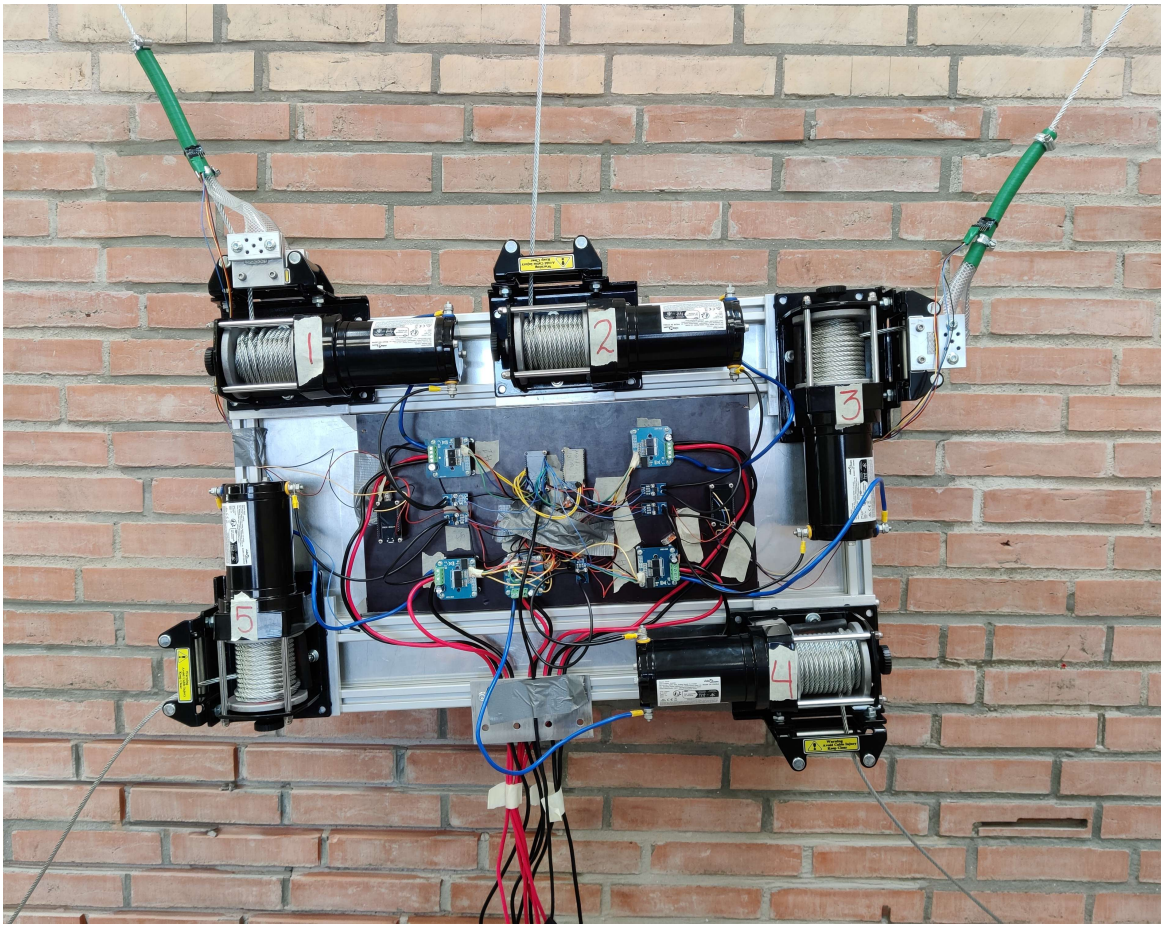
P6 Bachelor Project

Site-Tech mortar removal robot



Bachelor Project Report

P6 ROB6 B661

Aalborg University
The Technological Faculty for IT and Design

The Technological Faculty for IT and Design
Niels Jernes Vej 10, 9220 Aalborg Øst
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
P6 Project - Robots in an Application Context

