FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Jarkko Kotaniemi

COMPLIANT MOTION FOR A MOBILE MANIPULATOR

Master's Thesis
Degree Programme in Computer Science and Engineering
May 2024

Kotaniemi J. (2024) Compliant Motion for a Mobile Manipulator. University of Oulu, Degree Programme in Computer Science and Engineering. Master's Thesis, 50 p.

ABSTRACT

Weed infestation is a significant problem to crop health and pasture yields, requiring targeted and efficient solutions. Current weeding practices mean spraying large amounts of herbicides. With mechanical weeding, the usage of toxic chemicals can be avoided.

In this thesis an automatic mechanical weeding robot is developed by creating the kinematic model and control solution for a robot arm with forward, inverse, and differential kinematics as well as compliant motion control for maintaining contact forces. In addition to the arm control, a navigation system for a mobile platform is created that includes extended Kalman filter with sensor integration for position estimation and coordinate navigation with path planning.

This weeding robot is tested in real-world outdoor setting, where it demonstrates autonomous navigation to weeds, weed detection, and removing them from the soil. By offering a viable alternative to conventional herbicide-based approach, this solution shows promise for a sustainable agricultural practice.

Key words: robotics, compliant, mobile, manipulator, agriculture, weeding.

Kotaniemi J. (2024) Joustava liike liikuteltavalla manipulaattorilla. Oulun yliopisto, tietotekniikan tutkinto-ohjelma. Diplomityö, 50 s.

TIIVISTELMÄ

Rikkaruohot ovat merkittävä ongelma viljelyskasvien ja laidunmaan terveydelle ja tuotolle. Ne vaativat kohdistettuja ja tehokkaita ratkaisuja. Nykyisillä kitkentätavoilla levitetään suuria määriä kasvimyrkkyjä, joiden käyttöä voidaan välttää käyttämällä myrkkyjen sijaan mekaanista kitkentää.

Tässä työssä kehitetään automaattinen mekaaninen kitkentärobotti luomalla kinemaattinen malli ja ohjausjärjestelmä robottikädelle, jossa on suora-, käänteinen- ja differentiaalikinematiikka sekä joustava liikeohjaus kosketusvoimien ylläpitämiseksi. Robottikäden lisäksi, liikkuvalle alustalle kehitetään navigaatiojärjestelmä, joka sisältää laajennetun Kalman-suotimen anturi-integraatiolla paikan estimointiin sekä koordinaattinavigoinnin reitinsuunnittelulla.

Kitkentärobotti testataan aidossa ulkoilmatilanteessa, jossa se osoittaa autonomisen navigoinnin rikkaruohoille, rikkaruohojen tunnistuksen sekä rikkaruohojen kitkennän maasta. Tämä ratkaisu on lupaava vaihtoehto kestävälle maataloudelle tavanomaisen kasvimyrkkypohjaisen menettelyn sijaan.

Avainsanat: robotiikka, joustava, liikuteltava, manipulaattori, maatalous, kitkentä.

TABLE OF CONTENTS

FOREWORD

This master's thesis was done in VTT Technical Research Center of Finland. The work was part of the FlexiGrobots project, funded by the EU's Horizon 2020 research and innovation programme (grant agreement No 101017111).

I would like to thank everyone at VTT for help during this thesis. I want to personally thank Dr Tapio Heikkilä for guidance and supervision, and Dr Eric Halbach and Mr Niko Känsäkoski for their help in the work.

Oulu, 2024

Jarkko Kotaniemi

LIST OF ABBREVIATIONS AND SYMBOLS

CAN	Controller Area Network
DH	Denavit-Hartenberg
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IMU	Inertial Measurement Unit
PCA	Principal Component Analysis
PLC	Programmable Logic Controller
ROS	Robot Operating System
RTK	Real-Time Kinematic
UI	User Interface

1. INTRODUCTION

The adoption of robotic technology in agriculture promises to be instrumental in reducing the reliance on traditional, resource-intensive methods. Weed infestation is a significant problem to crop health and pasture yields, requiring targeted and efficient solutions. Current weeding practices mean spraying large amounts of herbicides. With mechanical weeding, the usage of toxic chemicals can be avoided. This thesis centres around a mobile weeding robot equipped with a robot arm, designed to autonomously navigate the field, and detect and pick up weeds. The focus is on two major points: creating kinematic model, control system and compliant motion for a 6-axis robot arm is attached to a mobile robot platform, and the navigation systems of the mobile platform.

This thesis will describe the theory of kinematic modelling for robots, compliant motion, and robot navigation. In addition to the theory, it will describe methods of creating such system on a real case. In this case is a Schunk 6-axis robot arm LWA4P is used in a system that uses a weeding tool to grab and pull weeds from the ground. The system uses compliant motion to ensure that the tool is pushed far enough into the ground and to make sure that the tool stays in contact with the ground and takes advantage of the leverage intrinsic to the weeding tool. The robot arm is mounted on mobile robot platform. Navigation of the mobile platform includes path planning based on GNSS (Global Navigation Satellite System) coordinates and EKF (Extended Kalman Filter) localization with sensor integration.

The objective of this thesis is to create an autonomous weeding robot that is capable of navigating to weeds on a field and removing them from the ground. The robot will use the previously mentioned methods to achieve these goals. This thesis will demonstrate the use of these methods in a field test and analyse the results of the tests.

Some developments have been reported in the topic of automatic and mechanical weed removal. A robot system that relies on use of a drill to destroy the root of the weed has been proposed [1]. A robot with a vision system has been used also for precise spraying of herbicides only over the weeds [2]. There are already some weeding robots in the market. Odd.Bot [3] has developed a precision weeding robot for high density crops. It carries out both precise spreading of herbicides and mechanical in-row weed removal, the latter one with a delta robot structure [3]. Pixel Farming Robotics has developed an agricultural robot for smart farming, for cropping, reduced tillage, and smart crop rotation, and a sensor system equipped with many 3D cameras to be trained to recognize individual plants within a vegetation with AI [4]. However, the robots by Odd.Bot and Pixel Farming Robotics operate in particular types of fields, such as potato fields and “pixel farms”, and have not been applied in open pasture fields. An integrated robot system, suitable for open pastures, and where weed detection and removal technologies were integrated, seems to still be missing.

2. ROBOT ARM KINEMATICS

Kinematics in the context of robotics is the study of motion of robot manipulators. It involves the analysis of the relationship between the robot's joint angles, velocities and accelerations and the resulting position, velocity, and acceleration of the robot's end-effector.

A manipulator can be represented as a kinematic chain of rigid bodies or links connected by means of revolute or prismatic joints. One end of the chain is constrained to a base, while an end-effector is mounted to the other end. The resulting motion of the structure is obtained by composition of the elementary motions of each link with respect to the previous one. Therefore, in order to manipulate an object in space, it is necessary to describe the end-effector position and orientation. [5 p.39]

By using forward kinematics, inverse kinematics, and differential kinematics, we can plan robot motions and trajectories, control the robot's motion, and ensure that it moves smoothly and accurately. This chapter will describe some of the methods used to model the kinematics of a 6-axis robot manipulator with a spherical wrist.

2.1. Forward Kinematics

Forward kinematics is the process of calculating the position and orientation of a robot's end effector given the joint angles or actuator inputs. It is used to determine how the robot will move in its environment.

To perform forward kinematics, we need to define the robot's kinematic chain, which is a series of rigid bodies connected by joints. Each joint can be modelled as a degree of freedom (DOF) that can change the orientation or position of the next rigid body in the chain.

Once we have defined the kinematic chain and the DOFs, we can use the Denavit-Hartenberg convention to represent the transformation between two adjacent rigid bodies. Denavit-Hartenberg parameters, often abbreviated as DH parameters, are a set of four parameters used to describe the spatial relationships between consecutive coordinate frames in a robotic manipulator or a kinematic chain. They play a crucial role in robot kinematics, helping to model and calculate the position and orientation of robot links and joints.

The four DH parameters are as follows: Link length a_i represents the distance between the z-axis of the current frame and the z-axis of the next frame, measured along the common normal. It defines the length of the common perpendicular between the two axes. Link twist α_i represents the angle between the z-axes of the current and next frames, measured about the x-axis of the current frame. It defines the twist or rotation between the two frames. Link offset d_i represents the distance between the x-axes of the current and next frames, measured along the z-axis of the current frame. It defines the offset along the common normal between the two axes. Joint angle θ_i represents the angle between the x-axes of the current and next frames, measured about the z-axis of the current frame. It defines the joint angle or the variable that controls the joint's movement. The parameters are visually shown in Figure 1.

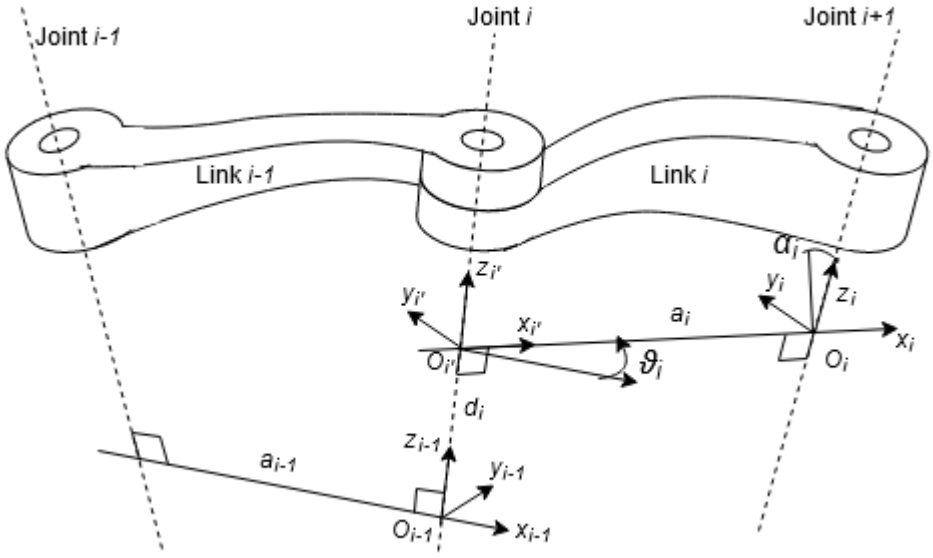

Figure 1. Denavit-Hartenberg kinematic parameters visualized.

The Denavit-Hartenberg parameters provide a systematic way to describe the geometric and kinematic relationships between robot joints and links. By assigning these parameters to each joint, a transformation matrix can be created that relates the coordinates of one frame to another. This matrix is typically used to compute the position and orientation of the robot's end-effector in a given configuration, enabling path planning, trajectory generation, and other essential tasks in robotics. The matrix is shown in Equation 1.

$$A_i^{i-1}(q_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \times \cos\alpha_i & \sin\theta_i \times \sin\alpha_i & a_i \times \cos\theta_i \\ \sin\theta_i & \cos\theta_i \times \cos\theta_i & -\cos\theta_i \times \sin\alpha_i & a_i \times \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

- a_i is the distance between O_i and O_{i-1} along the x-axis of the joint,
- d_i is the distance between O_i and O_{i-1} along the z-axis of the joint.
- α_i is the angle of the z-axes between joints about the x-axis,
- θ_i is the angle of the x-axes between joints about the z-axis.

The use of Denavit-Hartenberg parameters simplifies the mathematics involved in robot kinematics and makes it easier to perform forward and inverse kinematic calculations, which are essential for controlling and programming robots for various tasks and applications.

There is also a modified version of the DH parameters where the frames are set to the beginning of links unlike in the standard version where the frames are at the end of links. Both are equally valid if you know which of them are being used. [5 p.61-64]

A tool for visualizing the DH parameter is extremely useful when figuring out the parameters for a new robot [6]. The turning axes between joints are much easier to investigate when there is a model that can be moved around.

2.2. Inverse Kinematics for a Robot Arm with a Spherical Wrist

Inverse kinematics is the process of calculating the joint angles or actuator inputs required to achieve a desired end effector position and orientation. It is used to plan robot motions and trajectories.

The inverse kinematics problem means that in some configurations there can be multiple or even infinite solutions for a desired end-effector pose. There may also be no solutions. The problem also includes the fact that the equations to solve are in general nonlinear, so it is not always possible to find a closed-form solution.

For robot arms the spherical wrist is a very common configuration of the last three joints. For these types of configurations, there are standard solutions. A solution for the inverse kinematics of a spherical wrist with "shoulder up, elbow down" can be as follows.

First, we take the desired tool pose and go backwards from it with the length of tool offset and d_6 to get the wrist origin. The a_2 and d_4 values are part of the DH-parameters.

$$P_{wrist} = P_{tool} - (d_6 + d_{tool}) \times \overline{v}_{flange} \tag{2}$$

- P_{wrist} is the wrist pose.
- P_{tool} is the tool pose.
- d_6 is one of the DH-parameters.
- d_{tool} is the tool offset.
- $\overline{v}_{flange}$ is the flange direction.

