 40, pp. 98–104, Oct. 2012.

[34] P. J. Herrera, J. Dorado, and Á. Ribeiro, "A Novel Approach for Weed Type Classification Based on Shape Descriptors and a Fuzzy Decision-Making Method," *Sensors*, vol. 14, no. 8, pp. 15 304–15 324, 2014.

[35] A. Bakhshipour and A. Jafari, "Evaluation of support vector machine and artificial neural networks in weed detection using shape features," *Computers and Electronics in Agriculture*, vol. 145, pp. 153–160, 2018.

[36] P. Lottes and C. Stachniss, "Semi-supervised online visual crop and weed classification in precision farming exploiting plant arrangement," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5155–5161.

[37] H. S. Midtiby, B. Åstrand, O. Jørgensen, and R. N. Jørgensen, "Upper limit for context–based crop classification in robotic weeding applications," *Biosystems Engineering*, vol. 146, pp. 183–192, 2016.

[38] A. Milioto, P. Lottes, and C. Stachniss, "Real-time blob-wise sugar beets vs weeds classification for monitoring fields using convolutional neural networks," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-2-W3, pp. 41–48, 2017.

[39] M. Dyrmann, H. Karstoft, and H. S. Midtiby, "Plant species classification using deep convolutional neural network," *Biosystems Engineering*, vol. 151, pp. 72–80, 2016.

[40] A. Milioto, P. Lottes, and C. Stachniss, "Real-Time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2229–2235.

[41] C. Potena, D. Nardi, and A. Pretto, "Fast and Accurate Crop and Weed Identification with Summarized Train Sets for Precision Agriculture," in *Intelligent Autonomous Systems 14*, W. Chen, K. Hosoda, E. Menegatti, M. Shimizu, and H. Wang, Eds. Cham: Springer International Publishing, 2017, pp. 105–121.

[42] M. Dyrmann, A. K. Mortensen, H. Midtiby, and R. Jørgensen, "Pixel-wise classification of weeds and crops in images by using a Fully Convolutional neural network," in *Proceedings of the International Conference on Agricultural Engineering, Aarhus, Denmark*, 2016.

[43] X. Jin, J. Che, and Y. Chen, "Weed Identification Using Deep Learning and Image Processing in Vegetable Plantation," *IEEE Access*, vol. 9, pp. 10 940–10 950, 2021.

[44] A. Ahmad, D. Saraswat, V. Aggarwal, A. Etienne, and B. Hancock, "Performance of deep learning models for classifying and detecting common weeds in corn and soybean production systems," *Computers and Electronics in Agriculture*, vol. 184, p. 106081, 2021.

[45] L. Lac, J.-P. Da Costa, M. Donias, B. Keresztes, and A. Bardet, "Crop stem detection and tracking for precision hoeing using deep learning," *Computers and Electronics in Agriculture*, vol. 192, p. 106606, 2022.

[46] J. Champ, A. Mora-Fallas, H. Goëau, E. Mata-Montero, P. Bonnet, and A. Joly, "Instance segmentation for the fine detection of crop and weed plants by precision agricultural robots," *Applications in Plant Sciences*, vol. 8, no. 7, p. e11373, 2020.

[47] M. Halstead, A. Ahmadi, C. Smitt, O. Schmittmann, and C. McCool, "Crop Agnostic Monitoring Driven by Deep Learning," *Frontiers in Plant Science*, vol. 12, 2021.

[48] B. Deng, Y. Lu, and J. Xu, "Weed database development: An updated survey of public weed datasets and cross-season weed detection adaptation," *Ecological Informatics*, vol. 81, p. 102546, 2024.

[49] F. Magistri, J. Weyler, D. Gogoll, P. Lottes, J. Behley, N. Petrinic, and C. Stachniss, "From one field to another—Unsupervised domain adaptation for semantic segmentation in agricultural robotics," *Computers and Electronics in Agriculture*, vol. 212, p. 108114, 2023.

[50] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Eds., *Robotics: Modelling, Planning and Control*. Springer, 2009.

