Delft University of Technology
Master of Science Thesis in Embedded Systems

Automation of Repetative Tasks in Tomato Greenhouses

Emiel Rik van der Meijs

TU Delft Delft University of Technology

Author
Emiel Rik van der Meijs (E.R.vandermeijs@student.tudelft.nl)
(emiel@van-der-meijs.nl)

Title

Automation of Repetative Tasks in Tomato Greenhouses

MSc Presentation Date

July 6 2022

Graduation Committee

Koen Langendoen	Delft University of Technology
Chris Verhoeven	Delft University of Technology
Stijn Bosma	Lely Technologies

Abstract

Labour in greenhouses is in short supply, while a large portion of the work is still being done manually. To speed up the work and guarantee the food supply of the future, Lely has started a project on tomato greenhouse automation. The aim of the project is to create a robot that can perform multiple repetitive tasks simultaneously, to allow for a large portion of the weekly work to be automated.

Due to the conditions in the greenhouse, the decision was made to create a specialised robot arm. The hardware and software design of this robot arm will be discussed in this thesis. Arm control including inverse kinematics was built from scratch to optimise for a SCARA type robot arm while keeping the required processing power low.

To evaluate the arm design, lab tests were conducted on the greenhouse task of growth guidance. In this task clips are used to attach the plant to a rod hanging from the roof. During the lab test, such a clip was repeatedly placed and retrieved by the designed robot arm. An average of 20 consecutive successful runs has been achieved. Although these results are promising, there is room for improvement in both the detection of the clips and rod, as well as the smooth control of the robot arm.

Contents

Chapter 1

Introduction

Growing crops in greenhouses requires a lot of labour. Many people travel a long way to come work in greenhouses in Holland. A few years ago people came from Poland, but nowadays many Polish people are wealthy enough and the migrant workers come from further away. Clearly there will not be enough labour forever. This is worsened as the world population is still growing, with the food consumption along with it.

The most obvious task in a greenhouse is picking the fruit, but a spectra of others jobs are required to make sure the yield is optimal. To address the problem of decreasing labour, automation of repetitive jobs in the greenhouse is required. Some automation is already being worked on with robots cutting leaves and picking fruits, but all existing robots are only able to execute a single task. These projects will be discussed in Section 2.1. In order to automate a larger portion of the repetitive tasks and to do so in an economically viable manner, a robotic system that can do multiple tasks should be designed. The thesis will be focused on tomato plants as this is the largest crop by area. This will be discussed in Section 1.2 after discussing some challenges that arise with greenhouse automation in general.

1.1 Greenhouse automation

For the automation of labour in a greenhouse, a tailor-made solution has to be developed. This is because automating a greenhouse is different from other domains and difficult for several reasons that are specific to the greenhouse domain.

First, a plant grows in a certain pattern, but the exact size and length are different for each plant. Compare this to an industrial application where robots can do things like welding or assembly. This is possible because a workspace in an assembly line is very organised and every measurement is known exactly. Furthermore, a lot of these kinds of automation tasks deal with rigid objects, whereas a plant is very flexible and fragile. A single scratch on the skin of a plant can cause it to dry out and which in turn causes a reduction in fluids reaching the top of the plant with a reduction of growth and yield as a result. When a plant is dropped on the floor or destroyed another way, the produce of this whole plant is lost for the whole season, so during operation, the survival

of the plant should be a priority.

Another issue is the size of plants such as tomato or bell pepper plants. With smaller crops such as lettuce or houseplants, they can be placed on sleds. These sleds can be transported throughout the greenhouse and be retrieved when maintenance is required [5]. This way the plants come to the robots and each robot can have its own workspace where it can freely move around the plant if necessary and the robot does not need to move throughout the greenhouse.

However, with crops like tomatoes and bell peppers, the plants grow very long so they have to stay in place. This means the robot should come to the plants by moving through the greenhouse rows. In a row, see Figure 1.1, the space for the robot to turn in is much more constricted compared to a robot workspace in a separate warehouse attached to the greenhouse as used for small plants.

This cluttered environment also creates the issue that a plant can be partly occluded by other plants in the row and a object that was not seen before now needs to be avoided. Therefore, the trajectory should be adaptable whilst moving. This sets a constraint on the computation time of the planning algorithm that creates the motion that is each joint of the arm will need to follow in order for the end-point of the arm to reach a certain location. This planning algorithm will be referred to as path planning.

Lastly, the robots contact the plants should be minimised during its operation. This is important because viruses are a serious threat to the greenhouse plants and they are easily spread to other plants. These viruses are of large concern for the grower, as they can cause serious damage. When workers go into the greenhouse, they need to wear a special set of clothes and disinfect their hands and shoes, yet the plants still get infected sometimes. To prevent the robot from doing the same, it should avoid the plants or have a disinfection system. Avoiding the plant has the additional benefit that it minimises the change of the robot damaging the plants.

1.2 Tomato greenhouse operation

Lely, a company currently focused on dairy farm automation [9], is interested in entering the field of greenhouse automation. To start with, the work will focus on tomato greenhouses, as this is the largest greenhouse crop by total area in the Netherlands compared to other fruits such as cucumbers and bell peppers [1]. However, the differences between these crops are subtle, thus much of the research still applies for these other fruits. This means that after successfully applying the robot in a tomato greenhouse, branching of to other fruits should be easy, if the core design is solid. The design presented here is tested using a prototype, a simplified version of the final design where the core functionality is verified.

In this section the general layout of a modern greenhouse will be discussed, after which the different greenhouse tasks will be listed.

1.2.1 Greenhouse layout

To make effective use of the space in a greenhouse, plants are hung vertically in rows. During the growth of the plant, the top needs to be guided continuously

Figure 1.1: **Example of a plant row inside a greenhouse. In the middle on the floor the heating pipes are visible.**

upwards. One way to achieve this is by wrapping the plant around a string. This string is hung from a wire that is attached to the roof of the greenhouse. The plants grow longer as the greenhouse is high, so the extra piece of stem is guided horizontally at the bottom. This bottom piece is a bare stem, as will be explained in Subsection 1.2.4.

In most greenhouses there are heating pipes on the floor after every two rows of plants, which are also used as transport rails by the people working in the greenhouse. A picture of a row in a greenhouse is shown in Figure 1.1. Between the two plant rows there is a trough with substrate, where the plant roots are located in. The through also has droppers, which give minerals and water to the plants through their roots. The transport rails should be used by the robot system to drive along, as this would allow greenhouse owners to start using the system without extra investments for the transport of the robot. The under-carriage itself is outside the scope of the project, but should be compatible with this design philosophy.

The substrate trough is used to supply water and nutrients to the plants through their roots. This process is already fully automated and can possibly be further optimised with commercially available systems and for that reason will not be further discussed in this thesis.

1.2.2 Growth guidance

Tomato plants cannot grow vertically on their own. Their stems are made to grow along the ground instead. To increase productivity per square meter, the plants are guided with a wire that is attached to the roof of a greenhouse. The top of each plant is wrapped around the string every week to force vertical

growth. When the plants reach the top of a greenhouse, the tops of the plants are moved sideways in a row to allow the bottoms of the plants to lay horizontally in hooks. The plants can get 13 meters long, at which point most of the plant is laying horizontally in plant hooks. This part of the plant is only used to transport fluids towards the productive part of the plant.

Using a wire to wrap the plant around is one solution for guiding the plants. Another, but far less common solution, is to have a piece of metal wire hanging from the ceiling towards the top of the plant, where the plant is then attached to this wire with clips. These clips can be continuously reused by moving them upwards along the plant. The clips can also be used for multiple seasons, which is not the case with the plastic wire that is thrown out together with the plants. However, the clips are more expensive compared to the wire and the growth guidance task supposedly takes longer when using them, thus most growers do not use them.

That being the case, wrapping the wire around the plant with getting any of the branches stuck is a vary nuanced process, which a human can do easily, but will be difficult to automate. With the clippers, the process is much more routinely and thus thought to be easier to automate with a robot.

1.2.3 Sucker removal

On the node of a plant, an extra offshoot can grow which is referred to as a sucker. This offshoot will become a whole new plant if it is left to grow. The first suckers that appear are given their own growth guiding wire, which means that there will now effectively be multiple plants growing from a single set of roots. Using this method, the offshoots are used to increase plant density in the greenhouse without putting down more plant roots.

However, these suckers will keep appearing and growing during the season. This is undesired, as the greenhouse will get overgrown with plants that their roots cannot supply with enough nutrients and water, meaning the extra plants will not increase the yield per area. Hence, they should be removed as soon as possible to prevent energy going to waste.

1.2.4 Leave removal

On the bottom part of the plant where the tomatoes grow, the leaves are removed. This is done to give the tomatoes more light and to make them more accessible for harvesting. The bundle of stems at the bottom is also more manageable because of the fact that the stems are bare.
This process is usually performed with scissors, where the cut should be made close to the stem to make the wound of the plant is as small as possible and the stump that will dry out is short.

1.3 Problem statement

So there are three labour intensive tasks in tomato crop maintenance, besides harvesting: removing suckers, removing leaves, and guiding the plant. Because of the large amount of work required to maintain a healthy crop, the vision to automate some parts of the work have long existed. The problem is, however,

that there are many tasks that all require a significant amount of work. This means that thought should be given to multiple tasks simultaneously when designing a robot to prevent the need for the grower to invest in many different robots.