Next, the arm joint value θ_1 can be calculated with the *x* and *y* coordinates of the wrist pose P_{wrist}.

$$\theta_1 = \operatorname{atan}\left(\frac{y_{wrist}}{x_{wrist}}\right) \tag{3}$$

Some helper variables are assigned based on some of the DH-parameters and the components of P_{wrist}:

$$c_3 = \frac{{x_{wrist}}^2 + {y_{wrist}}^2 + {z_{wrist}}^2 - {a_2}^2 - {d_4}^2}{2 \times a_2 \times d_4} \tag{4}$$

$$s_3 = -\sqrt{1 - {c_3}^2} \tag{5}$$

Now the shoulder θ_2 and elbow θ_3 joint values can be calculated using the helper variables and the P_{wrist} components:

$$\theta_2 = \operatorname{atan}\left(\frac{(a_2 + d_4 + c_3) \times z_{wrist} - d_4 \times s_3 \times \sqrt{{x_{wrist}}^2 + {y_{wrist}}^2}}{(a_2 + d_4 + c_3) \times \sqrt{{x_{wrist}}^2 + {y_{wrist}}^2} + d_4 \times s_3 \times z_{wrist}}\right) \tag{6}$$

$$\theta_3 = \operatorname{atan}\left(\frac{s_3}{c_3}\right) \tag{7}$$

The theta offsets from the DH-parameters needs to be added, of which there is only θ_3. With the first three joints calculated, the final three can be calculated. The joints are the ZYZ Euler angles of the rotation matrix between frames 3 and 6. The rotation matrix can be calculated by following the kinematic chain from frames 0 to 3 with the calculated joint angles and multiplying the transpose rotation to the *tool orientation* of the *tool pose*.

$$R_6^3 = (R_3^0)^T \times tool\ orientation\ = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{8}$$

- R_6^3 is the rotation matrix between frames 3 and 6.
- R_3^0 is the rotation matrix between frames 0 and 3.
- R_{ij} are the components of the rotation matrix.

$$\theta_4 = \text{atan}\left(\frac{R_{23}}{R_{13}}\right) \tag{9}$$

$$\theta_5 = \text{atan}\left(\frac{\sqrt{(R_{13})^2 + (R_{23})^2}}{R_{33}}\right) \tag{10}$$

$$\theta_6 = \text{atan}\left(\frac{R_{32}}{-R_{31}}\right) \tag{11}$$

With all the joints values calculated, the robot can be set to the desired pose [5 p.94-95]. In the case where the calculated joint values go over the joint limits of the robot, the pose is invalid.

2.3. Differential Kinematics

Differential kinematics is the process of calculating the relationship between the end effector velocity and the joint velocities. It is used to control the robot's motion and ensure that it moves smoothly and accurately.

To perform differential kinematics, we use the Jacobian matrix, which relates the end effector velocity to the joint velocities. The Jacobian matrix is a partial derivative of the forward kinematics equation with respect to the joint angles or actuator inputs.

By multiplying the Jacobian matrix with the joint velocities, we can calculate the end effector velocity. This allows us to control the robot's motion by specifying the desired end effector velocity and using the Jacobian to calculate the required joint velocities.

In the context of robot kinematics, the Jacobian matrix is a tool for linking joint velocities and end-effector linear and angular velocities. It conveys the relationship

between the motion of a robot's individual joint and the resulting movement of its end-effector in task space.

The Jacobian matrix represents the partial derivatives of the end-effector's position and orientation concerning the joint variables.

A singularity in a robot's workspace occurs when the Jacobian matrix becomes singular or near singular. Singularity in this context means that the robot loses one or more degrees of freedom, and its manipulability becomes limited. In other words, the robot arm is unable to move freely in all directions at that particular configuration.

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6}{\partial q_1} & \cdots & \frac{\partial f_6}{\partial q_6} \end{bmatrix} \tag{12}$$

$\partial f_i / \partial q_j$ represents the partial derivative of the ith component of the end-effector's velocity or angular velocity with respect to the jth joint velocity.

In order to reach the desired joint velocities $\dot{q}$, the Jacobian must be inverted and multiplied to the desired velocities of the end-effector v_e. The Jacobian must be calculated at the specific joint configuration q. [5 p.105]

$$\dot{q} = J^{-1}(q) v_e \tag{13}$$

A singularity in a robot's workspace occurs when the Jacobian matrix becomes singular or near singular. Singularity in this context means that the robot loses one or more degrees of freedom, and its manipulability becomes limited. In other words, the robot arm is unable to move freely in all directions at that particular configuration.

A singularity can be identified by examining the determinant of the Jacobian matrix. If the determinant is zero or very close to zero, it indicates that the matrix is singular or near singular, and the robot is at or near a singular configuration. In mathematical terms, a singularity occurs when the Jacobian matrix is not of full rank.

In practical terms there are a few positions where a singularity occurs such as the wrist singularity, where the 4th and 6th axes become coincident. [7]

3. FORCE CONTROL AND COMPLIANT MOTIONS

For a robot to interact with its environment, there must be some forces contacting the manipulator's end-effector. High amounts of contact forces are usually undesirable because of the stress they cause on both the manipulator and the manipulated object. To reduce the stress, different force control schemes that control the motions through force feedback can be used.

For any motion control through force feedback to happen, there needs to be some way of sensing the forces involved. The most common force sensing method for industrial robots is the force sensor that is attached to the end-effector to measure the direction and magnitude of forces acting on it.

Compliance in this context implies the ability of a robot or system to yield or deform in response to external forces or disturbances. It essentially means that the system can change its position or force application to accommodate changes in the environment.

In practice, errors may give rise to a contact force causing a deviation of the end-effector from the desired trajectory. On the other hand, the control system reacts to reduce such deviation. This ultimately leads to a build-up of the contact force until saturation of the joint actuators is reached or breakage of the parts in contact occurs.

The higher the environment stiffness and position control accuracy, the more likely a situation like the one just described can occur. This drawback can be overcome if compliant behavior is ensured during the interaction. [5 p.365]

3.1. Passive Compliance

Passive compliance refers to the ability to yield or give way when encountering an external force. In robotics, passive compliance is achieved through various mechanisms such as compliant joints, flexible materials, or spring-loaded systems that can absorb and mitigate forces. These mechanisms enable the robot or tool to yield or deform slightly when encountering an obstacle or applying force, preventing damage to the robot itself and the objects it interacts with, or it can assist force control by reducing stiffness. In a stiff system contact forces rise very fast as the manipulator moves into the object or surface. [5 p.366][8]

3.2. Active Compliance

Active compliance is the capability to actively adjust and control the amount of force applied in response to stimuli. It involves the use of sensors to adjust the applied force or torque in real-time. This concept is important in situations where there is a need to interact with dynamic or uncertain environments, or when precision and adaptability is force applications are crucial. [5 p.367]

Active compliance control can be divided into two categories which are force control and impedance control. Force control is a control technique where both the desired interaction force and robot position are controlled. Force is measured in real time to realize the feedback control. It is essential to properly design the mechanical part and sensory systems at the contact point of the robot. Force control always

requires a good sensory system. Insufficient sensory information can lead to errors in the controller.

Impedance control uses the different relationships between the acting forces and manipulator position to adjust the mechanical impedance of the end-effector to the external forces. The most common types of impedance control are stiffness, damping, and general impedance. The primary issues in the implementation of impedance control happen in the control stage. The more accurate parameters are applied in the contact model, the less errors will be generated.

Other aspects of impedance and force control are that both are dependent on the effectiveness of the electrical hardware. Noise and fluctuations in the signals affect the performance of force control. [9]

4. MOBILE ROBOT NAVIGATION

Navigating a mobile robot in a real environment involves a mix of mathematics, sensor integration and advanced algorithms. In this chapter are described the kinematics of differentially wheeled robots, the transformation of GNSS coordinates to local coordinates and the extended Kalman filter.

4.1. Kinematics of a Differentially Wheeled Robot

A robot with two separately driven wheels mounted rigidly, is a differential wheeled robot. Such robots usually have one or two castor wheels mounted on the back to prevent tilting. In order for a differential wheeled robot to turn, the two driven wheels have to be driven at different speeds. When the robot turns, it turns around the instantaneous center of rotation (ICR) with the radius R. The ground speeds v_L and v_R of the wheels lead to the angular velocity ω of the robot. The width of the robot is b. The parameters are shown in Figure 2.

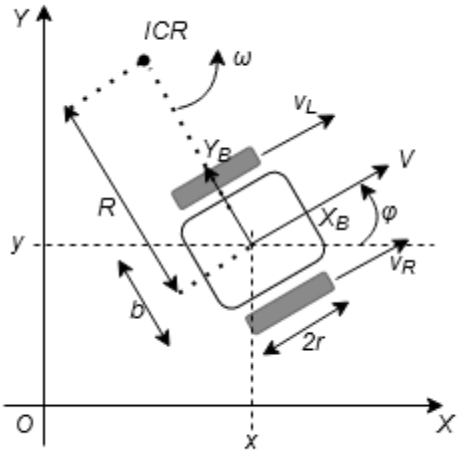

Figure 2. Differential drive kinematics' parameters visualized.

$$\omega = \frac{v_R - v_L}{b} \tag{14}$$

$$R = \frac{b}{2} \times \frac{v_R + v_L}{v_R - v_L} \tag{15}$$

The velocity V of the robot in the midpoint between the wheels is:

$$V = \omega \times R = \frac{v_R + v_L}{2} \tag{16}$$

The kinematic model of the robot in global coordinates is:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} cos\varphi & 0 \\ sin\varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix} \quad (17)$$

- $\dot{x}$, $\dot{y}$, and $\dot{\varphi}$ are the changes of the x, y coordinates and change of the orientation.
- V is the velocity.
- ω is the angular velocity.

If instead of calculating how the robot moves based on wheel velocities, the velocity and angular velocity of the robot are used as inputs for a controller. The wheel velocities are:

$$v_R = V + \frac{\omega \times b}{2} \quad (18)$$

$$v_L = V - \frac{\omega \times b}{2} \quad (19)$$

After the wheel velocities are calculated and the wheels are turning at that velocity, the robot will follow the input parameters if there is no wheel slip involved. [10 p.478]

4.2. GNSS Coordinate Transformation to Local Coordinates

The GNSS coordinates can be transformed to local coordinates. Radius of the Earth and the local latitude are needed for the transformation. The local coordinates are in relation to the initial latitude and longitude. Towards the east x is positive and y is positive towards the north. In small enough areas the local surface of the Earth can be assumed to be flat. The latitude can be turned into north-south distance by multiplying the change in latitude by the circumference of a sphere, in this case the Earth. For east-west distance we need to consider the latitude. At higher latitudes the east-west distance around the sphere shrinks so the distance needs to be scaled. The change in circumference is visualized in Figure 3.

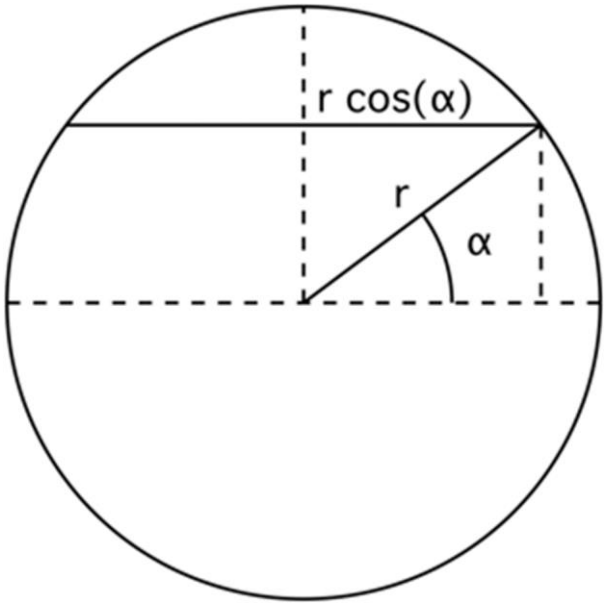

Figure 3. The circumference of a circle around a sphere at a certain latitude.

$$C_{Earth} = R_{Earth} \times \frac{\pi}{180} \tag{20}$$

$$C_{Earth\ at\ latitude} = R_{Earth} \times \frac{\pi}{180} \times cos\left(65.057° \times \frac{\pi}{180}\right) \tag{21}$$

$$y_{local} = C_{Earth} \times (lat_{cur} - lat_{init}) \tag{22}$$

$$x_{local} = C_{Earth\ at\ latitude} \times (lon_{cur} - lon_{init}) \tag{23}$$

- R_{Earth} is the radius of Earth,
- C_{Earth} is the circumference of Earth at the along the longitude lines,
- $C_{Earth\ at\ latitude}$ is the circumference of Earth at the latitude of 65.057° (Oulu, Finland).

The error of the transformation will be more exaggerated nearer to the poles. [11]

4.3. Extended Kalman Filter

The Extended Kalman Filter (EKF) is a powerful tool in mobile robot navigation that addresses the challenges of uncertainty and dynamic environments. As robots navigate through real-world scenarios, sensor measurements are susceptible to noise and inaccuracies. The EKF serves as a sophisticated estimation algorithm, combining sensor data with a dynamic system model to refine and update the robot's state estimation.

The EKF algorithm goes as follows: Make an inertial prediction estimate based on the current state and the control inputs, calculate the process covariance, and predict the state covariance. Update the estimate with sensor information by taking the

difference of the prediction and measurement and using that residual and the predicted state covariance to calculate the Kalman gain, which is used in the update the estimate. Finally, update the state covariance. The algorithm is described mathematically below.

The current state of the robot is described as the state vector $\boldsymbol{X}_t$.

$$\boldsymbol{X}_t = [x_t, y_t, \varphi_t, v_t] \tag{24}$$

where x_t and y_t are the robot's x and y coordinates, φ_t is the robot's heading angle, and v_t is the robot's current velocity.