[51] C. Neupane, A. Koirala, Z. Wang, and K. B. Walsh, "Evaluation of Depth Cameras for Use in Fruit Localization and Sizing: Finding a Successor to Kinect v2," *Agronomy*, vol. 11, no. 9, p. 1780, 2021.

[52] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[53] P. Mesmer, M. Neubauer, A. Lechler, and A. Verl, "Robust design of independent joint control of industrial robots with secondary encoders," *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102232, 2022.

[54] D. Collins, "Velocity control loop tuning for servo systems," https://www.motioncontroltips.com/how-are-servo-system-velocity-control-loops-tuned, 2017, accessed: 2024-09-14.

[55] "Writing a Hardware Component," https://control.ros.org/rolling/doc/ros2_control/hardware_interface/doc/writing_new_hardware_component.html, accessed: 2024-09-14.

[56] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, August 2010.

[57] J. Kuffner and S. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning." in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 2000, pp. 995–1001.

[58] T. Kunz and M. Stilman, "Time-Optimal Trajectory Generation for Path Following with Bounded Acceleration and Velocity," in *Robotics: Science and Systems VIII*. The MIT Press, 2013.

[59] A. Ahmadi, M. Halstead, and C. McCool, "BonnBot-I: A Precise Weed Management and Crop Monitoring Platform," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 9202–9209.

[60] P. Lottes, J. Behley, N. Chebrolu, A. Milioto, and C. Stachniss, "Robust joint stem detection and crop-weed classification using image sequences for plant-specific treatment in precision farming," *Journal of Field Robotics*, vol. 37, no. 1, pp. 20–34, 2020.

[61] H. Goëau, P. Bonnet, and A. Joly, "Overview of PlantCLEF 2023: Image-based Plant Identification at Global Scale," in *CLEF-WN 2023 - 24th Working Notes of the Conference and Labs of the Evaluation Forum*, ser. Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2023), vol. 3497. Thessalonique, Greece: Mohammad Aliannejadi and Guglielmo Faggioli and Nicola Ferro and Michalis Vlachos, 2023, pp. 1972–1981.

[62] G. Coleman, W. Salter, and M. Walsh, "OpenWeedLocator (OWL): an open-source, low-cost device for fallow weed detection," *Scientific Reports*, vol. 12, no. 1, p. 170, 2022.

[63] J. Peñuelas and I. Filella, "Visible and near-infrared reflectance techniques for diagnosing plant physiological status," *Trends in Plant Science*, vol. 3, no. 4, pp. 151–156, 1998.

[64] X. Zhang, N. Li, L. Ge, X. Xia, and N. Ding, "A Unified Model for Real-Time Crop Recognition and Stem Localization Exploiting Cross-Task Feature Fusion," in *2020 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2020, pp. 327–332.

[65] Y. Xiong, Y. Ge, Y. Liang, and S. Blackmore, "Development of a prototype robot and fast path-planning algorithm for static laser weeding," *Computers and Electronics in Agriculture*, vol. 142, pp. 494–503, 2017.

[66] A. Joly, L. Picek, S. Kahl, H. Goëau, V. Espitalier, C. Botella, B. Deneu, D. Marcos, J. Estopinan, C. Leblanc, T. Larcher, M. Šulc, M. Hrúz, M. Servajean, J. Matas, H. Glotin, R. Planqué, W.-P. Vellinga, H. Klinck, T. Denton, A. M. Durso, I. Eggel, P. Bonnet, and H. Müller, "LifeCLEF 2024 Teaser: Challenges on Species Distribution Prediction and Identification," in *Advances in Information Retrieval*, N. Goharian, N. Tonellotto, Y. He, A. Lipani, G. McDonald, C. Macdonald, and I. Ounis, Eds. Cham: Springer Nature Switzerland, 2024, pp. 19–27.

[67] D. Steininger, A. Trondl, G. Croonen, J. Simon, and V. Widhalm, "The CropAndWeed Dataset: a Multi-Modal Learning Approach for Efficient Crop and Weed Manipulation," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2023, pp. 3718–3727, iSSN: 2642-9381.