Theme:
Technological Project Work

Project Period:
ROB P6 6. Semester

Project Group:
B661

Participant(s):
Wallat Bilal
Christoffer Johan Soos
Frederik Saldern Nielsen
Simon Haaning Andersen
Glenn Gadesgaard Svendsen

Supervisor(s):
Jan Dimon Bendtsen

Copies: 1

Page Numbers: 92

Date of Completion:
May 25, 2023

Abstract:

Manual removal of mortar between bricks is a laborious and physically demanding task. To alleviate the challenges faced by workers and enhance efficiency, Site-tech has developed an automatic mortar removal robot. While this solution can autonomously remove mortar within a specified area, manual motor actuation is still required for the robot to relocate to a new area, using their developed wire system.
The objective of this project is to autonomize the repositioning system for the Site-Tech robot. To accomplish this, a motion control system has been developed. This system encompasses motion planning and dynamic calculations, incorporating a torque-controlled motor control system that enables the robot to reposition itself. The effectiveness of the system has been evaluated using a scaled-down model of the Site-Tech robot.
Although the developed system did not fully meet all the requirements during testing, it serves as a proof of concept for an autonomous solution that can be further refined and implemented in full-scale robots operated by Site-Tech.

Preface

Aalborg University, May 25, 2023

Christoffer Johan Soos
csoo20@student.aau.dk

Simon Haaning Andersen
san20@student.aau.dk

Glenn Gadensgaard Svendsen
gsvend20@student.aau.dk

Frederik Saldern Nielsen
fsni20@student.aau.dk

Wallat Bilal
wbilal18@student.aau.dk

Contents

Acronyms

AAU Aalborg University. 29, 61

ADC Analogue to Digital Converter. 46, 47

COM Center of Mass. 13, 18, 19, 20, 21, 26, 27, 39, 56, 57, 59, 60, 92

DAC Digital to Analogue Converter. 47

DC Duty Cycle. 29, 31, 37, 46, 48, 49

EMF Electromotive Force. 29, 30, 48, 61

I/O Input and Output. 46

IMU Inertial Measurement Unit. 19, 21, 33, 34, 36, 38, 39, 40, 59, 61

LUT Look-up Table. 47

PID Proportional–Integral–Derivative. 30, 31, 50, 57

PWM Pulse-width Modulation. 29, 30, 31, 37, 38, 46, 48, 49, 50

TCP Transmission Control Protocol. 36, 38, 40, 41

1 Introduction

Maintaining the integrity of the outer walls of buildings is important to ensure their longevity and safety, the mortar between bricks needs to be renewed every 30-50 years. Currently, this is done manually, but it is a physically demanding task, requiring workers to use handheld tools, doing this for multiple hours at a time can take a toll on their health and well-being. The company Site-Tech is developing a mobile platform to help automate the process.

By using this robotic solution, it is possible to transition from physically demanding work to a primarily supervisory position. This change enables a single worker to guide multiple robots at a time, which improves the worker's efficiency.

The platform utilises computer vision and machine learning to register the mortar. A gantry system inside the robot is capable of automatically removing the detected mortar.

The platform is moved along a wired system designed and developed by Site-Tech. The wire system allows the robot to reach a larger area of the wall without increasing the physical size of the system. The current wire system requires manual input to reposition the robot, which can be a time-consuming task. Site-Tech has requested the wiring system to be more autonomous while keeping the cost of sensors as low as possible.

1.1 Problem Statement

"How can the prototype mortar removal robot developed by Site-Tech be improved to allow for autonomous repositioning on the wall, using the wire system proposed by Site-Tech?"

2 Current system

The system developed by Site-Tech contains various elements, including elements for moving the system, registering mortar, removal of mortar, together with a frame for holding everything together. For improving a working prototype, it is important to get some information on the current system proposed by Site-Tech.

2.1 Gantry system

Inside the frame, a gantry setup is used to hold the equipment for mortar removal. The equipment includes a camera with a corresponding light system, an angle grinder with a hardened disc, and two motors for controlling the rotation and translation of the grinder. By having the camera mounted next to the grinder, as visualised in Figure 2.1, it is possible to know the local position of the mortar relative to the grinder. However, due to the camera being placed on the right side of the grinder, it is only possible to remove the mortar in one direction.