The robot is controlled with velocity commands, so the input vector $\boldsymbol{u}_t$ has the input velocity v_t and angular velocity ω_t.

$$\boldsymbol{u}_t = [v_t, \omega_t] \tag{25}$$

The robot also receives its position from the GNSS system, so the observation vector $\boldsymbol{z}_t$ has the x and y coordinates as well as the heading from the compass.

$$\boldsymbol{z}_t = [x_t, y_t, \varphi_t] \tag{26}$$

The robot model is:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} cos\varphi & 0 \\ sin\varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix} \tag{27}$$

So, the motion model is:

$$\boldsymbol{X}_{t+1} = f(\boldsymbol{X}_t, \boldsymbol{u}_t) = F\boldsymbol{X}_t + B\boldsymbol{u}_t \tag{28}$$

where

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{29}$$

$$B = \begin{bmatrix} \cos(\varphi)\Delta t & 0 \\ \sin(\varphi)\Delta t & 0 \\ 0 & \Delta t \\ 1 & 0 \end{bmatrix} \tag{30}$$

The process noise covariance Q is:

$$Q = BMB^T \tag{31}$$

where

$$M = \begin{bmatrix} {\sigma_v}^2 & 0 \\ 0 & {\sigma_\omega}^2 \end{bmatrix} \tag{32}$$

where ${\sigma_v}^2$ and ${\sigma_\omega}^2$ are the variance the velocity and angular velocity inputs.
The motion function is:

$$\begin{bmatrix} x' \\ y' \\ \varphi' \\ v' \end{bmatrix} = f(\boldsymbol{X}, \boldsymbol{u}) = \begin{bmatrix} x + v\cos(\varphi)\Delta t \\ y + v\sin(\varphi)\Delta t \\ \varphi + \omega\Delta t \\ v \end{bmatrix} \tag{33}$$

The Jacobian of the motion function is:

$$J_f = \begin{bmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} & \frac{\partial x'}{\partial \varphi} & \frac{\partial x'}{\partial v} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} & \frac{\partial y'}{\partial \varphi} & \frac{\partial y'}{\partial v} \\ \frac{\partial \varphi'}{\partial x} & \frac{\partial \varphi'}{\partial y} & \frac{\partial \varphi'}{\partial \varphi} & \frac{\partial \varphi'}{\partial v} \\ \frac{\partial v'}{\partial x} & \frac{\partial v'}{\partial y} & \frac{\partial v'}{\partial \varphi} & \frac{\partial v'}{\partial v} \end{bmatrix} \tag{34}$$

$$J_f = \begin{bmatrix} 1 & 0 & -v\sin(\varphi)\Delta t & \cos(\varphi)\Delta t \\ 0 & 1 & v\cos(\varphi)\Delta t & \sin(\varphi)\Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{35}$$

The observation model is:

$$\boldsymbol{z}_t = g(\boldsymbol{X}_t) = H\boldsymbol{X}_t \tag{36}$$

where

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{37}$$

The measurement noise R is:

$$R = \begin{bmatrix} {\sigma_x}^2 & 0 & 0 \\ 0 & {\sigma_y}^2 & 0 \\ 0 & 0 & {\sigma_\varphi}^2 \end{bmatrix} \tag{38}$$

where ${\sigma_x}^2$ and ${\sigma_y}^2$ are the variances of the x and y coordinates of the GNSS measurement and ${\sigma_\varphi}^2$ is the variance of the compass reading.
The observation function states that:

$$\begin{bmatrix} x' \\ y' \\ \varphi' \end{bmatrix} = g(\boldsymbol{X}) = \begin{bmatrix} x \\ y \\ \varphi \end{bmatrix} \tag{39}$$

The Jacobian of the observation function is:

$$J_g = \begin{bmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} & \frac{\partial x'}{\partial \varphi} & \frac{\partial x'}{\partial v} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} & \frac{\partial y'}{\partial \varphi} & \frac{\partial y'}{\partial v} \\ \frac{\partial \varphi'}{\partial x} & \frac{\partial \varphi'}{\partial y} & \frac{\partial \varphi'}{\partial \varphi} & \frac{\partial \varphi'}{\partial v} \end{bmatrix} \tag{40}$$

$$J_g = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{41}$$

With the model specified the state covariance is initialized and the algorithm can begin.

$$P_t = P_{est} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{42}$$

Predict:

$$\boldsymbol{X}_{pred} = F\boldsymbol{X}_t + B\boldsymbol{u}_t \tag{43}$$

$$P_{pred} = J_f P_t {J_f}^T + Q \tag{44}$$

Update:

$$z_{pred} = H\boldsymbol{X}_{pred} \tag{45}$$

$$y = z - z_{pred} \tag{46}$$

$$S = J_g P_{pred} {J_g}^T + R \tag{47}$$

$$K = P_{pred} {J_g}^T S^{-1} \tag{48}$$

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_{pred} + Ky \tag{49}$$

$$P_{t+1} = \left(I - KJ_g\right)P_{pred} \tag{50}$$

where y is the residual vector, S is the innovation or residual covariance, and K is the Kalman gain. $\boldsymbol{X}_{t+1}$ contains the new estimated position and heading. [12]

Figure 4 shows a simulation of the algorithm described being used. As the process goes on the prediction covariance goes down. In this simulation the robot is moving forward with variable velocity and angular velocity. There is some added random noise

in the control variables, but the true values are used as the inputs. The measurements of the location are randomly distributed around the nominal position. The simulation was done on a python adaptation using the EKF algorithm described in this chapter.

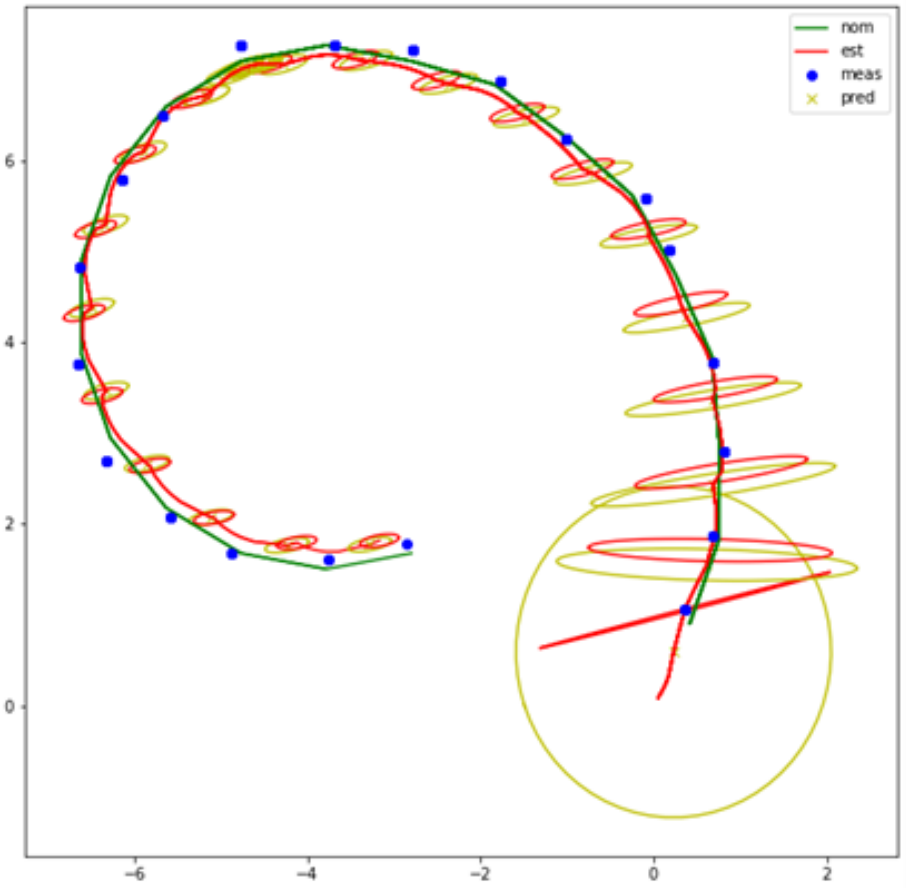

Figure 4. Simulation of the extended Kalman filter.

5. MOBILE TEST PLATFORM

The mobile test platform used in this thesis is a platform built originally by Probot Oy. It has been upgraded and modified over time to fit the needs of the project it is currently a part of. In this chapter are descriptions of the components of the platform with pictures.

The platform is made from steel. It has two motors in the front for movement and two caster wheels in the back for balance. Inside the frame is the battery system with two 48 V 16 Ah batteries for electricity and an amplifier as well as an analogue input module for the force sensor. On top there is a robot arm and emergency switches for the motors and the arm separately. An overview of the robot is shown in Figure 5.

Figure 5. Weeding robot outside.

The hardware architecture is shown in Figure 6. The robot is controlled by a laptop computer. There is an IMU sensor for angular velocity and compass measurements. There are two CAN bus converters, one for the motors and one for the arm and force sensor. The separate CAN buses are because they use a different baud-rate. On the arm is the force sensor, a 3D camera, and the weeding tool.

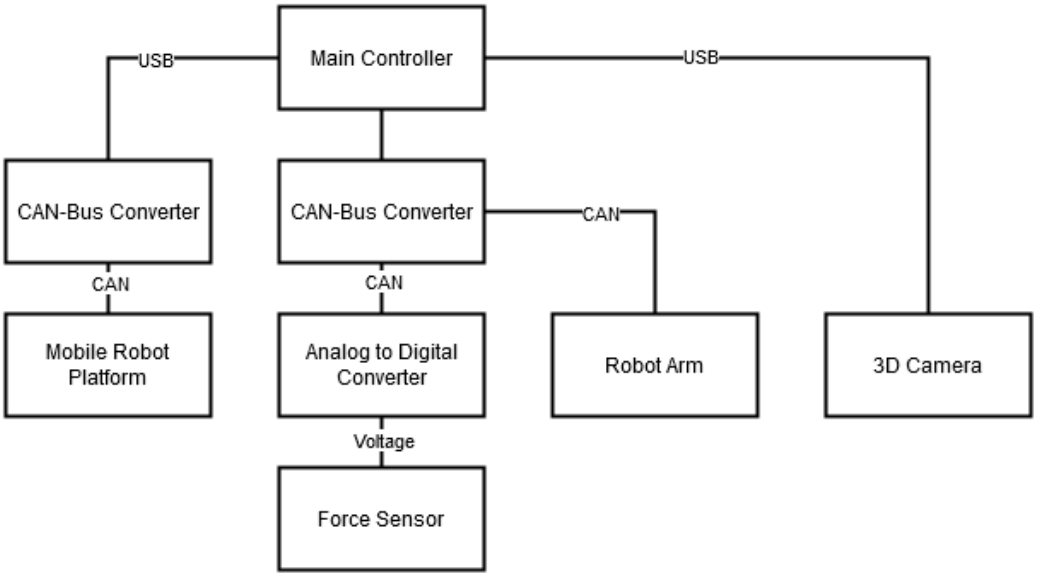

Figure 6. Hardware component diagram.

5.1. Mobility Module Motors

The motors used by the mobile platform are Probot Oy's mobility modules. They are controlled through the CAN bus with commands to set their "torque" value. The value correlates to the speed of the wheels up to a certain point so it is adequate for controlling them.

The wheels are set so that by controlling the speeds of the motors, the robot's forward and angular velocity can be controlled with equations for differential wheeled robots.

The modules have quite wide and large tires that give them acceptable grip and weight distribution for offroad driving.

5.2. Schunk LWA4P Robot Arm

The Schunk LWA4P is a lightweight 6-DoF robotic arm consisting of three major compact joints called ERB modules that integrate two perpendicular axes along with their overall electrical control circuits. Communication to the modules is done by CAN bus with the CANopen standard messages. The robot is shown in Figure 7.

The robot arm used to be controlled with a KEBA PLC, but it was changed to a direct CAN bus controller running on the same software that controls the whole system [13].

The robot weighs 15.3 kg and has a recommended workpiece weight of 6 kg including the gripper. Each axis has a maximum angular velocity of 72 °/s and the maximum angular acceleration of 150 °/s^2. All the axes have the possible angle of rotation of +/- 170° except the third axis which is limited by software to +/- 156.5°. The nominal voltage of the robot is 24 VDC, the average current input is 3 A, and the maximum current input is 12 A.

Figure 7. Schunk LWA4P with a gripper attached.

The DH-parameters of the robot are described in Table 1 [14]. Because of the construction of the joint servos, joints 3 and 5 rotate in the opposite direction than expected. The directional correction is done in software so that each joint turns in the expected direction.

Table 1. Denavit-Hartenberg parameters for Schunk Powerball LWA 4P.

Link	$\mathbf{a_i}$	$\boldsymbol{\alpha_i}$	$\mathbf{d_i}$	$\boldsymbol{\theta_i}$
1	0	$\pi/2$	0	0
2	0,35	0	0	0
3	0	$\pi/2$	0	$\pi/2$
4	0	$-\pi/2$	0,305	0
5	0	$\pi/2$	0	0
6	0	0	0,075	0

5.3. PEAK CAN Bus Converter

To access the CAN bus connected to the motors, the robot arm and the force sensor, a USB to CAN bus converter by PEAK System is used. CAN commands are sent and received to and from the converter using Qt's CAN libraries.

The CAN bus is an automation bus used in vehicles, machines, and industrial devices. Data in the bus is transmitted to every device with an identifier that the receiving modules use to decide if the data is meant for them.

5.4. ME-Systeme Force Sensor

The force sensor used was ME-Systeme K6D40 (Figure 8). It is a 6-axis force-torque sensor meaning it can sense linear forces and torques in the 3 different axes. The way it senses these values is by having six embedded strain gauges set in a specific geometric arrangement inside the device. The actual force and torque values are then

given by the cross product of a calibration matrix and a vector of the six sensor signals. The calibration matrix is unique for each force sensor and is provided by the manufacturer.