In order to achieve this, the design requirement is set that the robot should have multiple robot arms that all can perform one or preferably more of the tasks. If a task is difficult, the arms should also be able to work together on it. This way, even if one full robot arm is added to automate one extra task, this robot arm alone is still a lot cheaper compared to a full robot with undercarriage and batteries. The arms can be put to use even more effectively if they are multipurpose. With a design concept where multiple robot arms are all attached to a vertical pole, different arms are at different heights of the greenhouse row. Since distinct plant tasks also need to be performed at distinct heights, the robot arms can perform the individual tasks mostly independent from each other.

From Figure 1.1 it can be seen that there is only a small but high strip of space between the plant rows where leaves do not grow. If there are many wires running along the 3.5 meter long pole, these wires could get stuck in the plants, which could destroy the robot or the plants. To prevent this, the number of wires running between arms and the base of the robot should be kept to the absolute minimum. To achieve this, an early design decision was made to keep real-time control on the robot arm. For the prototype, a power and Ethernet cable are still running to the robot arm, but in a later version the networking could be replaced with a wireless connection and the power could be delivered to the robot arms using sliding contacts on the pole. In this scenario, using fully centralised control could be negatively impacted by random WiFi latency and is therefore undesirable.

In order for the robot arms to interact with the plants, tools need to be placed on the arms. These tools are commonly referred to as end-effectors. The design of these tools will also impact the effectiveness of the robot arms at each task, so a separate study into the design of one or several of these end-effectors should be conducted to fully automate the greenhouse work.

The aim of the project is to design and control a robot arm that can be used to automate many, if not all, of the tasks that are performed regularly in a greenhouse.

The problem statement addressed in this thesis is as follows:

> How to design a robot arm that can automate tomato plant maintenance in a greenhouse?

With the following requirements:

1. Ability to automate multiple, that is 4, tasks.
2. The cost per arm should be kept under 5,000 euros.
3. Multiple arms should be able to work together on a difficult task.
4. The path planning and signal processing should be done on the arm itself.
5. The robot should have a high success rate, 99.9%, at critical tasks where the plant could be destroyed such as growth guidance.
6. Planning of a path should be finished within 10 milliseconds.
7. The tip of the arm should be position accurate within 1cm.

1.3.1 Growth guidance automation

To determine the effectiveness of the design of an arm, tests will be conducted with growth guidance. Growth guidance was chosen as a task because it is something no company or research group has publicly said to be working on.

The guidance is not done with a string, as is done in most Dutch greenhouses, but with 2 clips at the top, since this is easier to automate. The so-called clippers are shifted upwards across the plant, such that the plant top will be at approximately the same height at all times. The mechanical design of the end-effector can be seen in Figure 1.2. It also shows the camera that will be used to localise the clippers. The end-effector that will grab these clippers is outside the scope of the project as it was designed by a colleague, however automating the gripper will be discussed.

The end-effector will have some tolerance when grabbing a clipper. The robot arm should be able to place the end-effector at the required position within its 1cm tolerance as stated by requirement 7.

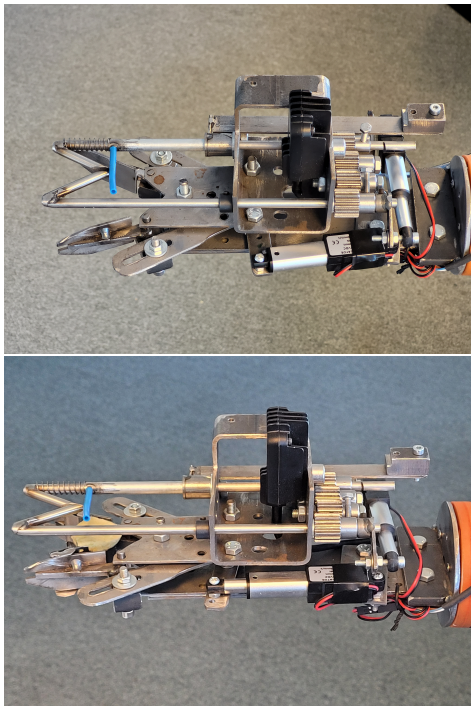

Figure 1.2: **Picture of the growth guidance end-effector (top) and while holding a clipper (bottom). Linear actuators were added to drive the claw and the foldable guidance system that is located above the clipper on the left.**

Chapter 2

Related work

In this chapter, some other works that can form a foundation for this project are listed. First, several projects in the horticulture industry will be discussed, after which software will be listed that can be useful for the project.

2.1 Related projects

As mentioned previously, the thought of automating greenhouse tasks has crossed many minds over the past 25 years. Some projects that stand out are reviewed below.

2.1.1 Farm Technology Group Wageningen University

In 2002, the Farm Technology group of the Wageningen University (WUR) finished a project trying to conceptualise a cucumber harvesting robot [24]. During this project, a robot arm and platform that could drive on the heating pipe rails were used that were both store-bought. The robot used cameras to estimate the size of each cucumber in order the select which ones to harvest. At this time an off-the-shelf robot arm was used to move towards the cucumbers and cut them. 95% of the cucumbers were detected with the machine vision system and 80% of the total was successfully harvested. As 95% is detected, we can derive that 15% is unsuccessfully harvested after detection and wasted. The remaining 5% that was not detected at all could be given another attempt at some other moment, but this would reduce the amount of hectares that a single robot can handle in continuous operation. In this scenario, the robot would still waste 15%, meaning the robot will need to be cheap and fast in order to be economically comparable to human workers. Having a very high success rate whilst still being relatively fast and cheap will, in general, help the viability and is therefore of high concern for the project.

More recently, bell pepper harvesting was attempted within the WUR group. There were two consecutive projects, CROPS [4] (Clever Robots for Crops) from 2010 to 2014 and SWEEPER [2] from 2014 to 2018. The first project had a more general scope within the greenhouse space, but for the second project the focus was shifted towards a single fruit, namely bell peppers. Figure 2.1 shows the robot. The measured success rate for harvesting the bell peppers was

Figure 2.1: **The robot produced in the SWEEPER project. Image credit to the Farm Technology group.**

61%, which is a lot lower than the success rate as stated in requirement 5 in Chapter 1.

The 61% is also significantly lower than the result the group had 16 years earlier with cucumber harvesting, which was 80%. This can be explained when looking at the way the bell peppers grow, as seen in Figure 2.3. Bell pepper plants grow vertically and cannot be bent the way tomato- or cucumber plants can, so the bottoms of the bell pepper plants are not horizontal. Thus, there is no open space around the bell peppers that allows them to be harvested more easily as is the case with the cucumbers, which are shown in Figure 2.2.

In all three projects we see the use of an articulated robot arm (see Figure 2.1) with a large control box attached to it, which is a good idea for a proof-of concept project, but will be inefficient to use on a commercial scale. Path computation will be demanding, especially for the articulated robot arms, as will be discussed in Section 2.2.2 and some movements are difficult to achieve. For example, flipping the 'end-effector' upside down might require a sequence of movements for all joints in the arm, which is then further complicated by all the plants that are in the way. As the aim of the project is to build a robot with multiple arms, the control box seen on the robot in Figure 2.3 is much larger than can feasibly be put on a robot multiple times for multiple arms, meaning a smaller more efficient system should be chosen or developed for this system.

2.1.2 Leaf cutting robot

The hortimotive company Priva is focused on smart climate and water control. It recently announced that is has been working on a leaf cutting robot for quite some time. The robot, called Kompano [18], was announced to be ready for deployment, but not much further news about robots being sold to end-customers has come to light. The robot also uses a cart that can traverse the heating tubes to get around and uses a stereo camera to determine where to cut. As it has a flash next to the camera module, the robot can work through the night. The robot also has a UV-light decontamination system to lower the change that the robot will spread viruses. The robot can do up to one hectare of greenhouse per week with an accuracy of 85%. Whether in the case of the other

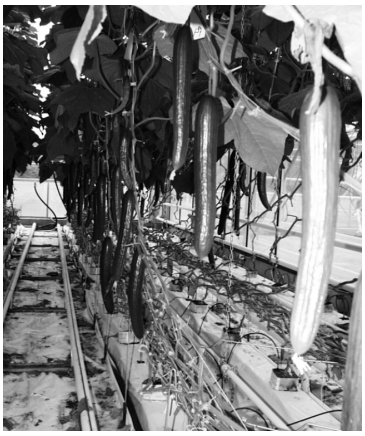

Figure 2.2: **Cucumbers hanging below the stem [24].**

Figure 2.3: **Bell peppers alongside a vertical stem [2].**

15% the plant is damaged or the leaf is left on the plant is not specified. With this robot, price is the main concern. Growers in 2020 had on average almost 7 hectares of greenhouse [1] and thus also need 7 Kompano robots, making it a very large investment. Even so, since the robot only cuts leaves, this investment will only decrease the labour volume by a portion.

2.1.3 Panasonic

Another robot that can pick fruits was announced by Panasonic in 2018 [15]. Its claimed to be as fast at picking tomatoes as a human. This means that their target is to pick 10 tomatoes per minute. To achieve this, the robot arm can move rather quickly. It has a machine learning algorithm to select tomatoes that are ripe enough for harvesting. Panasonic's robot has, however, supposedly only been deployed on test farms. In the article, they also discuss how in Japan the workforce is both decreasing and rapidly ageing, meaning the pressure of greenhouse automation is even higher in this country. From the fact that the robot is not sold on large commercial scale even with the large staff shortage we can conclude that the tests were not as successful as expected.