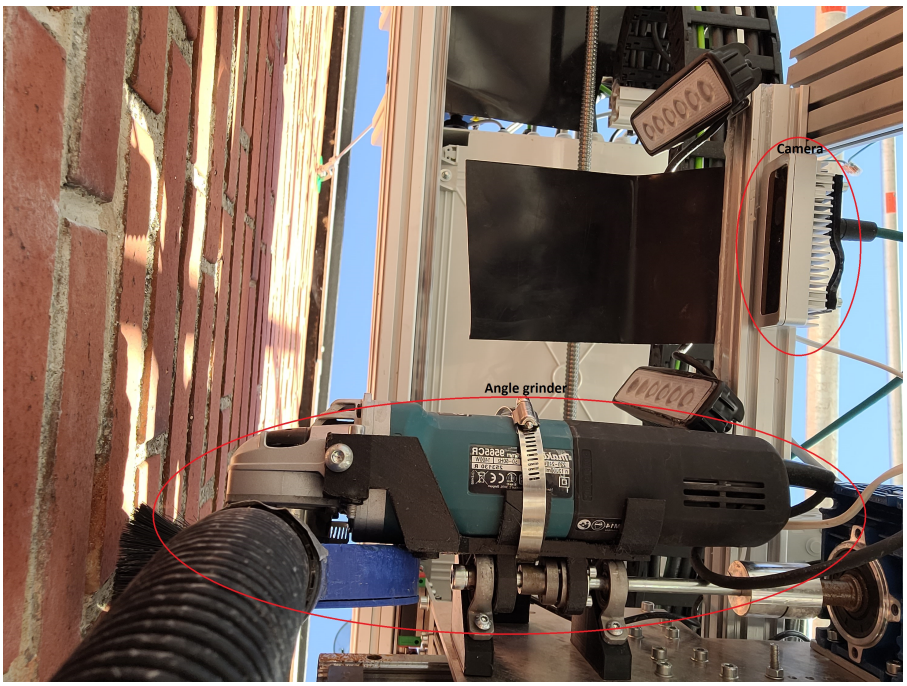


Figure 2.1: Placement of the camera and the angle grinder on the gantry base

When the system removes mortar, the camera is used to determine the position and orientation of bricks relative to the gantry. In Figure 2.2, the digital representation of the local placement of the bricks and grinder can be seen. This allows the system to compensate for inaccuracies in frame position and orientation. When the local image is made, the system creates a path using the depth of

the mortar which needs to be removed and the optimal placement of the grinder in both the X, Y and Z axis, and rotational Z axis, as seen in Figure 2.4.