The sensor can measure nominally 200 N in the x- and y-axes and 500 N in the z-axis. Torque measurements are 10 Nm in every axis. The accuracy of the measurements is 0.2%. [15]

Figure 8. ME-Systeme K6D40 force-torque sensor.

5.5. Electrical Systems

The robot is powered by two 48 V lithium-ion batteries in parallel. The voltage is reduced to 24 Volts which is used by all the other components. All the components are controlled through the CAN bus. The robot arm and the force sensor are in a separate bus to the motors because they have a different baud rate. The force sensor is connected to an amplifier, which in turn is connected to a CAN-analogue input module, where the analogue signal is converted to digital CAN data.

The electrical schematic is in Figure 9.

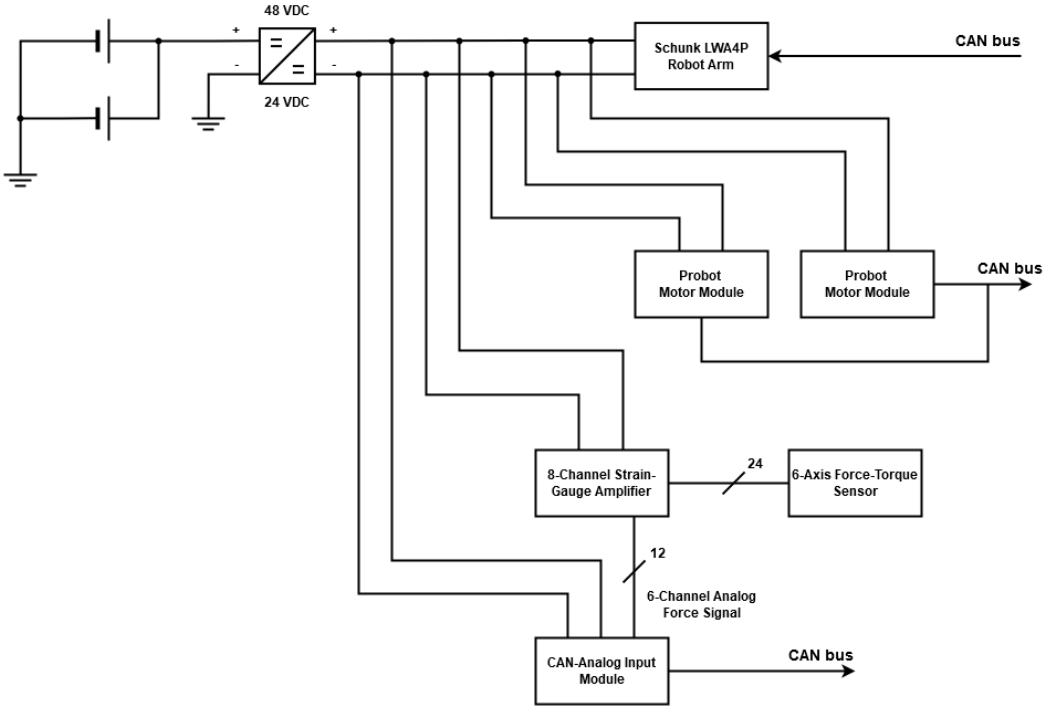

Figure 9. Electrical schematic of the robot.

5.6. Intel Realsense 3D Camera

For detecting weeds, an Intel Realsense D415 3D camera was used (Figure 10). The camera provides an RGB image and a 3D point cloud (Figure 11). The pixel coordinates of the RGB image can be projected to the point cloud, so that when the weed is detected in the RGB image, a 3D position of the weed can be found.

The D415 uses stereo vision technology, capturing the scene with two images and by computing the disparity on the two images, depth can be retrieved [16]. The operating range for the camera is between 0.5 and 3 meters. The accuracy of the depth image is around 1 to 2 mm. The field of view of the depth sensing is 65° × 40°. The resolution is of the depth image is up to 1280 × 720, while the resolution of the RGB image is 1920 × 1080. The camera uses a rolling shutter.

Figure 10. Realsense D415 3D camera.

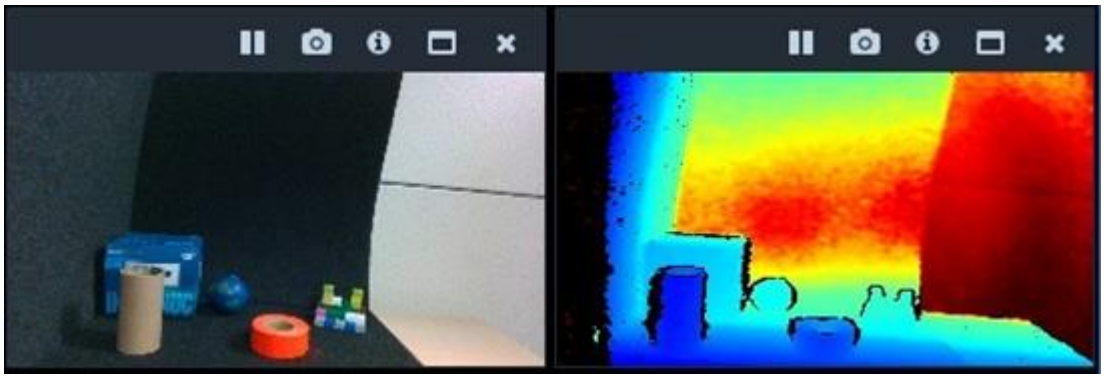

Figure 11. RGB and depth image of the Realsense D415. In the Depth image, blue is closer and red is farther away.

5.7. Fiskars Weeding Tool

The tool for removing weeds is a Fiskars Xact weed puller (Figure 12) that is attached to the robot arm's flange with the force sensor on a linear rail with a spring (Figure 13). The rail and spring are there to give some elasticity to the tool and to make it so that the force control does not need to control a totally rigid connection.

Manual alternatives of weed management prevent the negative impacts of chemicals. Many weed species have developed resistance to weedicides and the misuse of chemical herbicides leads to risk of environmental safety. [17]

The weeding tool uses its lever mechanism to pinch the weed in its claws and when it is turned back the lever becomes a fulcrum for extra leverage for pulling. The weeding tool has three serrated, stainless-steel claws firmly grab weeds and roots.

Figure 12. Fiskars Xact weed puller attached to the robot.

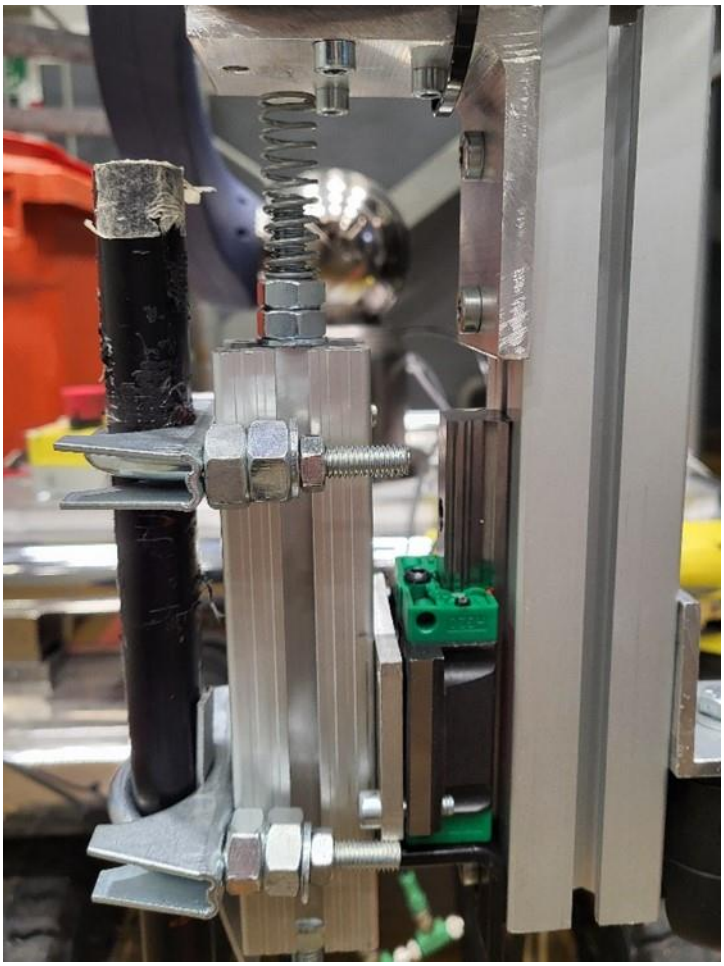

Figure 13. Spring-loaded linear rail with the tool attachment.

The reason for having a spring-loaded linear rail in between the tool and the robot is that it helps the force control. Without any flexibility, contact forces will result in fast changes of measured forces. With the spring, there is a buffer zone where the forces will change slowly and there is more time to control them. It also prevents reaching the force limits of the robot and helps protect it from damage.

The spring constant of the spring in the spring-loaded linear rail was calculated by measuring the compression of the spring by setting objects of different mass on top of it. Measured values shown in Table 2. The average of the calculations was used as the spring constant which is 2061 N/mm.

Table 2. Spring constant based on compression and mass.

Mass	Compression	Force	Spring constant
m [g]	x [mm]	F [N]	k [N/mm]
769	4	7.544	1886
1307	6	12.82	2137
2423	11	23.77	2161

5.8. U-Blox GNSS Receiver

For outside navigation purposes, the platform has a U-blox ZED-F9P high precision GNSS receiver. The receiver on its own is capable of around 5-meter accuracy, which can dramatically be improved to around 5 – 10 cm accuracy using Point Perfect GNSS augmentation service. The receiver is connected by USB and has a multiband antenna for satellite connection. The receiver concurrently uses GNSS signals from the GNSS constellations: GPS, GLONASS, Galileo, BeiDou, and NavIC.

Point Perfect is a U-blox service for improving the GNSS accuracy by sending correction data. Instead of having a separate base station to compare GNSS errors like with RTK (Real-Time Kinematic), the service uses a virtual base station combined with a correction model of regional GNSS errors, to achieve fast and precise positioning. [18]

6. WEEDING ROBOT CONTROL SOFTWARE

The software that controls the robot was written in C++ with Qt. The programs themselves are the weeding robot controlling software that includes functionality for moving the platform and the robot arm, GNSS based navigation, kinematic model for the Schunk LWA4P robot arm with forward, inverse, and differential kinematics and the ability to plan and send movement commands to the arm. Also included is the functionality for creating and executing missions for automatic use.

The other pieces of software used are the weed detector software that takes images with a 3D camera and calculates a 3D position of weeds from that image as well as a ROS node for interfacing with the GNSS receiver. The control software sends a command to the weed detector to take an image with the camera and the detector responds with the calculated weed position. The U-Blox GNSS receiver ROS node communicates with the receiver itself and posts the received coordinates to a ROS topic. The control software subscribes to that topic to receive the latest coordinates. The connections are shown in Figure 14.

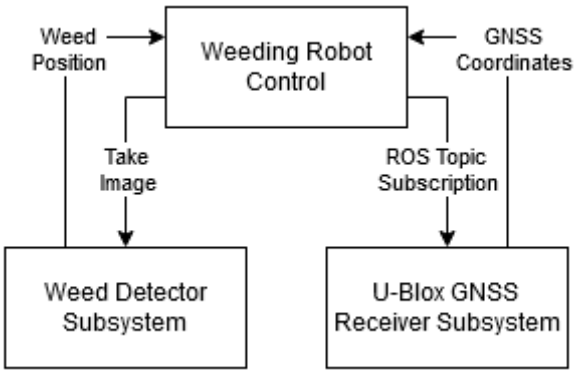

Figure 14. Software subsystem connections and data flow.

The control software functionality is split into multiple classes for easier development and asynchronous execution. Each class has a set of responsibilities that is supposed to carry out. A short description of each class and its purpose are as follows:

- WeedingRobot Control UI is the parent class that has every other class as a child object, and it also has the UI elements, including control buttons, status messages, and data displays.
- MissionControl contains the mission, which is a list of tasks. It sends the next task in the list to TaskControl to execute.
- MissionTaskDialog is a UI class for editing the current mission.
- TaskControl contains information on how each task is executed and it also keeps track of what control mode the robot is in. The modes indicate how the robot is controlled, for example mission mode, joystick mode, and coordinate drive mode.
- PathControl is the class that keeps track of the path the platform moves in. It receives coordinate waypoints and creates a path for the platform. It also calculates the direction where the robot needs to go.
- CoordinateDriveDialog is a UI class for selecting coordinates for the robot to move towards.

- TrajectoryControl takes the set direction of the platform and calculates the required motor speeds to turn and move towards that direction.
- MotionControl receives the motor speeds and modifies them into the format required by the motors. In simulation mode it also sends motion information to LocationEstimator.
- CANBusConnection sends data and receives data to and from the CAN bus.
- LocationEstimator receives data from the sensors and uses the EKF to estimate its position and direction.
- SensorData reads data from the IMU sensor and the GNSS receiver and sends it forward.
- RobotArmControl controls the robot arm. It has a state machine that contains all the bytes required to be sent in order to enable the arm. It also computes the movement paths for point-to-point motion as well as the movement increments for liner motion. Finally, it has the force control for compliant motion.
- RobotArmMoveDialog is a UI class for sending motion commands to the robot arm.
- RobotKinematics is a class that calculates the forward, inverse, and differential kinematics for the robot arm.
- ForceSensor is a class that converts the voltage signals from the CAN bus to Newtons and Newton-meters with a calibration matrix. It also filters the force signal with a low-pass Butterworth filter.