[68] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," 2021.

[69] H. Goëau, P. Bonnet, and A. Joly, "Overview of PlantCLEF 2022: Image-based plant identification at global scale," in *CLEF 2022 - Conference and Labs of the Evaluation Forum*, ser. CEUR Workshop Proceedings, vol. 3180, no. 153, Bologne, Italy, 2022, pp. 1916–1928.

[70] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "DINOv2: Learning Robust Visual Features without Supervision," 2024.

[71] M. Contributors, "Openmmlab's pre-training toolbox and benchmark," https://github.com/open-mmlab/mmpretrain, 2023.

[72] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022.

[73] Daraan, "CropAndWeedDetection," https://github.com/Daraan/CropAndWeedDetection, 2024.

Chapter 8

Datasheets

CubeMars AK70-10

Specifications

Application	Legged Robot,Exoskeleton,AGV	Peak torque (Nm)	24.8
Driving way	FOC	Peak current (ADC)	23.2
Operation ambient temperature	-20℃~50℃	Kv (rpm/V)	100
Winding type	Delta	Kt (Nm/A)	0.123
Insulation class	H	Ke (V/krpm)	11.2
Insulation High-voltage	1000V 5mA/2s	Phase to Phase resistance (mΩ)	272
Insulation resistance	1000V10MΩ	Phase to Phase inductance (μH)	113
Phase	3	Inertia (gcm^2)	414
Pole pairs	21	Km (Nm/√W)	0.24
Reduction ratio	10:1	Mechanical time constant (ms)	0.74
Back drive(Nm)	0.48	Electrical time constant (ms)	0.42
Backlash (°)	0.12	Weight (g)	521
Temperature sensor	NTC MF51B 103F3950	Maximum torque weight ratio (Nm/kg)	47.6
Noise dB 65CM away the motor (Environment 45dB@1200rpm)	58	CAN connector	A1257WR-S-4P
Basic load ratings (dyn. C) N	2000	UART connector	A1257WR-S-3P
Basic load ratings (stat.C_0) N	2520	Power connector	XT30PW-M
Rated voltage (V)	24/48	Inner loop encoder type	Magnetic encoder
Rated torque (Nm)	8.3	Inner ring encoder resolution	14bit
Rated speed (rpm)	148/310	Outer ring encoder type	-
Rated current (ADC)	7.2	Outer ring encoder resolution	-
		Number of encoder	1

Teknic CPM-SCSK-2321S-EQNA

Specifications at 24 VDC	
Peak Torque	191 oz-in (1.3 N-m)
Cont. (RMS) Torque	98 oz-in (0.7 N-m)
Max Speed	1010 RPM
Peak Power	39 W (0.05 hp)
Cont. (RMS) Power	39 W (0.05 hp)

More Specifications

Motor Frame Size	NEMA 23 - 2.33 in (59.18 mm) sq.
Length	4.09 in (103.89 mm)
Input (bus) Voltage Range	24-75 VDC (90 VDC max)
Peak Torque (@75 VDC)	492 oz-in (3.5 N-m)
Cont. (RMS) Torque (@75 VDC)	98 oz-in (0.7 N-m)
Max Speed (@75 VDC)	3170 RPM
Achievable Resolution	0.057 degrees
Repeatability	0.03 degrees
Shaft Diameter	0.250 in (6.35 mm)
Weight	2.1 lb (0.9 kg)
Rotor Inertia	0.9 oz-in^2 (0.2 kg-cm^2)
General Purpose Input Voltage Range	4.0 to 28 VDC
Maximum Radial Load	25 lbf (111.2 N)
Maximum Thrust Load	5.0 lbf (22.2 N)
Ambient Temperature [1]	-40° to +70° C
Ambient Humidity	0-95%; non-condensing
Environmental Protection (IP Rating)	IP53
Regulatory Certifications	UL recognized; cUL recognized; CE; RoHS
Safety [2]	Supports STO (IEC 60204-1) Safety Requirements
Country of Origin	USA
Warranty	3 Years

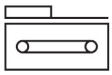
[1]The RMS torque limit on certain motors is derated for operation in ambient temperatures above +40 degrees C. Contact Teknic for derating assistance.