2.1.4 Plantalyzer

Wageningen University in cooperation with several horticulture companies has created an autonomous vehicle that collects data about fruits. This cart drives between the plants autonomously. Using multiple cameras it counts tomatoes and determines their ripeness. This data is used to constantly predict yield, which can be used to accurately determine how much produce can be sold each day, a few weeks ahead of time. The autonomous vehicle is shown in Figure 2.4.

Crop forecasting is a useful tool that can help reduce produce going to waste and also gives the farmer a change to collect a higher price for their fruits. However, both growers and researchers agree that the Plantalyzer and other data gathering robots are unlikely to be deployed at a large scale if no active work is performed by the robots. Especially in Holland, the growers do not see

Figure 2.4: **The Plantalyzer vehicle. Image credit to Hortikey.**

the direct benefit of the precise yields to compare against the large costs and IT infrastructure associated with these types of systems, as they think they are good enough in predicting the upcoming yields themselves.

2.2 Software

Section 2.1 showed some related projects that were worked on over the last 25 years. A first note is that current robots all specialise in a single task. This task is most often picking the fruit since this is a larger task compared to the others. However, from an efficiency standpoint, having a robot that does multiple tasks in a single pass is better. This would also prevent the various robots from driving into each other if both want to go into the same path which also supports requirement 1 along the monetary reason provided in Section 1.3.

Below we discuss some software tools that can be used to help build the project.

2.2.1 ROS2

ROS [12] stands for Robot Operating System and it is a platform that aims to speed up robotisation, which has made it widely used in the robotics industry. ROS provides a set of algorithms and software packages, as well as a inter-process communication layer. The vast majority of these software packages are open source. The operating system part of the ROS name is not a super accurate description, as ROS is a middle ware application that runs on top of an OS, usually Ubuntu Linux, instead of an OS itself. Since Ubuntu can run on both large desktop computers and small single board computers like the Raspberry Pi, it can be used to make a heterogeneous, networked system which can communicate.

In the ROS ecosystem, robot control is split up into execution units called nodes. These nodes usually get their own system thread but can also share a thread. The nodes can communicate with each other using services or with a publish-subscribe protocol over channels, which are called topics. ROS2 is an

iteration of the original ROS, where the network communication was simplified. With ROS2, running two nodes on two different systems on the same network can be accomplished with less configuration work compared to ROS.

The ROS2 system was used as an abstraction layer on top of the networking that needs to take place between the robot arm discussed in this thesis and a central hub. The communication enabled by ROS2 will handle both the division of the work between robot arms and keeps track of the work that was performed. Manual control of the robot arm using a game controller is also connected to the robot arm from a laptop using ROS2.

Another advantage of the node system is the ease with which parts of the software stack can be moved around. Even though the finished software is light-weight to run, compiling the C++ code takes a significant amount of time, slowing down the development and testing process. Moving the path planning onto a fast laptop whilst keeping the communication with the joints on the arm's smaller processor helps to keep the compilation time low. Afterwards, all software can be easily moved into place and the functionality will be exactly the same.

Lastly, ROS2 provides the TF system [8], which is a framework that can listen to messages that specify the relations of coordinate frames with respect to each other and builds a tree from this specifying what frame is a child of what other frame. Using this information, the TF system allows the user to request a transform from any frame to any other frame, such that, for example, the location of an object with respect to the end point of a robot arm can be converted to the location and orientation of the object in the world frame. Then, when we want to move to the object, the position of the object in the world frame can sent to the robot arm, who will transform it to its own frame at that time, even if the robot has moved.

2.2.2 MoveIt motion planning

To control a robot arm, a software library is required. First of, the motors at the joints need to be able to be controlled to to specific angles. This would involve communicating with motor drivers. The vector of all the angles of the motors is called the state. This state can also be mapped to the end-effector location in the robot space, which is then called the pose. The process of using the geometry of the robot parts and angles of the joints to calculate the end position is called forward kinematics.

To do the reverse process is called inverse kinematics. Inverse kinematics are useful when the arm needs to be sent to a specific pose and we want to calculate the state that accomplishes this pose. However, this process is a lot more difficult, as there can be multiple states that achieve the end pose and a large number of ways to move to this state. MoveIt [17] is an open source software package that standardises and implements the core parts of robot control. It is also built on and well connected with the ROS(2) middleware. The inverse kinematics are implemented using a probabilistic solver to get solutions in a short time for up to seven degrees-of-freedom articulated robot arms, meaning a simpler robot arm would even quicker to solve.

It makes sense to use this package for the development of a robotic arm and the package was tested with the robot arm and the electronics, but ended up not suiting the needs of the project, as will be further discussed in the design

Figure 2.5: **FANUC SR-6iA SCARA robot arm. Image credit to FANUC.**

chapter.

2.2.3 Off-the-shelf SCARA arm

SCARA stands for Selective Compliance Articulated Robot Arm and these types of robot arms are used in assembly lines. The advantage of SCARA arms is that the arm requires less computation to control compared to an articulated robot arm. An example of one such company is FANUC. Fully operational SCARA arms can be bought from several companies. They have many options available and the control software is already included. This means that the arm can be programmed at a high level and the kinematics and motor control is functional out of the box. Figure 2.5 shows a picture of the FANUC SR-6iA, which is a model that has an arm length of 650mm, which is the right length for a tomato greenhouse since it will be long enough to reach between the plants, while more length causes will make it more difficult to manoeuvre the arm without touching the surrounding crops.

However, the software is closed-source which is less desirable and the physical design of the arm is more suited to a flat work area where the robot can move over the area, such as an assembly line. In a greenhouse, the plants are approached from the side and the robot arm should be able to move in between the branches and leaves, so the robot arm should be as flat as possible, whereas these arms are designed to move the end-effector vertically, which gives them more height at the tip of the arm. For this reason together with their price, the FANUC SCARA arms are not suitable for the automation of greenhouse tasks. In the design section a SCARA arm is discussed that is suitable for the greenhouse.

2.3 Discussion

The projects that were listed fall behind in their success rate, where 80% is at the highest side of the spectrum. From a sustainability perspective this is undesirable and it also means that growers will have less incentive to adopt the technology. We can see that in general, none of the robots are sold on a large scale.

The use of off-the-shelf robot arms does not provide the flexibility that is required in a greenhouse. Most SCARA arms are designed with a large vertical actuator at the end, which will interfere with the leaves, whilst redundant articulated robot arms, will have trouble turning in the small area between the plants. It will also require a lot of processing power, which conflicts with requirements 4 and 6.

Developing a flat robot arm that can manoeuvre in the gaps between leaves can lead to an economically viable solution that can really alleviate a significant amount of the repetitive work in the greenhouse sector.

Chapter 3

Design

The design of the robot arm consists of two aspects, hardware and software design. The hardware design describes the physical components of the system and why they were chosen. The software design discusses the software which is needed to create a functional robot arm and subsequently automate the growth guidance. First, the hardware design will be discussed followed by several sections discussing different components of the software. Lastly, a rough estimate of the cost of the robot arm will be considered.

3.1 Hardware Design

This thesis is focused on the arms of the robot, but as previously discussed, the design of the arms can only be operational if it is coupled to a full system that can be deployed in a greenhouse. In the proposed greenhouse automation system, there will be a cart driving on the heating pipes with a vertical pole that has arms on it. The cart houses the batteries that will power both the under-carriage wheels and the arms. The cart will also house a computer that acts as a central hub. Figure 3.1 shown the system in a schematic.

The central hub computer could also handle path planning, however, this would obstruct the scalability that is desired. Ideally, in the final product, an arm can handle its own closed loop control and communicate with the central controller only for deciding which task to do next. Having a central controller can increase efficiency and this controller can keep track of the tasks that are in range of all the robot arms. If all tasks are completed, the central controller is responsible for moving the cart along the rail. The efficiency can be increased with the central controller making sure the arms perform tasks in such an order that the arms do not interfere with each other.

Each arm is outfitted with a single board computer capable of running Linux and ROS2 2.2.1. The single board that was chosen for the development phase is a Raspberry Pi [19] since its easy to get up and running and has wide support. In general, using single board computers gives both excellent performance per Watt and price to performance ratio. As seen in the schematic, a camera is also attached to the arm and connected to the Raspberry Pi to create an in-hand camera view, which means the camera is behind the end-effector as can be seen in Figure 1.2. The camera will be used to detect the plants, clippers and fruits

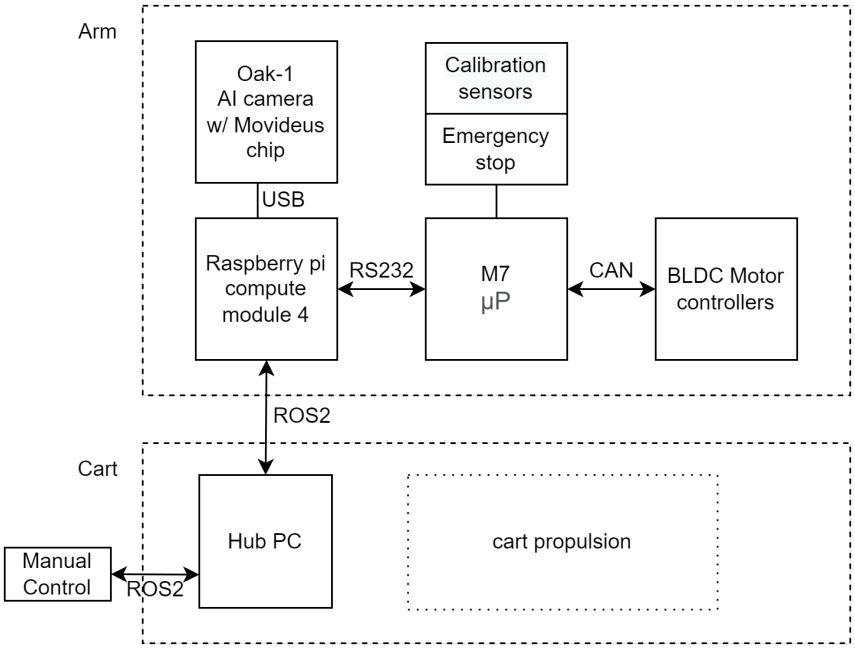

Figure 3.1: **Layout of the electronics and communication of the designed system.**

in related sub-projects of the greenhouse automation project.