The connections are visualized in Figure 15. These classes are connected to either each other or external software and hardware. The class connections are done using Qt's signal and slot system, where objects are connected to each other by connecting a signal to a slot. Emitted signals trigger a slot method in the recipient object. The signal can contain data that is used as a parameter in the method execution. External connections to hardware are mostly done with the CANBusConnection class to the CAN-bus through a PEAK USB-CAN converter. The IMU sensor is connected through USB directly and is communicated with serial communication. The external software connections are done in the case of the U-Blox GNSS sensor with the use of a ROS topic subscription and in the case of the Weed Detector, using a TCP/IP socket.

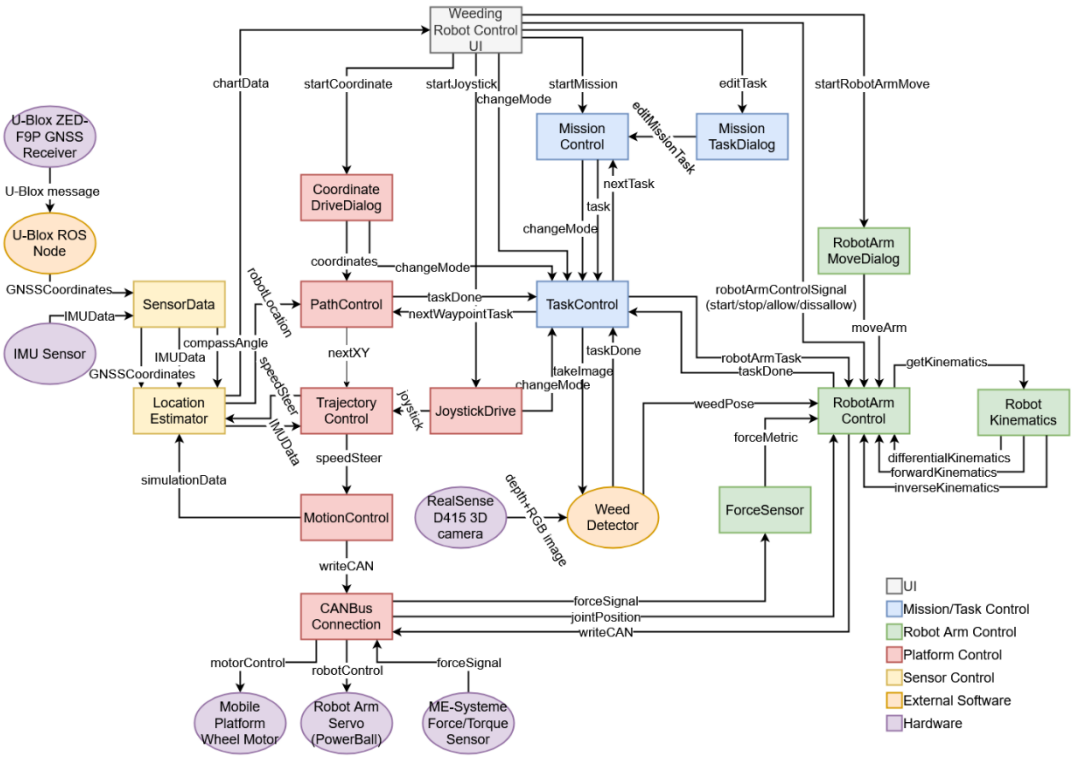

Figure 15. Signal connection diagram with the software classes, external software, and the hardware.

Typical execution sequence of the control software in the case of running a weeding mission is in the following way:

1. The robot is started in the UI by pushing a start button. This starts all the objects in standby mode, where they are ready to execute.
2. The robot arm is turned on from the UI by pushing the robot arm start button and waiting for the status message to say it is started.
3. The robot arm also needs to be allowed to move. This is done by pushing the allow button. This disengages the breaks from the robot arm which is audible as a clicking sound. The robot arm is now ready to receive motion commands.
4. The mission is set or loaded in the UI. The mission includes the transfer tasks of the platform moving to specified coordinates, taking pictures of the weed, and the weeding action.
5. Finally, the start mission button is pressed, and the mission execution begins.

The user interface of the software is shown in Figure 16. It was made using QtCreator, which offers easy methods to create UIs with widgets and connections to them. On the top of the UI are the buttons to power on the systems and start the control loops. Below that are buttons to start mission execution, manual control with JoystickDrive, a coordinate move with CoordinateDrive, and buttons to start and enable movements on the robot arm. In addition, there is a checkbox to switch the control to simulated mode, where all sensor data and movements of the platform are simulated. Next, there are charts to track the location, speed, steering of the robot, as well as the compass angle. Under the compass chart, there are options to show different visualization of the compass data, and values to calibrate the compass. On the bottom

left is the mission and buttons to edit the mission by adding and deleting tasks or modifying existing tasks. Finally, there is a text box displaying status messages.

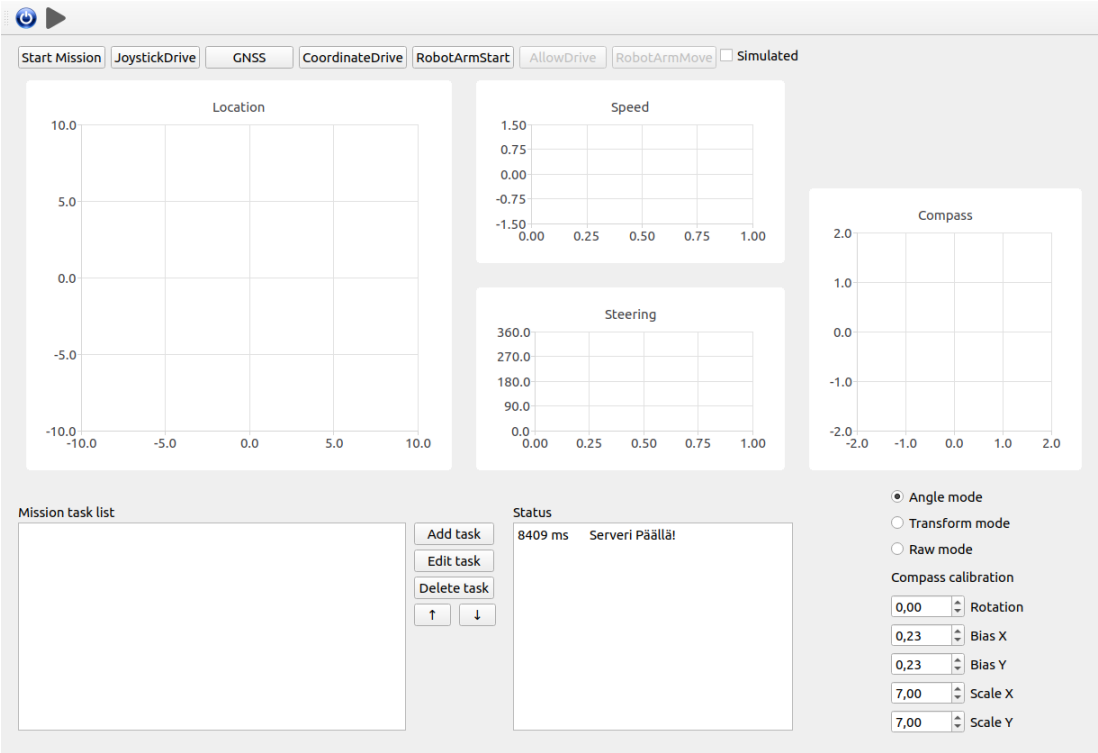

Figure 16. UI of the control software.

The user can monitor the execution of the mission and calibrate some of sensors from the UI. The UI also allows the manual operation of both the platform and the robot arm. With the “JoystickDrive” selected, the user can use arrow keys of the keyboard to control the movements of the platform. “CoordinateDrive” prompts the user to type GNSS coordinates or to select pre-programmed positions to navigate to. “RobotArmMove” gives a way to type out a pose for moving the arm to with either point-to-point or linear movements. “Add task” and “Edit task” are used to modify the mission. When clicked they open a prompt to select a task type and give the parameters for it. In the case of “Edit task”, the selected task in the mission task list will be modified. The same prompt is used for both adding and editing. “Delete task” is used to remove a task from the list. The up and down arrows are used to switch the positions of tasks in the list.

The compass side of the UI has different modes to display the data given by the sensor. The compass is affected by its surroundings. For example, big metal objects will twist the reading and it works slightly differently indoors and outdoors. For this reason, there are different display modes and calibration settings in the UI. Angle mode shows the compass reading as a point projected on a unit circle. Transform mode uses the calibration values to rotate, translate and scale the reading. Raw mode shows the data point as given by the sensor. The rotation is used to make sure that when the platform is pointing north, the reading will be in the top position. The bias values are used to translate the readings so that they are round the centre position. The scale values are used to make sure the distance of the reading from the centre is even in every direction.

7. APPLICATION USE CASE: WEEDING ROBOT

For the application of this thesis, a robotic system of automatic weeding was devised. A mobile robot platform with a 6-axis robot arm uses a combination of machine vision and compliant motion to pull weeds from the ground. The platform and equipment used are described in Chapter 4.

The full sequence of actions done by the robot are as follows: First, the robot receives a list of coordinates of the weeds in a field. Second, the robot navigates to the coordinates using a compass and its own location gained from a GNSS receiver. Third, the robot uses a machine vision system to detect the exact location of the weed. Fourth, the robot uses its robot arm to push the weeding tool into the ground at the required force needed. Fifth, the robot uses a force sensor to compliantly control the weeding tool to pull the weed out of the ground. The robot then repeats the steps until all the weeds in the list have been taken care of.

7.1. Mission Control

The goal for weed removal is given as weeding mission, which is set by the weed map of weed locations in the pasture field. The mission consists of tasks for moving the platform, detecting weeds, and performing the weeding action for all the weeds in sequence listed in the weed map. Each task is composed in a fixed manner of primitive actions like robot arm motions, image acquisition, and navigation actions. This multilayered control principle is illustrated in Figure 17.

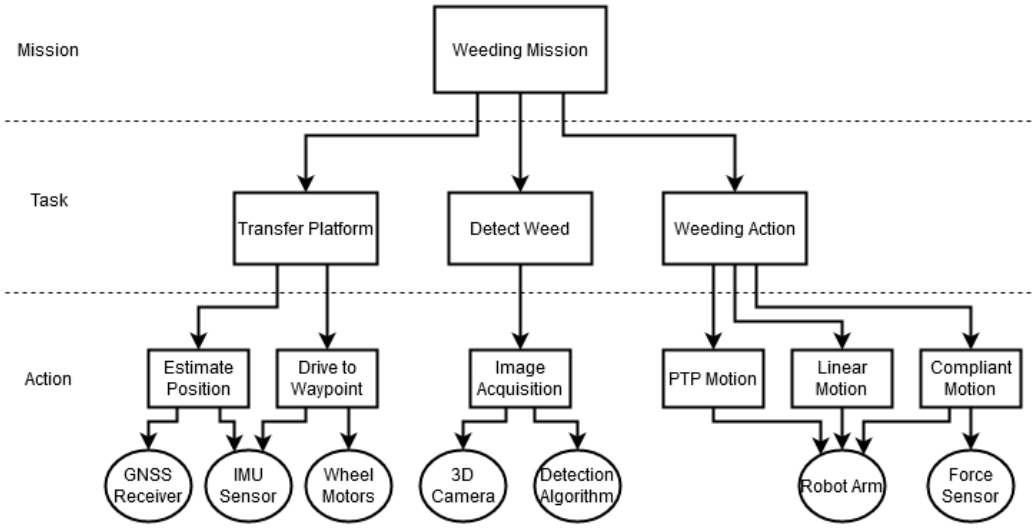

Figure 17. The flow of the mission from tasks to actions.

For each weed in the weed map there are three tasks to complete. First is to navigate the platform to the weed. This is performed by the transfer platform task, which in turn is made up of actual driving commands to the wheels and estimating the current position and comparing it to the next waypoint in the navigation path. Secondly, when the platform has reached the weed coordinates, the image acquisition task is carried out. In this task an image is taken with the 3D camera and the position of the weed is calculated. Finally, the weeding action -task is executed based on the weed position.

The weeding action includes point-to-point, linear, and compliant motions, with the inclusion of force sensing. With this method of mission control, it is simple to add new task-steps in the mission. It allows a more abstract way of programming the robot.

7.2. Navigation

The navigation system uses the Extended Kalman filter with GNSS receiver and a compass integration to estimate the position and orientation of the platform. It then calculates the direction it must move to reach the next waypoint. The waypoints are determined by the desired end position and orientation in such a way that there is a straight period before the last waypoint to make certain that the platform is pointing in the correct direction. The platform is quite slow to turn, so it is desirable to have a straight path to stabilize its movements (Figure 18).

The platform calculates the desired angular velocity based on the angle and the distance to the next waypoint in relation to the current position and orientation of the platform. The desired angular velocity is used as the set point and the gyroscope measured angular velocity is used to get the error signal for a PID controller. [19]

If the next waypoint is too much to either side of the platform, the speed is lowered so that turning is easier. The wheels of the platform have a very low maximum speed that prevents turning while moving forwards too fast.