[2]Wired according to Teknic's instructions, ClearPath surpasses STO (IEC 60204-1) Safe Stop 0 safety requirements. Contact Teknic for more information.

Toothed belt axis
ELGC-TB-KF-60-1000

Part number: 8062781

FESTO



General operating condition

Data sheet

Feature	Value
Effective diameter of drive pinion	24.83 mm
Working stroke	1000 mm
Size	60
Stroke reserve	0 mm
Toothed-belt pitch	3 mm
Mounting position	optional
Guide	Recirculating ball bearing guide
Design	Electromechanical linear axis With toothed belt
Type of motor	Stepper motor Servo motor
Symbol	00991212
Position detection	Via proximity switch Via inductive sensors
Max. acceleration	15 m/s²
Max. speed	1.5 m/s
Repetition accuracy	±0.1 mm
Duty cycle	100%
LABS (PWIS) conformity	VDMA24364 zone III
Suitability for the production of Li-ion batteries	Metals with more than 1% by mass of copper, zinc or nickel by mass are excluded from use. Exceptions are nickel in steel, chemically nickel-plated surfaces, printed circuit boards, cables, electrical plug connectors and coils
Cleanroom class	Class 7 according to ISO 14644-1
Storage temperature	-20 °C ... 60 °C
Degree of protection	IP40
Ambient temperature	0 °C ... 50 °C
Impact energy in end positions	2.5E-4 J
Note on the impact energy in the end positions	At maximum homing speed of 0.01 m/s
2nd moment of area Iy	441000 mm⁴
2nd moment of area Iz	542000 mm⁴
Max. drive torque	1.49 Nm
Max. force Fy	3641 N
Max. force Fz	3641 N
Max. force Fy total axis	600 N
Max. force Fz total axis	1800 N
Fy at theoretical life value of 100 km (only guide consideration)	13400 N

Feature	Value
Fz at theoretical life value of 100 km (only guide consideration)	13400 N
Max. idle running transfer resistance	15.6 N
Max. moment Mx	29.1 Nm
Max. moment My	31.8 Nm
Max. moment Mz	31.8 Nm
Max. moment Mx total axis	29.1 Nm
Max. moment My total axis	31.8 Nm
Max. moment Mz total axis	31.8 Nm
Mx at theoretical life value of 100 km (only guide consideration)	107 Nm
My at theoretical life value of 100 km (only guide consideration)	117 Nm
Mz at theoretical life value of 100 km (only guide consideration)	117 Nm
Distance between slide surface and guide centre	54.6 mm
Max. feed force Fx	120 N
Frictional torque independent of load	0.194 Nm
Torsional mass moment of inertia It	29800 mm^4
Mass moment of inertia JH per metre of stroke	0.0851 $kgcm^2$
Mass moment of inertia JL per kg of working load	1.5411 $kgcm^2$
Mass moment of inertia JO	0.8804 $kgcm^2$
Feed constant	78 mm/U
Reference service life	5000 km
Maintenance interval	Life-time lubrication
Moving mass	482 g
Weight of slide	139 g
Product weight	6027 g
Basic weight for 0 mm stroke	1775 g
Additional weight per 10 mm stroke	43 g
Dynamic deflection (moving load)	0.05% of the axis length, max. 0.5 mm
Static deflection (load in standstill)	0.1% of the axis length
Interface code, actuator	T42
Material end cap	Painted die cast aluminium
Material profile	Anodised wrought aluminium alloy
Note on materials	RoHS-compliant
Material cover tape	Stainless steel strip
Material drive cover	Painted die cast aluminium
Material guide slide	Steel
Material guide rail	Steel
Material pulleys	High-alloy stainless steel
Material slide	Die-cast aluminium
Material toothed belt	Polychloroprene with glass fibre