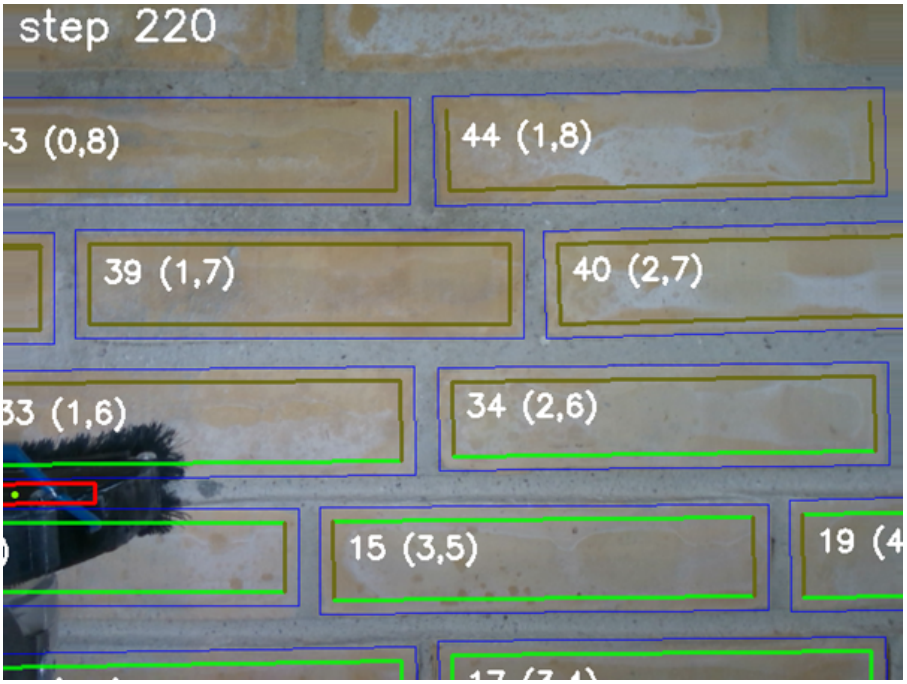


Figure 2.2: Digital representation of the grinder and brick's position and orientation on a local map of the wall

When running the program, the gantry starts in the bottom left corner of the frame and moves in the direction to the right, when it reaches the utmost right corner of the workspace, the gantry moves up, then goes back to the left, and does the same movement again. This is repeated until the robot has arrived in the top right corner, removing mortar from one horizontal line at a time which corresponds to one brick. When one horizontal line is removed, the system removes the vertical lines between the current line and the preceding line and moves on. The bricks have a length of 23cm and a height of approximately 5cm, as seen in Figure 2.3.

Figure 2.3: Image of an approximation on the height of a brick

2.2 Tool holder

To remove mortar, Site-Tech has made a tool holding base. This base contains a motor for linear movement, which is perpendicular to the wall. They also implemented a motor for tool rotations, which can rotate around the same axis, visualised in Figure 2.4.

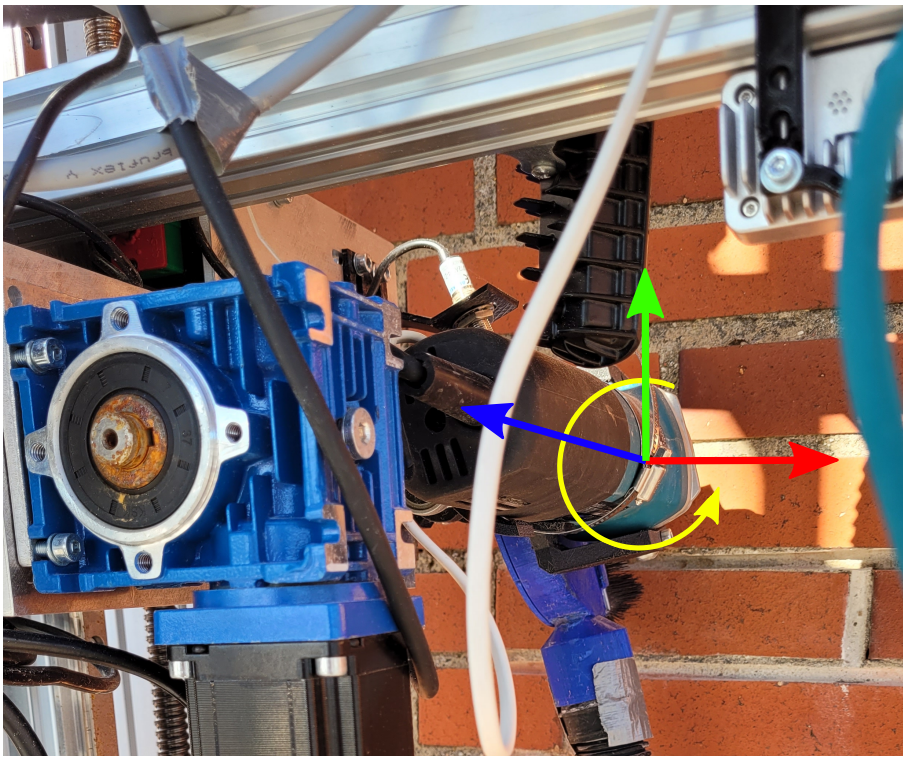


Figure 2.4: Image of the tool holder. The yellow arrow indicates angular rotation, and the blue indicate linear movement perpendicular to the wall