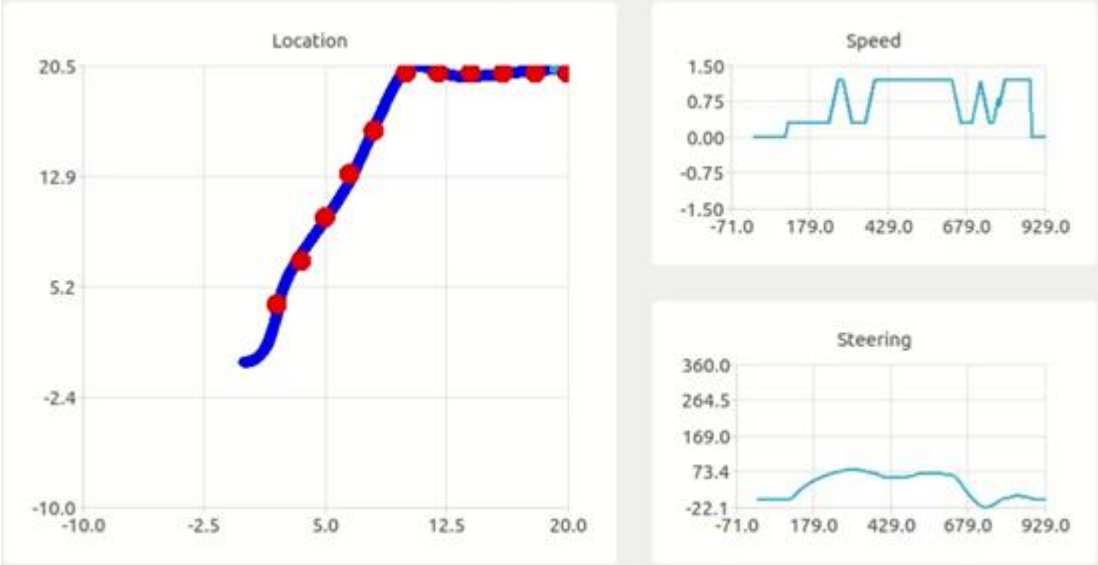

Figure 18. Robot UI with the current and past location, speed, and heading of the robot. Red points are the waypoints of the navigation path and blue line is the actual path, where the platform has moved.

The robot navigating in the field is shown in Figure 19. When repeating the same navigation path, there was slight drift towards south between attempts. This was caused by drift in the GNSS coordinate readings. The drift stabilization can take up to 30 minutes. After the readings stabilize the accuracy of the navigation is ±0.25 m of a specific point.

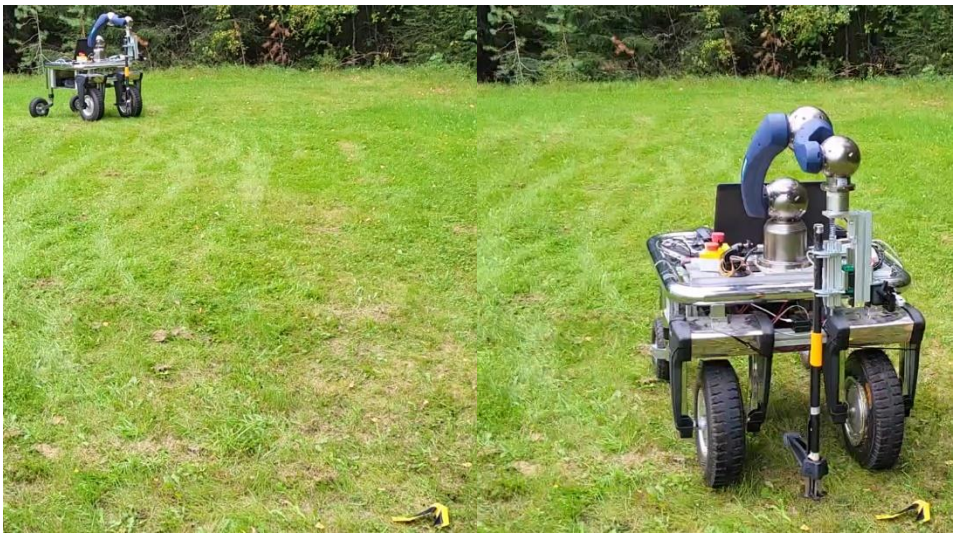

Figure 19. Robot navigating in the field.

7.3. Weed Detection

The weed detection works in 2D images by first filtering and thresholding the image to create a black and white image where white pixels correlate to leaves of the plant and black pixels mark the background. The white pixels that are connected are considered as potential segments. The segments need to be clustered and too small and too large segments are discarded. If a segment cannot be reasonably simplified as an ellipse it is also discarded. A principal component analysis is calculated for each leaf segment. The PCA is used to get the centre of the leaf and the largest eigen vector is used to create a line. The lines will gather around the approximate root location of the plant. Intersection points of the lines that are too far from other intersection points are likely outliers and are discarded. A centre point of the remaining intersection points is used as the root location. [20]

In the following Figure 20 is the result of the weed detection. The red lines are the directional lines of the segmented leaves, and the intersection points are marked white. The centre of the intersection points is marked light blue, which is very close to the actual root position of the weed.

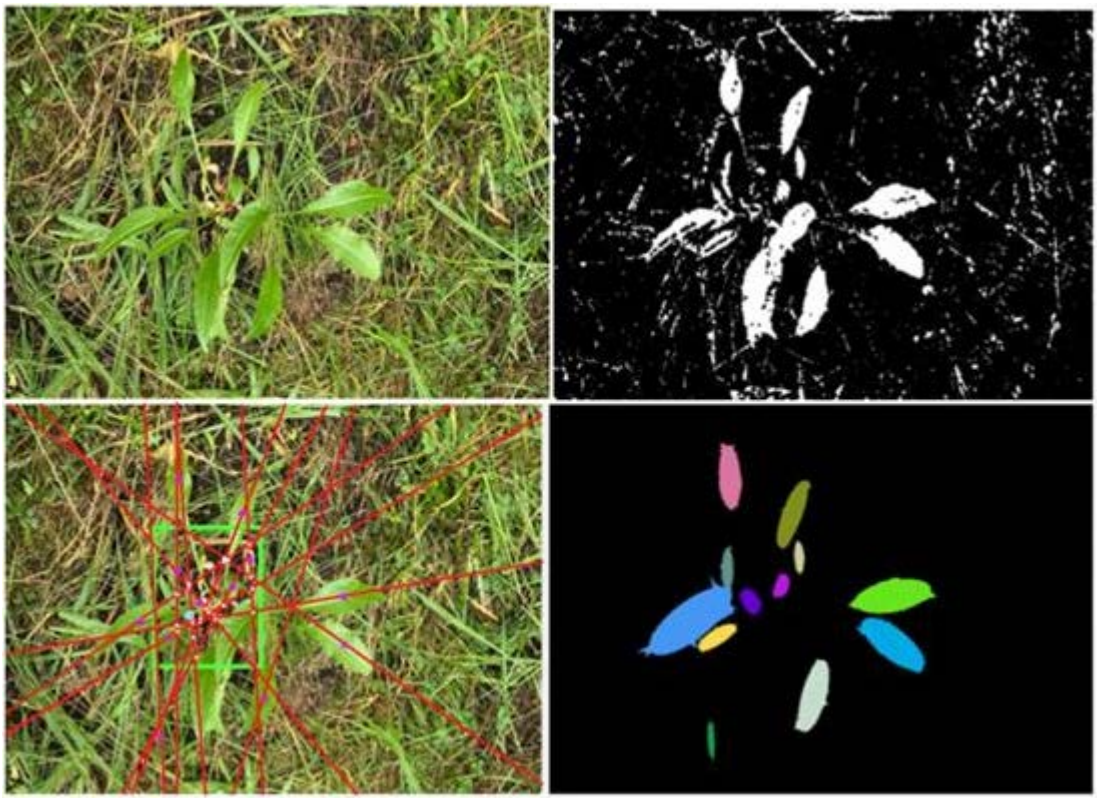

Figure 20. Detection of the weed with some of the processing steps. Top left is the original image, top right is the filtered image, bottom right is the segmented image, bottom left image has the detection result. Software by Niko Känsäköski.

7.4. Weeding Action

The robot arm is directed by specifying the position and rotation of its flange. To guide the tool to the weed's location, the weed's position in the robot arm's coordinate system is determined by tracing the transformations from the base to the flange, through the camera, to the weed. The inverse transformation from the flange to the tool yields the required flange position. Homogeneous transformations simplify the process, reducing it to a straightforward matrix multiplication.

A homogeneous matrix combines the rotation and translation into a single matrix. Multiplying homogeneous matrices together will give the transformation from one to the other. An inverse homogeneous matrix is the transformation backwards.

$$H = \begin{bmatrix} R_{11} & R_{12} & R_{13} & x \\ R_{21} & R_{22} & R_{23} & y \\ R_{31} & R_{32} & R_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{51}$$

- R_{ij} are the components of the rotation matrix.
- x, y, and z are the cartesian positions.

$$H = H_{flange}^{base} \times H_{camera}^{flange} \times H_{weed}^{camera} \times H_{tool}^{flange^{-1}} \tag{52}$$

- H_b^a are the homogenous transformation matrices between coordinate frames shown in figure 21.

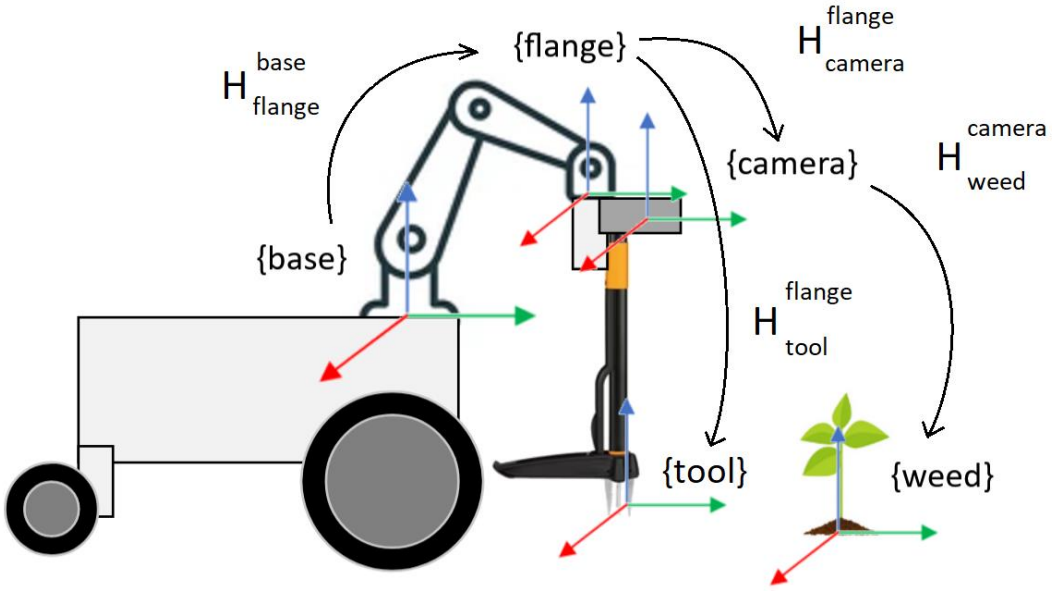

Figure 21. Robot coordinate transformation frames.

The motion of the weeding was planned by using a video of a person using the tool and taking pixel coordinates of certain points of interest in the tool and recording them. The force needed for ground penetration was calculated based on the compression of a spring with a known spring constant. Figures 22 and 23 show how the weeding action was investigated. In the background is a scale where each number represents 10 cm.

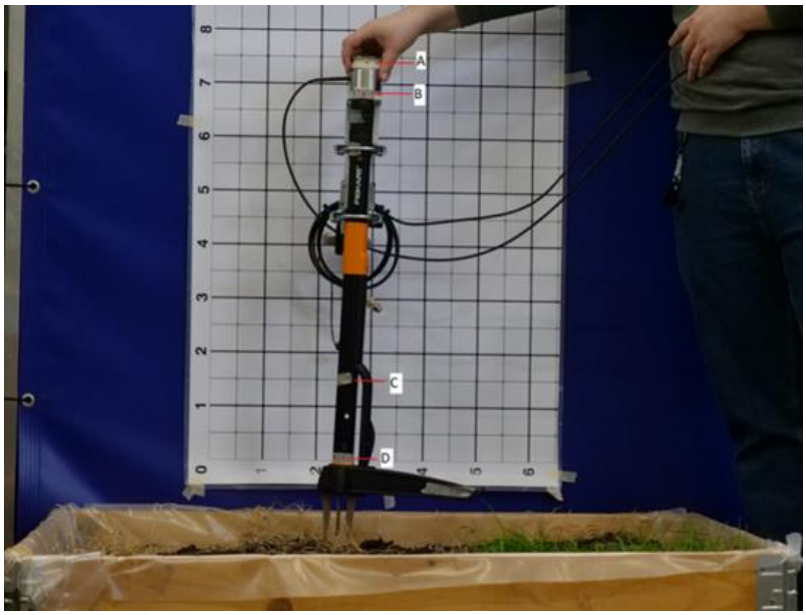

Figure 22. Screenshot of the video with points of interest marked.