The motor controllers that were selected all support control and feedback over a CAN bus. CAN busses are serial busses originally developed for the automotive industry that do not need a central controller. This means that a single CAN bus runs across the arm connecting all joints to a processor. An M7 microprocessor is added in between the Raspberry Pi and motor controllers to consolidate the communication. The M7 is also used to handle the emergency stop button that disables the motors, causing them to stop in place. As the Raspberry Pi boards are not capable of driving the CAN bus directly, this approach also saves on an expensive USB to CAN adaptor since the microprocessor has the CAN bus connection.

For the communication between the M7 and Linux board, a serial connection is used. This is again because this omits the need for a USB to CAN adaptor. Also, it means that there is a full duplex link available for asynchronous communication between the processors. Both devices have a hardware serial port that is directly connected to the CPU, these ports were used for extra stability.

To allow for an arm that is simple from a mechanical standpoint, 'pancake' motors were chosen for the joints. These motors are flat motors with a built-in planetary reducer and form the connection between parts of the arm without extra components like bearings and gears. There are multiple brands that make these types of motors, but none of the motors we ordered as products seemed to be fully finished, with features of the products being added or changing from month to month. One important reason for choosing a motor was that it should be able to move at both fast and slow speed, to allow for the camera to capture

different sides of the plants. However, two of the three brands that were tested could not follow a trajectory at slow speeds with high accuracy and thus could not be used for the project and Igus Rebel joints were chosen in the end. The disadvantage of these motors is that no parameters such as position PID can be configured with the API that was supplied by Igus meaning only default parameters were used during the project.

3.2 Robot arm control

Controlling the robot arm can be seen as a two-part system, a lower and higher part. Setting a robot state using the motors and feeding actual angles back to the ROS system is the low-level part of the system and will be discussed in Section 3.2.1. At a higher level, these robot states need to be generated based on some specified end location or trajectory. The remaining sections will focus on this high-level control.

3.2.1 Joint control

In the Hardware design section, the electrical layout of the motor control was described. In this section an overview of the software aspect of the joint control stack is given.

To help with simple control of the joints, a PlayStation controller was used to control each joint independently. This way the motors could be rotated and be given a zero-angle matching the zero angle in the software. The zero angles for each joint are defined as having the arm stretched forward. Later on, a calibration sequence was implemented to easily and more precisely set the zero angle of the joints using induction sensors. Setting the zero-angle is required every time the robot turns on, as all motors have incremental encoders and consequently lose their absolute angle when losing power.

Each motor is named after the joint in the human arm, where the wrist is split in two and the height motor is called the slider joint. There is a vertical wrist joint that rotates around the vertical axis and a horizontal one.

To visualise the information that the joints give back, a GUI is a useful addition to the software package that controls the arm. The ROS ecosystem (Section 2.2.1) has a tool called RVIZ that functions to visualise autonomous systems and their surroundings. There are many plugins available that allow for visualisation of different systems, some of which are part of the MoveIt system and allow for easy visualisation of the robot arms.

A custom plugin was also written. The plugin creates a so-called panel, which allows for a sidebar to be visible in RVIZ. The panel has a few buttons that can be used to enable and disable motors, as well as calibrating them. There is also a file-tree widget included, which creates a collapsible information system for all the joints in all arms. Motor errors, angles and motor currents can all be easily looked into.

3.2.2 Inverse kinematics

To implement arm control in Cartesian coordinates, software is required as discussed in Section 2.2.2. MoveIt! seemed like a good candidate for the job

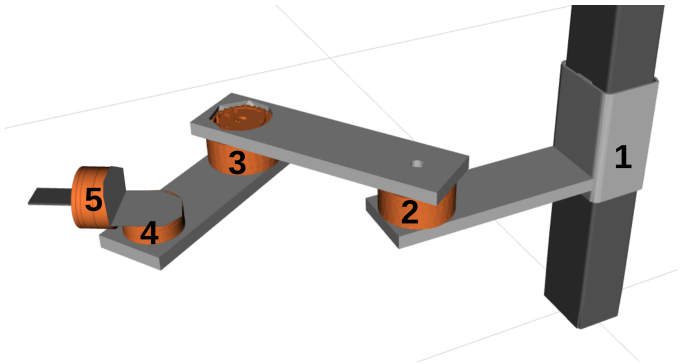

Figure 3.2: **Visual representation of the model that is provided to MoveIt for path planning and collision checking of a robot arm. Actuated joints are numbered.**

since it is well integrated with ROS2. The package has visualisation, a GUI and so on to help planning. Tutorials can be found on [16]. These tutorials help to get MoveIt up and running for a simulated robot, specifically an articulated robot arm. There are many configuration files included in the tutorial that define the robot and its motion characteristics.

To make the SCARA robot arm move, MoveIt was configured with joints and motion limits that were compatible with the configuration that was designed for a greenhouse. In Figure 3.2 a screenshot is shown of the model that is provided to the MoveIt package. The model uses the URDF [13] format to specify the geometry of the robot. Two sets of geometry meshes can be added for each component of the robot, allowing the collision checker to use a heavily simplified mesh, while the user sees a detailed mesh. The under-carriage is left out since its only relevant for autonomous driving of the vehicle. Getting MoveIt to work with the real robot arm proved to be more difficult and impractical than expected for several reasons. Firstly, in all tutorials the robot was only simulated. This meant that position setpoints were fed back directly into the visualisation portion of MoveIt and only the planning was demonstrated and tested. In order to control a real robot arm, the arm state setpoints needed to be picked up by a ROS node, which would pass these along to the motors. Then, the actual angle of each motor should be passed back to the visualisation tool to overcome this complication and the arm could be moved to coordinates using MoveIt.

Another observed problem was the amount of processor resources used by the system. Tasks like planning and requesting to start following a plan took in the order of seconds to execute and the laptop running the software was mostly utilised with running a single arm. This was still on a laptop, meaning current single board computers will not be able to run MoveIt, especially if camera processing is added. Thus, the control of all arms would need to be performed from a single location. Then, the decentralised modular design is lost.

This leads into the last issue, where MoveIt is only designed to control a single 'planning group' at a time. The planning groups are a set of joints that are actuated simultaneously, so each arm should be its own planning group to allow different arms to move to different locations. So, to move robot arms

simultaneously and independently, multiple isolated MoveIt systems need to run on the robot. This would require a laptop or desktop class processor to be fitted to each robot arm, which is undesirable due to space and budget constraints.

To conclude, the MoveIt system is a powerful system with lots of interesting and useful tools, but unfortunately its more suited for larger, single arm robots. For the project, the visualisation part of MoveIt can still be used to verify the robot state and a planned robot trajectory, but a tailer-made solution is required for the planner and real-time controller.

3.2.3 Trigonometric approach to kinematics

In this section the tailor-made kinematics solution is presented. The kinematics describe the relationship between a pose in $x, y, z, \theta, \psi, \phi$. The inverse kinematics are derived first followed by the forward kinematics. Some implementation details are left out of the equations to keep compact.

Inverse kinematics

Due to the SCARA arm design, many degrees-of-freedom are decoupled from each other and directly correlate to a single motor, meaning the inverse kinematics are simplified. For instance, the z-axis is controlled solely by the slider motor, labelled **1** in Figure 3.2, on the pole and there is a motor at the end of the arm that controls the roll ψ independently of all other axes labelled **5**. The pitch ϕ can not be controlled in the current layout, but a motor could be added to the end of the arm to achieve this, if this would ever be necessary. The roll ψ and z axes can be controlled by controlling either motor **5** or **1** respectively and therefore disappear from the inverse kinematics equations.

The remaining three motors rotate around the vertical axis. These motors are responsible for the x, y and yaw control, but the motors are together controlling the end-effector position in these three degrees of freedom. Figure 3.3 shows a schematic overview of the arm on the robot cart in a greenhouse row. To solve for the motor angles, we start with the position of the wrist.