Orbbec Femto Bolt

ORBBEC

2. Product Specifications

Parameter	Specifications
Model	F00364-152
VID/PID	0x2BC5/0x066B
Technology	iToF
Shutter Type	IR: Global Shutter; Color: Rolling Shutter
Wavelength	850nm

Mode	Resolution	FoV	FPS	Range	Format
WFOV Unbinned	1024 x 1024	H 120° x V 120°	5, 15	0.25m – 2.21m	Y16
WFOV Binned	512 x 512		5, 15, 25, 30	0.25m –2.88m	
NFOV Unbinned	640 x 576	H 75° x V 65°	5, 15, 25, 30	0.5m – 3.86m	
NFOV Binned	320 x 288		5, 15, 25, 30	0.5m – 5.46m	
RGB	3840 x 2160	16:9 H 80°x V 51°	5, 15, 25, 30		YUY2, MJPG, H.264, H.265
	2560 x 1440		5, 15, 25, 30		
	1920 x 1080		5, 15, 25, 30		
	1280 x 720		5, 15, 25, 30		
	1280x960	4:3 H 65° x V 51°	5, 15, 25, 30		

Depth typical systematic error (accuracy)* < 11 mm + 0.1% distance
Depth random error std. dev.(precision)* ≤ 17 mm,
Passive-IR mode supported
*15% to 95% reflectivity at 850nm, 2.2 µW/cm2/nm without multi-path interference.
Depending on object reflectivity, depth may be provided outside of the operating range indicated above.

Parameter	Specifications
IMU	6 DoF; Frequency range: 50-2,000Hz; Data format: float
Mirror Mode	Supported, non-mirror by default
Processing	Host
Data Connection	Type-C USB 3.2 Gen 1
Trigger Sync	8 Pin-Connector for multi-device sync
Power Mode	DC /USB Type-C
Power Input	DC 12V 2A Type-C 5V 3A
Operating Modes	DC Power + USB Type-C Data Type-C Power + USB Type-C Data*[1] *[1] Depth & IR mode support 640 x 576 and below, Y16 format. Color mode supports 1,920 x 1,080 and below, YUY2 and MJPG format.
Power Consumption	4.35W average
Operating Environment	10°C - 25°C, 8%RH ~ 90%RH (non-condensing), Indoor/Semi-Outdoor
Supported Functions	D2C, Multi-Camera Sync
Dimensions	115mm x 40mm x 65mm
Weight	335g
Certifications	ROHS, Reach, WEEE, CE, FCC, Class 1 Laser Product
Installation	Bottom: 1 x ¼-20unc Sides: 4 x M2.5

ORBBEC. All Rights Reserved. https://orbbec3d.com/
Note: Orbbec reserves the right to change any information in the document without prior notice.
Contact us: info@orbbec3d.com | 2800 Livernois Rd, Troy, MI 48083 USA