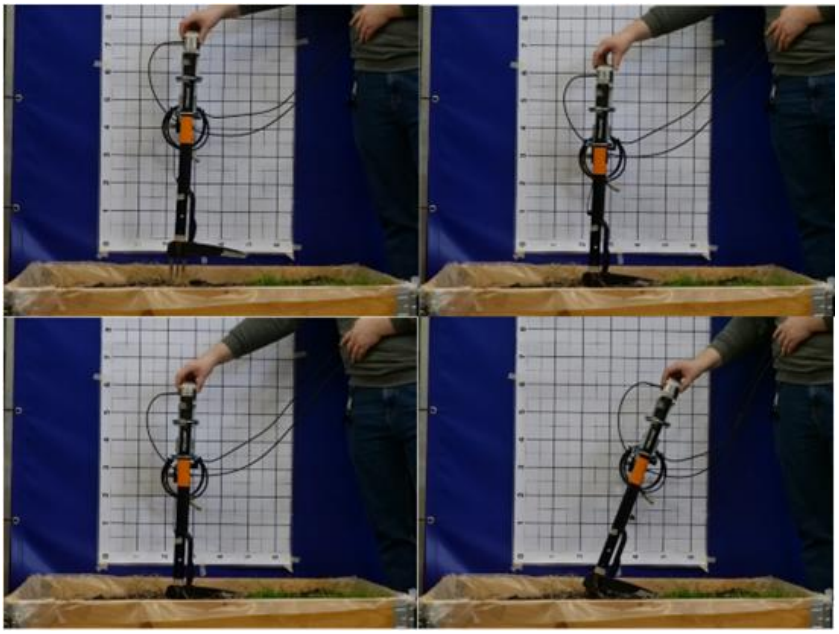

Figure 23. Set of 4 frames from the video where the weeding action is performed.

The recorded path of the person using the weeding tool was used as the path the robot arm should take in order to perform the weeding action. The path is shown in Figure 24 are the recorded points from the video. In the first part of the motion, the tool starts at position (0,0) and goes straight down by 9 cm to penetrate the ground. In the second part, the flange of the robot goes towards the robot (right on Figure 23 and Figure 24) for 28 cm and rotates simultaneously by 20° (Figure 25) to perform the pulling action using the lever arm at the bottom of the tool.

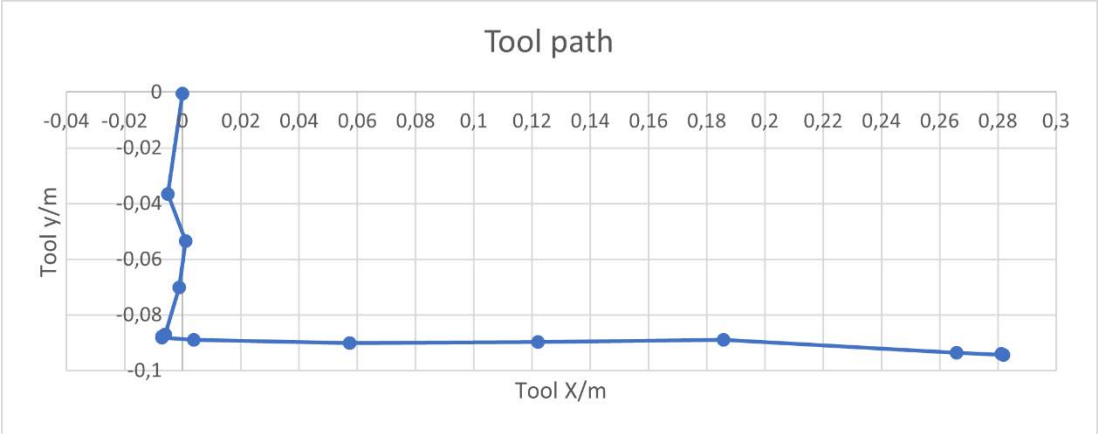

Figure 24. Position of point of interest A aka the flange position.

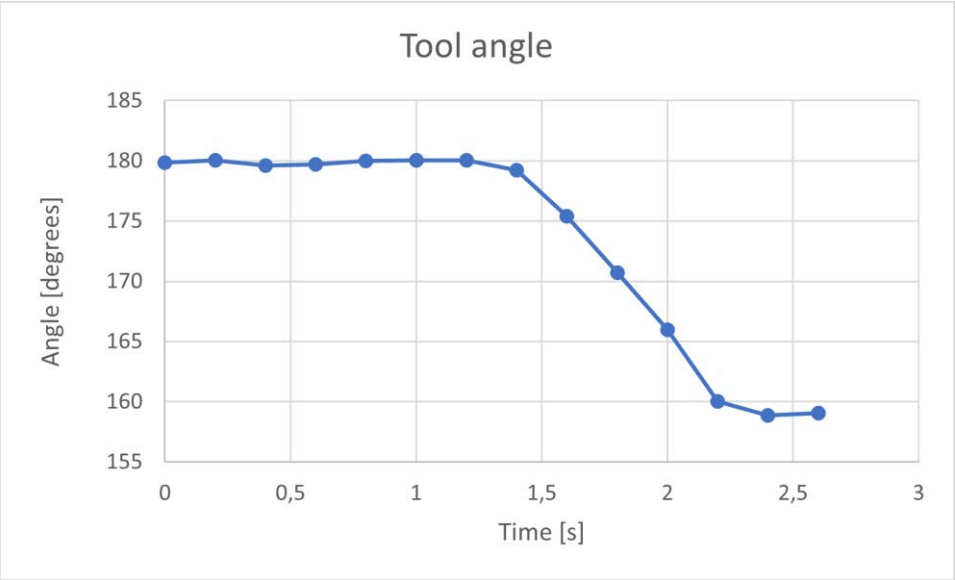

Figure 25. Angle of the tool over time. Calculated from the points of interest A and D.

The video of the weeding motion was also used to figure out the force needed for ground penetration and how the force evolves during the motion. Figure 26 shows how the spring compresses during the motion, and Figure 27 shows the force calculated based on the spring compression and spring constant from Chapter 5.7.

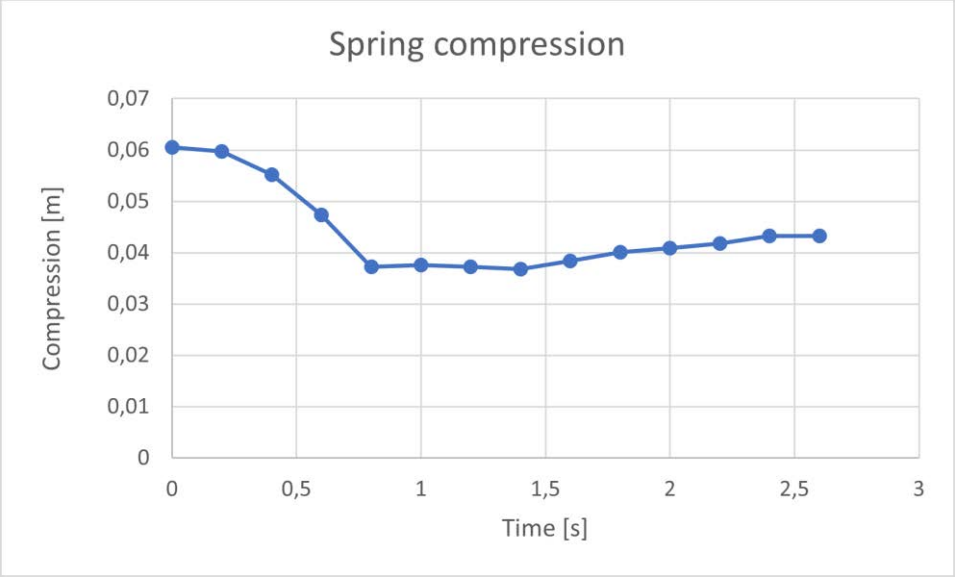

Figure 26. Compression of the spring over time. The compression is based on the distance of points of interest A and B.

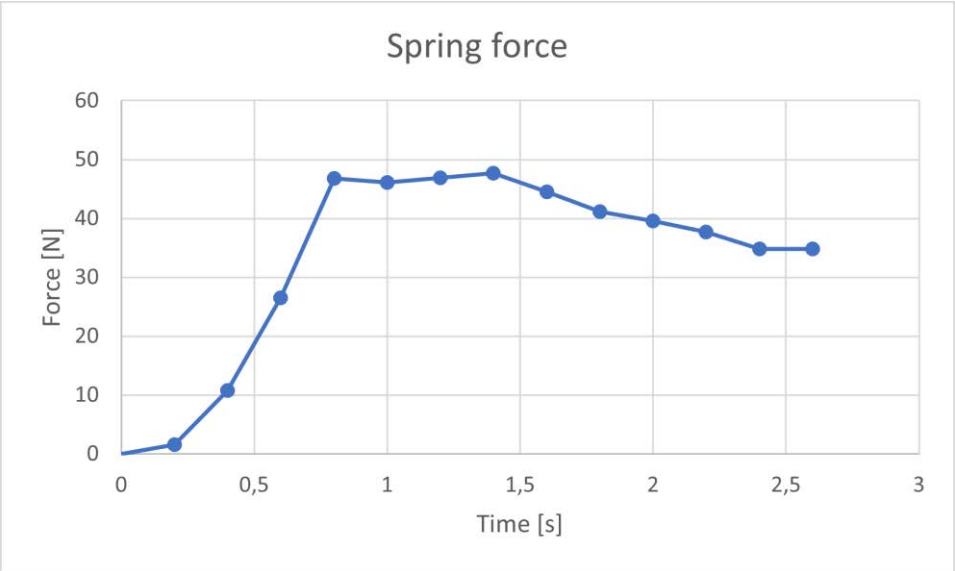

Figure 27. Calculated spring force based on the compression over time.

Based on the figures and graphs above, the robot needs to exert about 40 to 50 Newtons of force when penetrating the ground and then hold the force steady when turning the tool for the weeding action. In Figure 27 the force goes down during the rotation part of the motion but in the end, it was decided that a constant force would be simpler for keeping the lever arm of the tool firmly pushed to the ground.

7.5. Force Control

The force control is used to maintain the contact force of the tool with the ground. This is done to ensure that the tool uses the lever action properly. Compliant motions are split into two motions: guarded and unguarded motions. In the guarded motion, the arm is lowered towards the ground at a constant velocity, until a specified force is detected. The distance lowered is saved as ΔX_{guard}. In the unguarded motion the set motion path of the weeding action is followed with the addition of ΔX_{guard}.

The contact force is maintained by a proportional control, where the desired force is the set point, and a filtered measured force signal is used to get the error signal for the controller. The measured force is filtered using a low pass 3rd order Butterworth filter with a cut-off frequency of 10 Hz. The output of the P-controller is another distance ΔX_f which is added to the modified path ΔX_{path}. ΔX_f is in the direction of the z-axis of the tool, which is also the axis where the force is measured. The modified and force-controlled path is fed into the inverse differential kinematics in order to get the next motion increment. The functionality of the force controller is described as a block diagram in Figure 28.

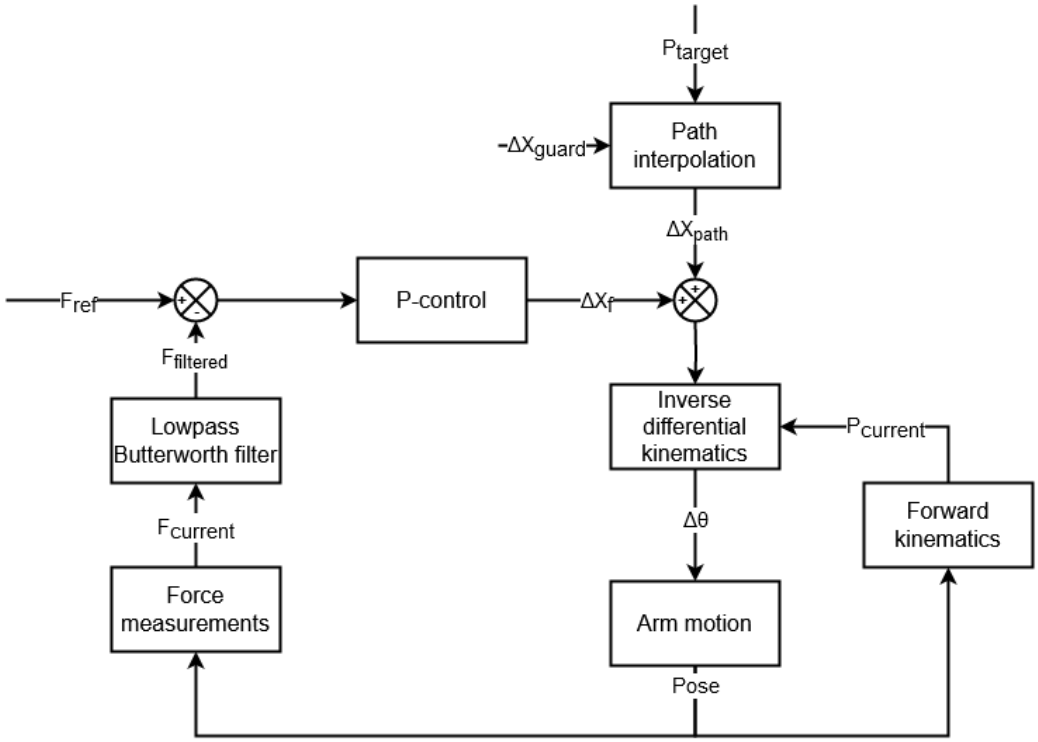

Figure 28. Block diagram of the force control system.

The force control is tested with the following set up. Attached to the flange of the robot arm is a wheel in a spring-loaded track seen in Figure 29. The spring-loaded track is attached to the wheel to allow slower changes in the forces as mentioned in Chapter 5.7. The guarded motion force limit is set to 40 Newtons and in the unguarded motion the same force is desired. In the guarded motion the arm is lowered towards a flat surface, where the wheel will touch, until the force limit is reached. Next, the arm will follow a sideways path while maintaining the desired force. The wheel spins on the surface to minimize friction. After the motion is done, the arm moves away from the surface.

Figure 29. Force control test with a spring-loaded track and a wheel attached to the robot flange.