With a desired position X, the wrist position W can be calculated as the point at distance D, the length of the hand, in the inverse direction of the desired rotation of the hand, the yaw θ. From position W, given by Equation (3.2) and the fixed shoulder position S, length C can be calculated using the Pythagorean theorem [23], see Equation (3.1). Now, all three sides of the imaginary triangle ABC are known, so the cosine law [10] can be used to calculate the angles required to form this triangle, where we are interested in γ and α, given by Equations (3.3) and (3.4) respectively. However, it should be noted that two mirrored triangles exist and determining a single solution will be discussed later. δ represents the angle between the x-axis and the wrist position as seen from S and is expressed as Equation (3.5).

$$C = \sqrt{A^2 + B^2}. \tag{3.1}$$

$$W(x, y) = (X(x) - D\cos\theta, X(y) - D\sin\theta). \tag{3.2}$$

$$\cos\gamma = \frac{A^2 + B^2 - C^2}{2AB}. \tag{3.3}$$

$$\cos\alpha = \frac{C^2 + B^2 - A^2}{2CB}. \tag{3.4}$$

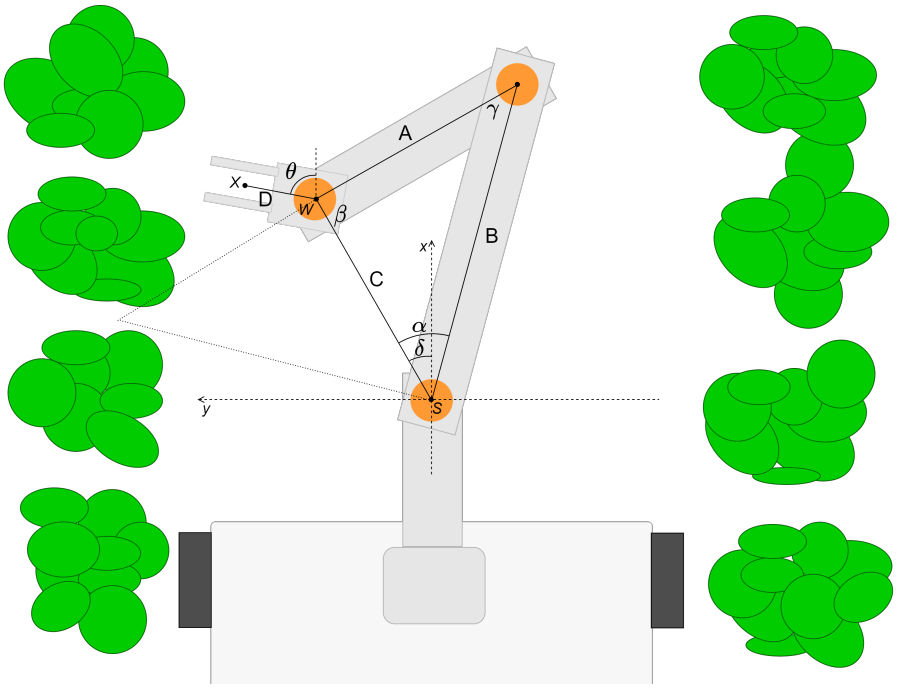

Figure 3.3: **Schematic top view of a robot arm in a greenhouse row, illustrating the triangle that can be used to calculate the joint angles.**

$$\tan\delta = W(y)/W(x). \tag{3.5}$$

The shoulder angle which controls motor **2** is now described by Equation (3.6) and the elbow angle by Equation (3.7). It should be noted here that the angle of the elbow motor, labelled **3** is inverted since the motor is upside down. For all motors, counterclockwise rotation always corresponds to a positive angle to follow the right hand rule.

$$M2 = \delta \pm \alpha = \arctan(W(y)/W(x)) \pm \arccos\frac{C^2 + B^2 - A^2}{2CB}. \tag{3.6}$$

$$M3 = \mp(\pi - \gamma) = \mp(\pi - \arccos\frac{C^2 + B^2 - A^2}{2CB}). \tag{3.7}$$

Since there are two options for the location of the elbow joint, as denoted by the $\mp$ in the equations for $M2$ and $M3$ one still needs to be chosen to return a definitive robot state. However, as the robot will drive up and down a greenhouse row, it can perform its tasks on one side of the row during both times. In Figure 3.3, where the robot is moving towards plants on the left side, the left triangle drawn out with a striped line is undesirable since in this configuration the robot arm will interfere with a plant. To minimise the elbow joint touching the plants, the middle of the arm will always be facing away from the plants, thereby predetermining which equation is rejected. Now, a simple algebraic solution to the inverse kinematics is found, which can be quickly calculated, even if it needs to be done many times when planning a trajectory.

Forward kinematics

To calculate the forward kinematics, the contributions of all arm segments to x, y and theta need to be calculated. This is done using the sine and cosine, however, the angles of previous joints add up to the angle of the next element. Therefore, θ is described by Equation (3.8). It should be noted again that the elbow angle is inverted. The x and y location of the end point X are described by Equations (3.9) and (3.10).

$$\theta = M2 - M3 + M4. \tag{3.8}$$

$$x = \cos(M2) * C + \cos(M2 - M3) * C + cos(\theta) * D. \tag{3.9}$$

$$y = \sin(M2) * C + \sin(M2 - M3) * C + \sin(\theta) * D. \tag{3.10}$$

3.2.4 Collision checking

After implementing the inverse and forward kinematics, an important part of MoveIt still needed to be simplified, collision checking, which is implemented with algorithms such as [20]. The collision checking of MoveIt is implemented in 3D space, but this is not required for the robot arm. Different robot arms cannot reach each other vertically, if the correct distance is maintained along the pole using the sliding joints. However, the different segments of the arm can hit each other, so checking when these collisions will occur should also be implemented. However, the location and orientation of these segments do not linearly correspond to the angles. The advantage is that this problem is again in 2D and can be solved using rectangle intersection tests. The rectangles that correspond to segments of the arm are visualised in Figure 3.4. The blue segments are checked against each other as are the green segments, while green segments are not checked against blue segments and vise versa, since the height difference between the segments prevents collision in general.

3.2.5 Path smoothing

To move the robot arm from one pose to another using the inverse kinematics, a path is created. For simplicity, only linear paths are created, where a more complex path will require multiple way points to be passed to the path planner. Linear paths are not the most efficient in energy usage, since a joint, for the current design especially the elbow, can move clockwise and then counterclockwise in the same path, wasting energy that could have been preserved if a curved path was chosen. Multiple optimisation methods were studied in literature [3][11][21]. In the greenhouse it is desired to move between the leaves without touching them, thus linear paths are preferred as they can predictably be laid out between plants. When the planner creates a curved motion between the plants, the next plant in the row could get swept down by the end of the robot arm.

An S-curve helps to avoid vibrations in the robot arm when it moves. This is accomplished by slowly speeding up and slowing down instead of moving with a constant velocity along the path. To further prevent jitter, the acceleration is also continuous. This is called a second order S-curve and it is a good trade-off between computation of the path itself and smoothness, as discussed in [7].

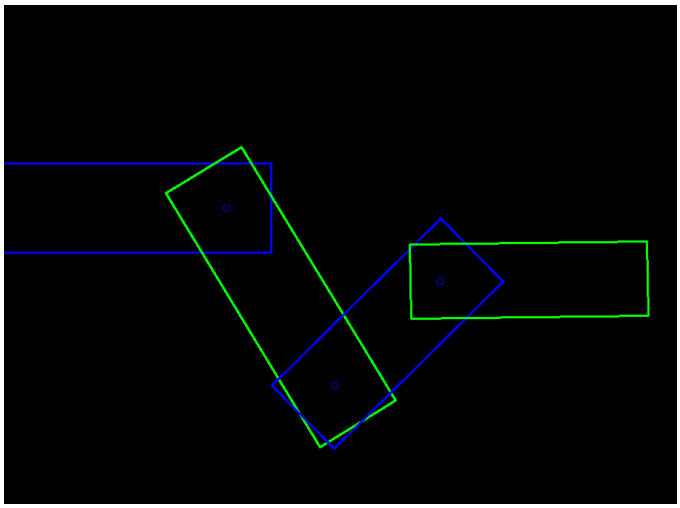

Figure 3.4: **Visualisation of the collision checking rectangles that represent each part of the arm. As different parts of the arm are at alternate heights, these parts cannot hit each other and only parts with matching height can, who therefore have a matching colour.**

Figure 3.5 shows an S-curve as generated by the algorithm. In this example, the curve is asymmetric since the start speed is 0.002 and the end speed 0.

3.2.6 Visual servoing

Visual servoing is the process of controlling a robot arm based on the current camera input. This means that the target position can be adjusted during a movement of the robot arm. The process works especially well with an in-hand camera, where the camera moves towards an object. If the target position can be more accurately measured when the camera is closer to the target, visual servoing can positively impact the accuracy of the system. A disadvantage of visual servoing is however that the path that the arm traces to the goal position is not computed in advance, so none of the smoothing optimisations can be performed on it.

In this project, there will be multiple cases where a clipper, a rod or in future work also a sucker need to be approached straight-on. When the object is approached, it is also easily visible in the centre of the camera view. If the location of the object can be extracted from the view, a control loop can keep the object in the centre of the view and therefore also keep the end-effector centred on the object.

3.3 Growth guidance implementation

For the implementation of the specific tomato greenhouse task of growth guidance as explained in Section 1.2.2, some extra software is required on top of the path planning and robot control. This software will be discussed here.