Appendix A

Hyperparameters

Listing A.1: MMPretrain example config file. Apart from the dataset and classification head, the same hyperparemeters were used for all of our training.

```
auto_scale_lr = dict(base_batch_size=1024)
data_preprocessor = dict(
    mean=[
        123.675,
        116.28,
        103.53,
    ],
    num_classes=7806,
    std=[
        58.395,
        57.12,
        57.375,
    ],
    to_rgb=True)
default_hooks = dict(
    checkpoint=dict(interval=1, max_keep_ckpts=5, type='
        ↪ CheckpointHook'),
    logger=dict(interval=25, type='LoggerHook'),
    param_scheduler=dict(type='ParamSchedulerHook'),
    sampler_seed=dict(type='DistSamplerSeedHook'),
    timer=dict(type='IterTimerHook'),
    visualization=dict(enable=False, type='VisualizationHook'))
default_scope = 'mmpretrain'
env_cfg = dict(
    cudnn_benchmark=False,
    dist_cfg=dict(backend='nccl'),
    mp_cfg=dict(mp_start_method='fork', opencv_num_threads=0))
load_from = None
log_level = 'INFO'
model = dict(
    backbone=dict(
        arch='base',
        frozen_stages=12,
        img_size=518,
        init_cfg=dict(
            checkpoint=
```

```
            'https://download.openmmlab.com/mmpretrain/v1.0/dinov2/vit
                ↪ -base-p14_dinov2-pre_3rdparty_20230426-ba246503.pth
                ↪ ',
            prefix='backbone',
            type='Pretrained'),
        layer_scale_init_value=1e-05,
        patch_size=14,
        type='VisionTransformer'),
    head=dict(
        in_channels=768,
        loss=dict(type='CrossEntropyLoss'),
        num_classes=7806,
        topk=(
            1,
            5,
        ),
        type='VisionTransformerClsHead'),
    neck=None,
    type='ImageClassifier')
optim_wrapper = dict(
    loss_scale='dynamic',
    optimizer=dict(lr=0.0001, type='AdamW', weight_decay=0.05),
    paramwise_cfg=dict(
        custom_keys=dict(
            cls_token=dict(decay_mult=0.0), mask_token=dict(decay_mult
                ↪ =0.0))),
    type='AmpOptimWrapper')
param_scheduler = [
    dict(T_max=20, by_epoch=True, type='CosineAnnealingLR'),
]
randomness = dict(deterministic=True, seed=0)
resume = False
test_cfg = dict()
test_dataloader = dict(
    batch_size=1024,
    dataset=dict(
        ann_file='annotation/test.txt',
        data_prefix='/mnt/data',
        pipeline=[
            dict(type='LoadImageFromFile'),
            dict(
                backend='pillow',
                edge='short',
                interpolation='bicubic',
                scale=518,
                type='ResizeEdge'),
            dict(crop_size=518, type='CenterCrop'),
            dict(type='PackInputs'),
        ],
        type='CustomDataset'),
    num_workers=20,
    persistent_workers=True,
    sampler=dict(shuffle=True, type='DefaultSampler'))
test_evaluator = [
```

```
    dict(topk=(
        1,
        5,
    ), type='Accuracy'),
    dict(type='SingleLabelMetric'),
]
test_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(
        backend='pillow',
        edge='short',
        interpolation='bicubic',
        scale=518,
        type='ResizeEdge'),
    dict(crop_size=518, type='CenterCrop'),
    dict(type='PackInputs'),
]
train_cfg = dict(by_epoch=True, max_epochs=12, val_interval=1)
train_dataloader = dict(
    batch_size=1024,
    dataset=dict(
        ann_file='annotation/train.txt',
        data_prefix='/mnt/data',
        pipeline=[
            dict(type='LoadImageFromFile'),
            dict(
                backend='pillow',
                edge='short',
                interpolation='bicubic',
                scale=518,
                type='ResizeEdge'),
            dict(crop_size=518, type='CenterCrop'),
            dict(type='PackInputs'),
        ],
        type='CustomDataset'),
    num_workers=28,
    persistent_workers=True,
    sampler=dict(shuffle=True, type='DefaultSampler'))
train_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(
        backend='pillow',
        interpolation='bicubic',
        scale=518,
        type='RandomResizedCrop'),
    dict(direction='horizontal', prob=0.5, type='RandomFlip'),
    dict(type='PackInputs'),
]
val_cfg = dict()
val_dataloader = dict(
    batch_size=1024,
    dataset=dict(
        ann_file='annotation/val.txt',
        data_prefix='/mnt/data',
```

```
        pipeline=[
            dict(type='LoadImageFromFile'),
            dict(
                backend='pillow',
                edge='short',
                interpolation='bicubic',
                scale=518,
                type='ResizeEdge'),
            dict(crop_size=518, type='CenterCrop'),
            dict(type='PackInputs'),
        ],
        type='CustomDataset'),
    num_workers=28,
    persistent_workers=True,
    sampler=dict(shuffle=True, type='DefaultSampler'))
val_evaluator = [
    dict(topk=(
        1,
        5,
    ), type='Accuracy'),
    dict(type='SingleLabelMetric'),
]
vis_backends = [
    dict(type='LocalVisBackend'),
]
visualizer = dict(
    type='UniversalVisualizer', vis_backends=[
        dict(type='LocalVisBackend'),
    ])
```