In the following Figure 30 is the forces measured in the test. Initially the force increases fast when the wheel touches the surface. Then, there is a delay before the next movement is triggered, where the force is constant. When the unguarded motion is executing, there is an oscillation of the force. After the motion is complete, there is another delay before the arm moves away from the surface. The delays are not part of the automatic execution of the compliant motion but are caused by the manual execution of the motion steps of the test.

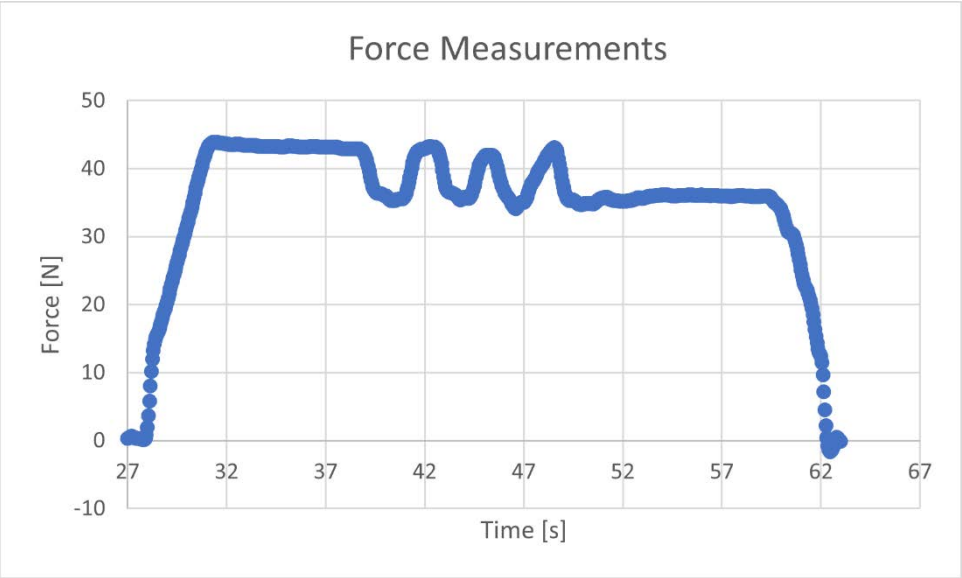

Figure 30. A section of force measurements of a force-controlled movement test. The desired force is set to 40 N. From 27 to 31 seconds is the guarded motion, 31 to 39 seconds waiting until the next movement, 39 to 49 seconds un-guarded motion, 49 to 59 seconds waiting again for the next movement, 59 to 62 seconds release motion away from the surface.

The previous test demonstrates that the force controller is functional. It does not hold the contact force steady, but rather oscillates ±5 N around the set force. The oscillation is probably caused by the P-control part of the force controller in addition to the delay from the Butterworth filter. The guarded motion also overshoots the desired force by about 5 newtons. After the force-controlled movement is finished, the force is lower than desired but that is because the motion stopped while the oscillation was on its lowest point.

7.6. Workflow Demonstration

In this chapter the demonstration of the final workflow is described. The workflow sequence is as follows: First, the robot navigates to the weed position in the field (Figure 31). Second, the robot uses its camera to take a picture of the ground (Figure 32). Third, the machine vision system calculates the root position of the weed and projects the pixel coordinate to the depth image (Figure 33). Finally, the 3D position is used to calculate the tool path and compliant motion control is used during the process to ensure the tool remains in contact with the ground. After the weed has been pulled from the ground (Figure 34), the workflow sequence repeats.

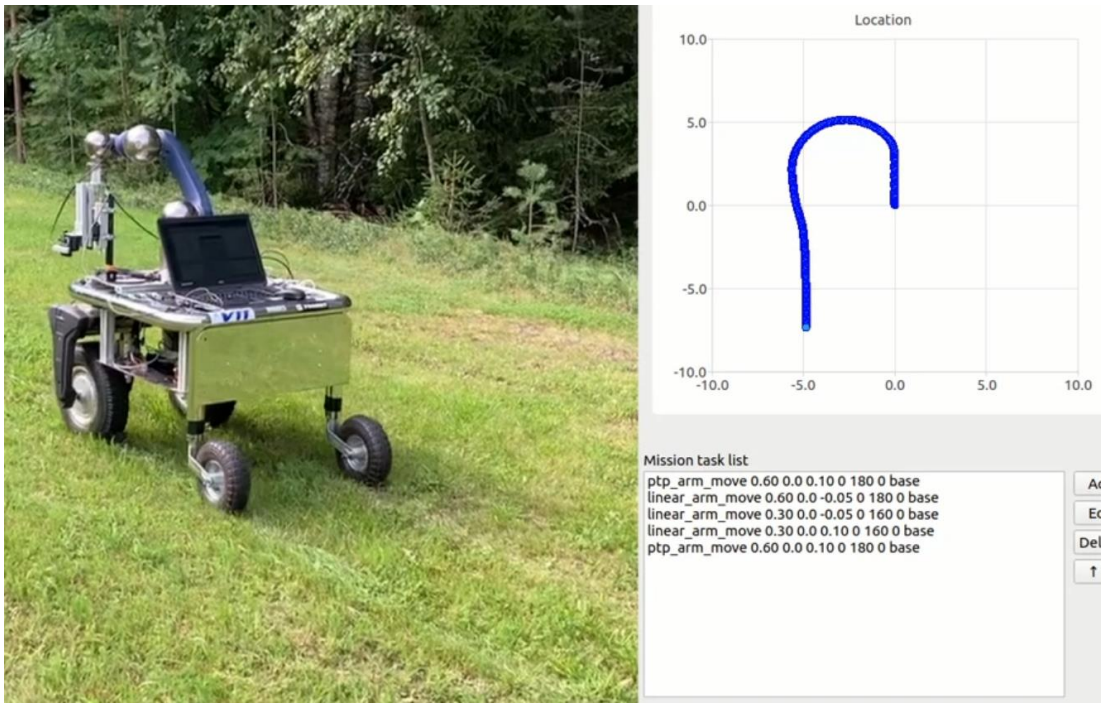

Figure 31. Robot navigating autonomously to the weed position in the field.

The robot navigating in the field can reach up to ±0.25 m accuracy of a specific point. The path the robot has taken is drawn in the UI. The navigation points are the weeds located roughly by separate means like by drone. In this demonstration case, the weed locations' coordinates were taken by moving the GNSS receiver to the weed's location and recording the coordinates.


Figure 32. Detecting the weed on the ground. Red is the weed and blue is the camera.

The accuracy of the weed detection and the following robot arm movement was good enough that the jaws of the weeding tool could grab the root of the plant. Figure 34 shows the calculated root location of the weed.

Figure 33. Root position of the weed found. Blue dot is the calculated root location.

The weeds in the field were grown in a pot indoors and planted outside for this demonstration. The weed detector had some issues locating the weed occasionally because it kept mistaking the fallen leaves as the leaves of the weed. Cleaning the surroundings of extraneous leaves helped the system in detecting them greatly.

Figure 34. Pulling the weed from the ground with the weeding tool.

When pulling the weed from the ground, the robot pulled the whole root of the plant out, therefore preventing the weed from regrowing at the same spot next year. This demonstration showed the potential of this technology in robotic mechanical weeding.

8. CONCLUSION

The objective was to develop a kinematic model and control system for compliant motion of a robot arm on a mobile platform with navigational capabilities for outdoors field work. Forward, inverse, and differential kinematics were developed for a Schunk LWA4P robot arm with a proportional force controller to maintain contact forces during linear motions. Navigation systems using extended Kalman filter with sensor integration for positional estimation, and coordinate navigation with path planning were developed. A robot that can execute pre-planned missions, where it navigates to coordinate positions, detects weeds, and picks them up, was created. In terms of the stated objectives this project was a success.

For the development of the weeding robot, there were some challenges mainly in the hardware side of things. The robot arm had issues right at the beginning which got worse over time. Since Schunk stopped any maintenance to their robot arms a few years ago, there were difficulties in fixing problems on it. Using most of the maintenance software that were used by Schunk's technicians had to be studied to fix issues. Figuring out which error message corresponded with each issue was extremely helpful but receiving a following error and a thermal load error did not initially explain the fault. Another major problem with the robot arm showed up when the weather was not warm enough. The arm could not handle outside temperatures well.

The problems persisted with broken tyres, electronics failures, too bright sunlight, and multiple redesigns but eventually through much hardship, a working system was created. In the end any trouble was just a challenge to overcome and taught me new problem-solving skills. This project was an excellent exercise in creative problem solving and a great reason to get into the fine details of robot kinematics.

In experiments the platform achieved ±0.25 m positional accuracy when given a specific location to move to. The accuracy is mostly based on the accuracy of the GNSS receiver. With a proper ground station, the accuracy could be close to ±0.05 m When detecting a weed, it achieved ±0.02 m accuracy of the root location. Moving the weeding tool, it achieved ±0.005 m accuracy that covers hand-eye and tool calibration inaccuracies. The needed force for weed removal is depended on the hardness on the ground between 50 to 150 N, but the robot arm could only reach 50 N, which is enough in loose soil.

9. REFERENCES

[1] Leonardo Enrique Solaque Guzmán et. al., Weed-removal system based on artificial vision and movement planning by A* and RRT techniques. Acta Sci., Agron. 41, 2019. DOI: https://doi.org/10.4025/actasciagron.v41i1.42687

[2] R. Aravind, M. Daman and B. S. Kariyappa, "Design and development of automatic weed detection and smart herbicide sprayer robot," 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS), 2015, pp. 257-261, DOI: 10.1109/RAICS.2015.7488424

[3] Rene Koerhuis, Odd.Bot robot takes on herbicide free weed elimination. Future Farming, December 20, 2018. URL: https://www.futurefarming.com/Tools-data/Articles/2018/12/OddBot-robot-takes-on-herbicide-free-weed-elimination-375027E. Accessed 17.4.2024.

[4] Robot One, Pixel Farming Robotics, URL: https://pixelfarmingrobotics.com/robot-one/. Accessed 18.4.2024.

[5] Siciliano B, Sciavicco L, Villani L & Oriolo G (2009) Robotics Modelling, Planning and Control. London. Springer-Verlag.

[6] Hatledal L (2017) DH Table Visualization. URL: https://org.ntnu.no/intelligentsystemslab/course/DH/index.html. Accessed 1.4.2024.

[7] Bonev I. What are Singularities in a Six-Axis Robot Arm. URL: https://www.mecademic.com/academic_articles/singularities-6-axis-robot-arm/. Accessed 19.3.2024

[8] Wang, W., Loh, R.N.K. and Gu, E.Y. (1998) "Passive compliance versus active compliance in robot-based automated assembly systems", *Industrial Robot*, Vol. 25 No. 1, pp. 48-57. DOI: https://doi.org/10.1108/01439919810196964

[9] Sadun, A.S. & Jalani, Jamaludin & Sukor, J.A. (2016) An overview of active compliance control for a robotic hand. 11. 11872-11876. URL: https://www.researchgate.net/publication/310511186_An_overview_of_active_compliance_control_for_a_robotic_hand. Accessed 17.4.2024

[10] Siciliano B, Khatib O (2008) Springer Handbook of Robotics. Berlin. Springer-Verlag.

[11] Ransom M. Transform Longitude and Latitude into Meters. URL: https://stackoverflow.com/questions/3024404/transform-longitude-latitude-into-meters. Accessed 19.3.2024.

[12] Sakai A. Extended Kalman Filter Localization. URL: https://atsushisakai.github.io/PythonRobotics/modules/localization/extended_kalman_filter_localization_files/extended_kalman_filter_localization.html. Accessed 19.03.2024.

[13] Vähäkuopus P (2021) Mobiilirobotin robottikäsivarren liikeohjaus. Bachelor's thesis. OAMK.

[14] Memar A & Esfahani E (2015) Modeling and Dynamic Parameter Identification of the SCHUNK Powerball Robotic Arm. ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Boston, USA. DOI: https://doi.org/10.1115/DETC2015-47703.

[15] Wetz S & Kabelitz H (2023) Multicomponent Sensor K6D / F6D / K3R Instruction manual. URL: https://www.me-systeme.de/produkte/sensoren/kxd/k6d/anleitungen/ba-k6d_en.pdf. Accessed 1.4.2024.

[16] Lourenço F & Araujo H (2021) Intel RealSense SR305, D415 and L515: Experimental Evaluation and Comparison of Depth Estimation. In VISIGRAPP (4: VISAPP) (pp. 362-369). URL: https://www.researchgate.net/publication/349384298_Intel_RealSense_SR305_D415_and_L515_Experimental_Evaluation_and_Comparison_of_Depth_Estimation. Accessed 17.4.2024.

[17] Farooq, O., Mubeen, K., Ali, H.H., Ahmad, S. (2019). Non-chemical Weed Management for Field Crops. In: Hasanuzzaman, M. (eds) Agronomic Crops. Springer, Singapore. DOI: https://doi.org/10.1007/978-981-32-9783-8_16

[18] PPP-RTK GNSS correction services (PointPerfect). URL: https://www.u-blox.com/en/technologies/ppp-rtk-gnss-correction-services-pointperfect. Accessed 17.4.2024.

[19] Kotaniemi J (2018) Mobiilirobotin navigointimenetelmän parantaminen odometrian avulla. Bachelor's thesis. OAMK.

[20] Känsäkoski N, Heikkilä T & Kotaniemi J (2022) Detection and Localizing of Rumex Seedlings for Robotic Weeding, 2022 18th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), Taipei, Taiwan, 2022, pp. 1-7, DOI: https://doi.org/10.1109/MESA55290.2022.10004464