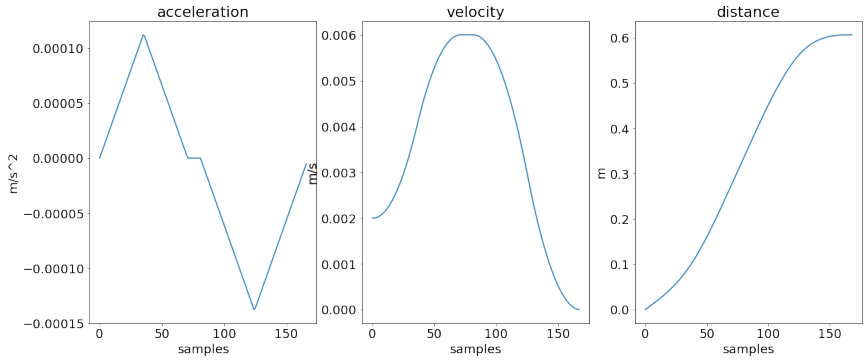

Figure 3.5: **S-curve and its first and second derivatives.**

3.3.1 Localisation

Since the camera is in-hand and the growth guidance end-effector should place and retrieve the clipper straight on, the accuracy can be improved using visual servoing. This requires the clipper rod to be detected in the image. Even though a complete vision pipeline is outside the scope of this thesis, a simple detection system is required to test the visual servoing against moving to fixed, programmed positions. To keep the machine vision algorithm simple, it is only made to work in a lab environment where there is nothing surrounding the rod. An estimate of the position of the clipper is provided beforehand, to create a starting point from where the arm can clearly see the rod and move to it. The detection of the rod in the image is used to steer the arm while moving to the clipper or rod, making the system more robust when the starting location provided by the external source is inaccurate as it will make adjustments when going to the location.

Clipper rod

Detecting the clipper rod from the image cannot be easily done in a single edge-detection algorithm due to the fact that the rod is made of metal and consequently reflects the light and colour of its surroundings. The plan was to use an edge detection algorithm implemented by the widely used open-source machine vision software library OpenCV [14] called the Hough transform [6]. However, the edge detection gives back many small segments detected instead of a full rod. Then, to merge the segments, a line fitting algorithm is used on the end points of the individual line segments. Figure 3.6 shows the line that is fitted after filtering. To filter the points, all horizontal line segments are removed from the set. For visualisation, these are drawn using green lines, which can be seen on the clipper end-effector. To further reduce erroneous measurements, an average of all measured points is taken, which is used to segment out a vertical strip of measured points in the image. Only these points are taken into consideration for the line fitting. To further prevent outliers from moving the fitted line away from the rod, the Huber cost function [22] was used for line fitting. The Huber function acts as the L2 or quadratic cost function at small distances and as the linear function above a threshold (C) to prevent outliers from having large costs, see Equation (3.11). Now, the blue centre point of the

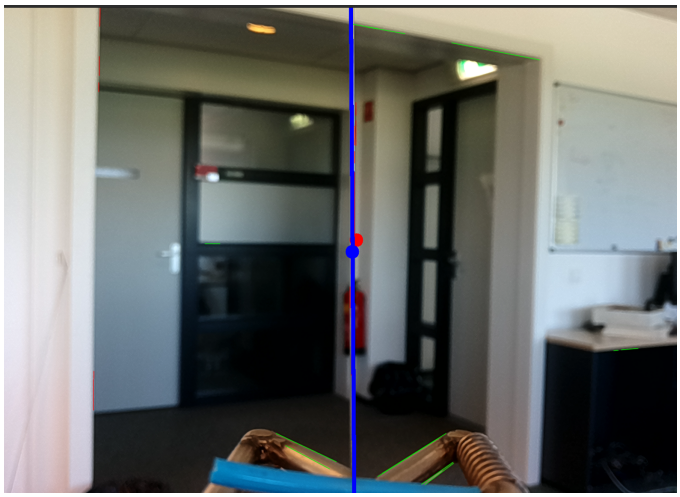

Figure 3.6: **Detection of the rod that is used to hang the plant off using clippers. Green and red are respectively rejected and accepted lines based on angle, the red dot is the mean point of all lines and the blue line is the finally fitted line.**

line can be used as the homing point for the visual servoing and the angle of the line can be used to rotate the clipper to match the rotation of the rod.

$$\rho(r) = \begin{cases} r^2/2 & \text{if } (r < C) \\ C \cdot (r - C/2) & \text{otherwise} \end{cases} \tag{3.11}$$

3.3.2 State Machine

Both placing and retrieving the clipper require a sequence of actions and for repeatability measurements of the system, placing and retrieving the clipper were connected to each other in a loop. To order these actions, a state machine was implemented, which is visualised in Figure 3.7.

The states are an alternation of arm movements and end-effector movements. When moving to the rod, with or without the clipper in hand, the arm can either be steered to a fixed pose that was saved ahead of time or be controlled using visual servoing. In the case of the visual servoing, the saved position is used to determine up to which depth the arm should move forward. This depth information cannot be determined from the 2D image.

3.4 Cost

The cost of the arm has been estimated in Table 3.1. The material costs are a rough estimate and assembly of the robot arm is not included. When building many robot arms the cost of the materials and parts will go down and the total cost can be assumed to be comparable with the 5,000 euros target (requirement 2). This means that cost-wise, the design is feasible.

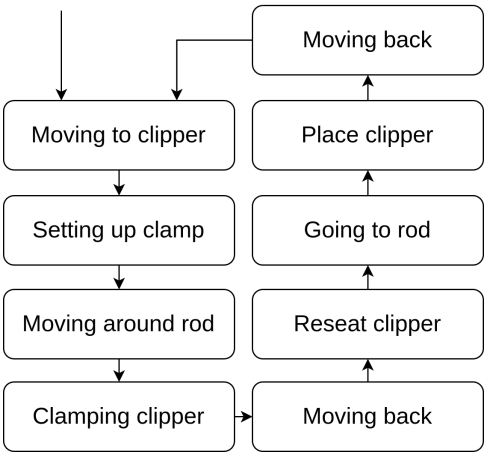

Figure 3.7: **State machine used to control the robot arm and clipper end-effector.**

Description	Price
Motors	€ 500 * 5
Raspberry Pi	€ 100
Camera	€ 200
Matarials for the arm	€ 800
End effector with motors	€ 100
Total	€3,700

Table 3.1: **Table with estimated prices for all components of a robot arm.**

Chapter 4

Testing and Results

To study the validity of the robot arm arm design, several tests were conducted on a prototype robot arm. These tests together with their results are outlined below. General tests on the performance of the arm will be discussed first, after which growth guidance tests will be examined.

4.1 General tests

As the robot arm design is aimed to not only automate the growth guidance, but also the other tasks in the greenhouse, general performance is also measured. It is also a validation of the arm control software, which should be able to move the end-effector to any pose using a smooth motion.

4.1.1 Positional accuracy

An important aspect of a robot arm is its ability to accurately track the trajectory that has been planned. To test this, the robot arm is randomly moving as described in Section 4.1.2 during which time the set-point and actual angles are recorded. An example segment of this recording is graphed at the top of Figure 4.1. Through subtracting the target angles generated by the planner from the angles measured by the encoders in the arm, a tracking error can be calculated. This tracking error is shown in the bottom half of Figure 4.1. Grey vertical lines are added at the randomly generated set points to indicate the moments where the arm is supposed to stand still.

The error compared to the target angles that were sent to the motors can be up to three degrees. This means that in the case of the shoulder motor, for whom a maximum of two degrees was observed, this error could lead to a displacement of 2.8cm, see Equation (4.1). Thus, to come closer to the targeted 1cm accuracy, the motors should be tuned to more aggressively follow the target position. That being said, with a slower speed the error decreases and the accuracy requirement of 1 centimetre is met at the target positions where grey lines are drawn in the figure. Thus, the s-curve helps to let the joints catch up to the targets, allowing them to match a end point more accurately.

$$80cm * \sin(2^\circ/180 * \pi) = 2.8cm. \tag{4.1}$$

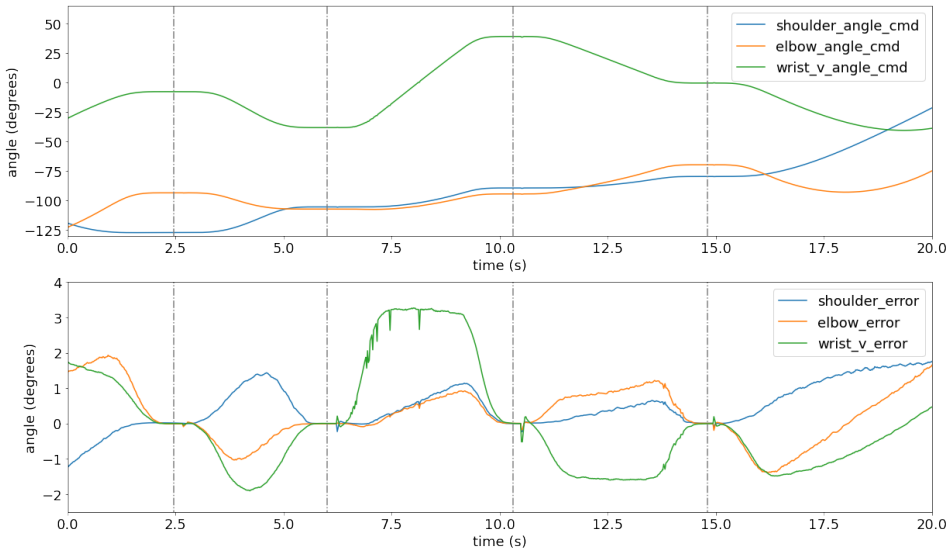

Figure 4.1: **Top: Absolute angle set-point sent to each joint when moving to several random locations in a row. Bottom: Tracking error of the joints relative to the set-points.**

4.1.2 Arm control planning measurements

Figure 4.2 shows the execution time of the planning algorithm for path to random points. Since the arm was sent from a random point to another random point in this test, measurements of over 200 different paths with different lengths were collected which allowed for performance profiling in a wide range of motions. This profiling is not possible during the clipper test because it only has 2 points to plan between. These timing measurements were performed on a Raspberry Pi CM4 running Ubuntu 20.04 in the performance power profile. With the default power profile, the processor appears to switch between lower and higher power states, thereby making the results far less consistent.

We can see from the figure that some setup time of around 2ms is required, after which the execution time grows linearly with the path length since every point needs to be checked and the number of points also grows linearly. These results only show successfully planned paths, meaning that when a collision is reported, the measured time of the path is discarded.

4.2 Growth guidance automation tests

Here the effectiveness of the robot arm at growth guidance automation is discussed.

4.2.1 Clipper repeatability tests

Multiple tests were conducted with the clipper state machine running in a loop as discussed in Section 3.3.1. Different experiments were done to check what works better and what the system can handle. Creating a score out of 0 to 100

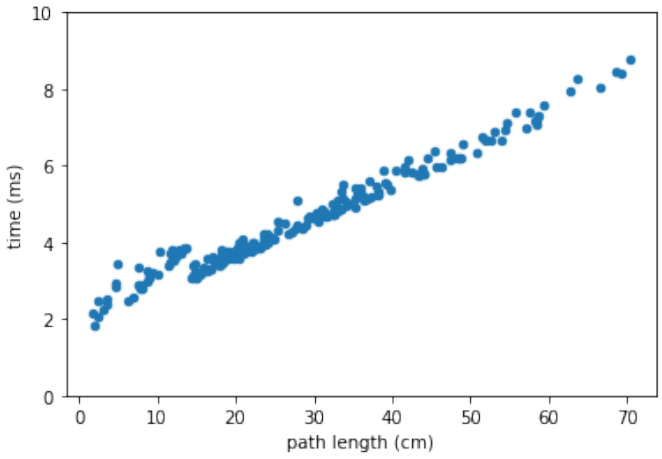

Figure 4.2: **Plot of the planning time of a path versus its length.**

percent cannot be done directly, since the clipper will fall or get stuck when the system fails, making future loops invalid. As an alternative, multiple attempts are performed and when a loop fails, the number of successful loops prior is recorded. Table 4.1 shows the results of these recordings for different scenarios that will be listed below, where first the rod tracking will be disabled after which these results will be compared to tracking results.

Fixed rod, fixed location. The rod is fixed in place at the bottom and the exact position was measured using the arm itself. In this case, the test loop could be ran over a hundred times in a row without failing, so only a single attempt was recorded.

Dangling rod, fixed location. The rod is hung from the ceiling with a few kilograms of weight attached to it, to more accurately simulate the greenhouse scenario where the clipper rods hang from a wire attached to the roof and a plant weighs the rod down. In this scenario the rod tracking is disabled. As any disturbance to the rod causes the mass to start swinging, during the second loop the gripper completely missed the rod and got stuck, ripping the rod and mass of the ceiling. For this reason, no further attempts were taken in this scenario.

Fixed rod, tracking enabled. The rod is fixed in place at the bottom and the position was measured. In this scenario the rod tracking was enabled which allows for a less accurate measurement of the rod location to still achieve reasonable results. The start location has 3 centimetres of tolerance in the direction perpendicular to the camera and only 1cm in the direction parallel to the camera.

Dangling rod, tracking enabled. The rod is hung from the ceiling with a few kilograms of weight attached to it. Here, any swinging motions of the rod are matched as closely as possible by the robot arm to increase the chance of a successful loop.

With the results listed in Table 4.1 a rough success rate can be estimated, which are shown in the right-most column. Due to the fact that the second experiment could not be safely tested more than a few times, 0% is listed in the table to indicate that the combination is not viable.

Scenario	Attempt	Successful cycles	Success rate
Fixed rod, fixed location	1	100+	100%
Dangling rod, fixed location	1	1	0%
Fixed rod, tracking enabled	1	12	95%
	2	14	
	3	7	
	4	50	
Dangling rod, tracking enabled	1	21	95%
	2	20	
	3	21	
	4	28	

Table 4.1: **Number of successful clipper placing and retrieving cycles, as well as the estimated success rates. Scenarios correspond to the descriptions in Section 4.2.1**

Looking at these results we can see that the system works best with a fixed rod location and without the rod tracking. This is because the rod tracking can slightly shift the arm sideways even when this is incorrect. When this happens, the clipper can get stuck behind the rod when the robot tries to take the clipper, causing the rod to start oscillating. This will make it more difficult to correctly track the rod and therefore the system fails during the next clipper placement. That being said, to get the position measured exactly to start of the fixed position loop took a few tries, whereas the adaptive tracking system required less precision. With a rod hanging from the ceiling, enabling or disabling the rod tracking creates a much larger split in the results. The fixed position is no longer usable since the rod can easily swing side to side a lot more than the guide can compensate for, which is only 1 centimetre while the rod can move 10 centimetres. Therefore, the arm has to adapt to the movements of the rod just to have more than one successful iteration in a row.

If the clipper is either not fully pressed in or the clamping mechanism sticks to the rod because the hooks get stuck, the rod will be pulled along. This can either cause a problem immediately if it does not let go, but in most cases the rod slips away and starts swinging. This means that during the next loop, the clipper can miss the rod as the rod is now swung backwards.

If the camera is shaken due to a small frantic movement, the rod detection becomes less reliable. The output of the detection is not filtered and directly sent to the PID controller of the rod tracking. This control loop amplifies the effect in the opposite direction to compensate for the movement but settles down after. If a shake happens just as the arm approaches the clipper, the arm hits the clipper and pushes it away. When it clamps down afterwards it will miss the clipper.

One in twenty tomato plants being dropped on the floor will not be be nearly good enough for the grower, which is why one in a thousand was specified in the requirements. So, a lot of work is still ahead.

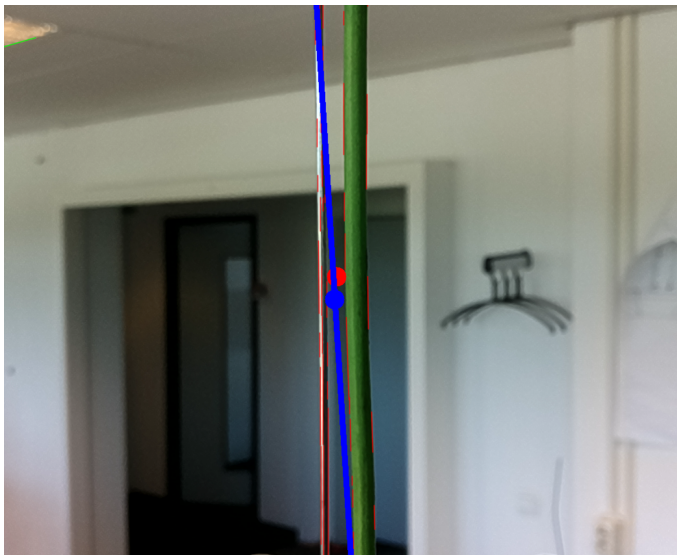

Figure 4.3: **Rod detection with a fake plant attached to the rod.**

4.2.2 Rod detection with a plant

The clipper rod detection system worked well for the repeatability test, but Figure 4.3 shows a situation where the simple algorithm fails. This result is not surprising when looking back at the design of the detection system. As the line detection algorithm also finds a lot of lines on the outline of the fake plant that are not filtered out through any of the implemented conditions, the lines are added to the line fitting array. Now, the fitting algorithm tries to match points on both the rod and plant with varying results. The line can be fitted on the rod as desired, but can also be fitted on the plant. Lastly, it can also be placed sideways to fit to both objects.

This problem is outside the scope of this project and suggestions for improvements will be given in the following chapters.

Chapter 5

Conclusions

A robot arm was developed that can be more effective and efficient at executing repetitive tasks in a greenhouse compared to previous automation attempts in the horticulture sector described in Section 2.1. Effective means the robot can achieve its tasks on every plant, so it should have a success rate of 100% and efficient means that the grower will earn back his investment in the robots compared to labour costs.

The arm design is focused on tomato greenhouses, which means the robot should move in the greenhouse rows to attend to the plants. As a consequence, the robot should be mobile, both on the rails between the plants and on the central concrete path. The proposed design only involves a robot arm, but the cost and complexity of the under-carriage inspired a design philosophy in which multiple arms are on one single robot. The plants require various maintenance steps that are currently performed by hand, but can be performed simultaneously by the proposed robot. Several important factors of the robot design were formulated into requirements, which will be shortly reviewed in this chapter.

Cost. The type of robot arm that is chosen is a SCARA arm. In this design, most movement happens in a horizontal plane, which can then be driven vertically along a central pole that supports multiple robot arms. Figure 3.2 illustrates the layout. The motors, electronics and other materials sum up to a total of €3,700 where assembly should be added. This means the cost of the arm is in the correct order of magnitude relative to the required €5,000 to make the robot efficient.

Fast Planning and signal processing on the arm itself. The planning (inverse kinematics) and control of the arm are done in a specialised way that is optimal for the type of robot. This means that a simple, low-powered ARM processor can perform the planning of a trajectory within the specified 10 milliseconds, as was shown in Section 4.1.2. Most required movements will be from the middle of the path towards a plant or vice versa, which is 30cm. For this distance, the planning time is only 5ms. This is a significant improvement compared to the off-the-shelf planning package MoveIt, which takes a second to calculate a trajectory on a laptop-class processor for a path of 5 centimetres. Our high-speed design obviates the need for a large control box housing a desktop PC. Using the specialised planner that was created allows the robot arm to move to any position without delays. Collision checking was also implemented

for checking the arm against itself, known as self-collision.

Automate multiple tasks. Another requirement for the project is to have the ability to automate multiple tasks. This constraint is set because the robots that are developed for a single task are too expensive to be efficient. If multiple tasks can be performed, the costs of the robot are shared across the labour costs of multiple greenhouse tasks. In this manner, the cost per automated task goes down. As the robot arm can plan its own path independently of other systems on the robot, working independently on multiple tasks is a possibility. The independence is made possible by the SCARA arm design, since it allows each arm to move in 2D in its own horizontal slice of the workspace.

However, due to time constraints, only growth guidance was tested. But, it can be argued that since the robot arm can smoothly move to any location around in its operating region, automating other tasks will be a similar process to the growth guidance automation. Therefore, the ability of the robot arm to automate four tasks can be achieved in the future, after the development of specialised end-effector tools and software to localise thieves, leaves and tomatoes.

Multiple arms working together. If a situation arises where having an arm help another increases the success rate, working together increases the effectiveness of the robot. An example of such a situation is when a sucker is on the back of the plant, so the arm cannot reach it on its own. Now, with a second arm pulling the plant into the middle of the greenhouse row, the sucker is more easily removable. This requirement is similar to the automation of multiple tasks requirements, where it is incorporated into the design, but not tested.

High success rate. With an estimated success rate of around 95% the results of the growth guidance repeatability test in Section 4.2.1 show that the system works rather well, but should be improved further to reach the ideal 99.9% target.The measured success rate is much higher compared to projects described in related work. That being said, the other projects measured the success rates on a complete action in the greenhouse, instead of a simplified action in a lab environment without any plants. Still, it can be argued that the placement and retrieval of the clipper is the most precise and critical part of the action and with a proper machine learning based detection of the clipper the success rate in the greenhouse can be close to this result.

The clipper rod detection algorithm improves the robustness of the growth guidance system in a lab environment but fails if plants surround the rod. However, this behaviour was already expected ahead of the tests. To prevent this problem in the future rod detection should be done using more advanced machine vision, most likely using machine learning.

Positional accuracy. The accuracy of the arm is not as good as anticipated and during motions the required accuracy of 1cm is not met. The results described in Section 4.1.1 show the measured accuracy and describe how the accuracy is acceptable after the robot arm stops moving at the end of a trajectory.

To conclude, the designed robot arm mostly satisfies the set requirements, but more tests should be conducted and some aspects of the design can still be improved, as will be discussed in the next chapter.

Chapter 6

Future Work

The work presented in this thesis can be seen as a basis for the automation of many greenhouse tasks. The tasks listed in Section 1.2 form the majority of the workload, but smaller tasks might arise as well. Keeping the total automation cost low means minimising the price per task per hectare. The fact that a specific task should be performed every n days, usually a week, limits the amount of greenhouse area that a single robot can cover. To be more productive, it was proposed in this thesis that a robot should do multiple tasks in parallel. This has not been tested in practice, however, and therefore some work is required here. In simulation, multiple arms can move independently, but this should be tested in practice. With these practical tests, extra attention should be given to the situation where multiple arms are working together on a task since this requires the arms to synchronise their movements, requiring a stable, low-latency connection between the arms across the ROS2 system.

For the growth guidance, the work needs to be expanded such that a plant can be caught with the clipper after it is removed. Then, the arm should be able to move this part of the plant towards the clipper rod such that the clipper can be attached, fixing the new part of the plant in place. This will require a more sophisticated rod localisation system as the current rod detection system fails when other items are in focus, however this problem can be solved using techniques such as machine learning. With this future localisation system, the clipper detection system can be tested in the greenhouse. Apart from this, there will also need to be an improvement to the reliability in placing and retrieving the clippers. To achieve this, the arm control should be improved to provide smoother motion during the visual servoing process and better error checking and handling should be performed to prevent other errors.

As discussed in the Results chapter, the joints did not perform as well as expected when tracking a path. This could be fixed with a parameter change within the motors, but the manufacturer of the joints (Igus), was reluctant to help with this reconfiguration. This, together with the fact that the motors regularly stop working after doing the exact same routine 50 times, gives reason to recommend the development of a robot arm that does not use the Igus joints. Instead, separate motors, reductions and drivers could be used, which would make the design more flexible long-term.

It has been argued that many arms should be fitted to a robot to optimise the cost ratio. Even so, there will be a practical limit to this strategy, since the

robot arms cannot pass each other on the central pole and will have to wait if another is in the way. Therefore, a study should be conducted into the optimal configuration for the greenhouse robot. This step should be taken after multiple tasks can be autonomously executed and data is acquired about the duration of each task and the height at which the tasks are performed. With this data the ideal configuration of the robot can be formulated.

Bibliography

[1] Agrimatie, WUR. Sectorresultaat glastuinbouw. https://www.agrimatie.nl/SectorResultaat.aspx?subpubID=2232&sectorID=2240, 2021. Last accessed: June 9, 2022.

[2] Boaz Arad, Jos Balendonck, Ruud Barth, Ohad Ben-Shahar, Yael Edan, Thomas Hellström, Jochen Hemming, Polina Kurtser, Ola Ringdahl, Toon Tielen, and Bart van Tuijl. Development of a sweet pepper harvesting robot. *Journal of Field Robotics*, 37(6):1027–1039, 2020.

[3] Rabab Benotsmane, László Dudás, and György Kovács. Trajectory optimization of industrial robot arms using a newly elaborated "whip-lashing" method. *Applied Sciences*, 10(23), 2020.

[4] Jan Bontsema, Jochen Hemming, Erik Pekkeriet, Wouter Saeys, Yael Edan, Amir Shapiro, Marko Hočevar, Roberto Oberti, Manuel Armada, Heinz Ulbrich, et al. Crops: Clever robots for crops. *Eng. Technol. Ref*, 1(1):1–11, 2015.

[5] Dry Hydroponics B.V. System - dry hydroponics. https://dryhydroponics.nl/system/. Last accessed: June 26, 2022.

[6] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, jan 1972.

[7] Yi Fang, Jie Hu, Wenhai Liu, Quanquan Shao, Jin Qi, and Yinghong Peng. Smooth and time-optimal s-curve trajectory planning for automated robots and machines. *Mechanism and Machine Theory*, 137:127–153, 2019.

[8] Tully Foote. tf: The transform library. In *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6, 2013.

[9] Lely Industries N.V. Lely. https://www.lely.com/. Last accessed: June 26, 2022.

[10] Math Open Reference. Proof of the law of cosines. https://www.mathopenref.com/lawofcosinesproof.html. Last accessed: June 25, 2022.

[11] Abdullah Mohammed, Bernard Schmidt, Lihui Wang, and Liang Gao. Minimizing energy consumption for robot arm movement. *Procedia CIRP*, 25:400–405, 2014. 8th International Conference on Digital Enterprise Technology - DET 2014 Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution.

[12] Open Robotics. Ros. https://www.ros.org/. Last accessed: June 14, 2022.

[13] Open Robotics. urdf - ros wiki. http://wiki.ros.org/urdf, 2019. Last accessed: June 14, 2022.

[14] OpenCV Team. Home - opencv. https://opencv.org/. Last accessed: June 18, 2022.

[15] Panasonic Corporation. Introducing ai-equipped tomato harvesting robots to farms may help to create jobs. https://news.panasonic.com/global/stories/2018/57801.html, 2018. Last accessed: May 23, 2022.

[16] Picknik Robitics. Moveit 2 documentation. https://moveit.picknik.ai/galactic/index.html, 2019. Last accessed: May 20, 2022.

[17] PickNik Robotics. Moveit motion planning framework. https://moveit.ros.org/, 2022. Last accessed: June 15, 2022.

[18] Priva B.V. Priva introduceert de eerste volautomatische bladsnijrobot voor de tomatenteelt in de wereld. https://www.priva.com/nl/ontdek-priva/blijf-op-de-hoogte/nieuws/priva-introduceert-volautomatische-bladsnijrobot, 2021. Last accessed: May 23, 2022.

[19] Raspberry Pi Foundation. Buy a raspberry pi 4 model b – raspberry pi. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/, 2019. Last accessed: June 17, 2022.

[20] Fabian Schwarzer, Mitul Saha, and Jean-Claude Latombe. *Exact Collision Checking of Robot Paths*, pages 25–41. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[21] Abhronil Sengupta, Tathagata Chakraborti, Amit Konar, and Atulya Nagar. Energy efficient trajectory planning by a robot arm using invasive weed optimization technique. In *2011 Third World Congress on Nature and Biologically Inspired Computing*, pages 311–316, 2011.

[22] Stanford Exploration Project. Huber function regression. http://sepwww.stanford.edu/public/docs/sep92/jon2/paper_html/node2.html. Last accessed: June 21, 2022.

[23] Stephanie Morris. The pythagorean theorem. http://jwilson.coe.uga.edu/emt669/student.folders/morris.stephanie/emt.669/essay.1/pythagorean.html. Last accessed: June 25, 2022.

[24] E. J. van Henten, J. Hemming, B. A. J. van Tuijl, J. G. Kornet, J. Meuleman, J. Bontsema, and E. A. van Os. An autonomous robot for harvesting cucumbers in greenhouses. *Autonomous Robots*, 13(3):241–258, Nov 2002.