Before they implemented this setup, they used a spindle motor, but due to difficulties cutting hard mortar, they had to use a stronger grinder, so they changed to an angle grinder, which allowed the robot to grind harder mortar. The angle motor also has the ability to move both horizontally and vertically, which allows the system to compensate for small errors in the orientation of the robot. Site-Tech has said the tool holder can compensate for deviations in the rotation around Z, of around 7.5deg in each direction around the robot centre.

2.3 Wire system

To ensure the prototype's mobility and stability on the walls, five wires with five anchor-points are used. The anchor-points are mounted inside of the wall at predefined places, as seen in Figure 2.6.
4 of the 5 anchor points are placed in the corners of the desired workspace, and the fifth wire is placed in the top centre of the workspace, as visualised in Figure 2.5. By utilising this setup, it is possible to

reposition the robot around the wall by adjusting the torque of the motors which changes the tension of the wires. Since each motor only can exert force on the frame in the direction of the wire, it is necessary to know the wire directions. Using this principle, it is also possible to adjust the workspace while the robot is attached to the wall, where it is possible to loosen one of the wires and move the anchor point, which gives the system the ability to relocate without it being taken down.

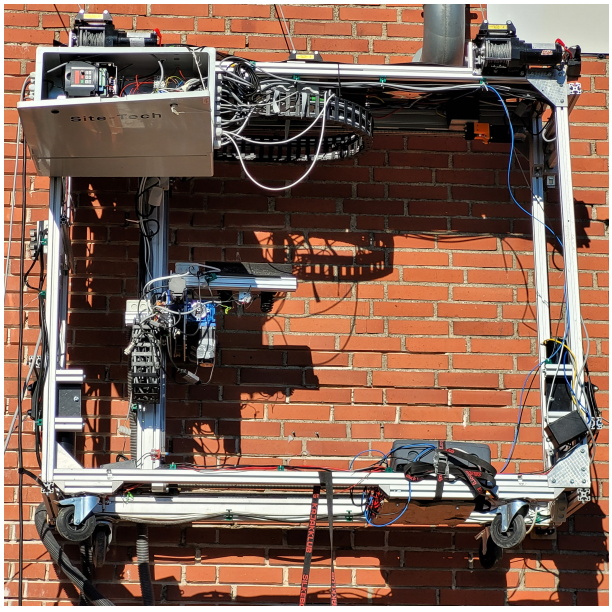


Figure 2.5: Image of the full system

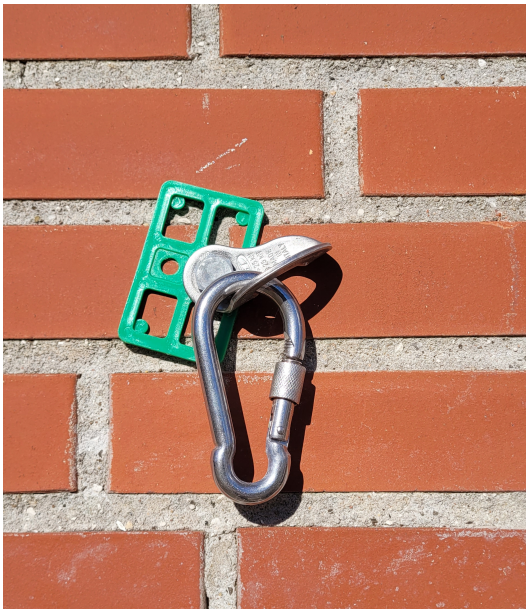


Figure 2.6: Image of an anchor point on a wall

2.4 Static friction

The robot utilises static friction in order to prevent sliding along the wall while removing mortar, this is done with four wood blocks each placed at the end of a support beam, as visualised in Figure 2.7. By applying force on the wooden block using the tightened wires, the frame is able to maintain a fixed position.

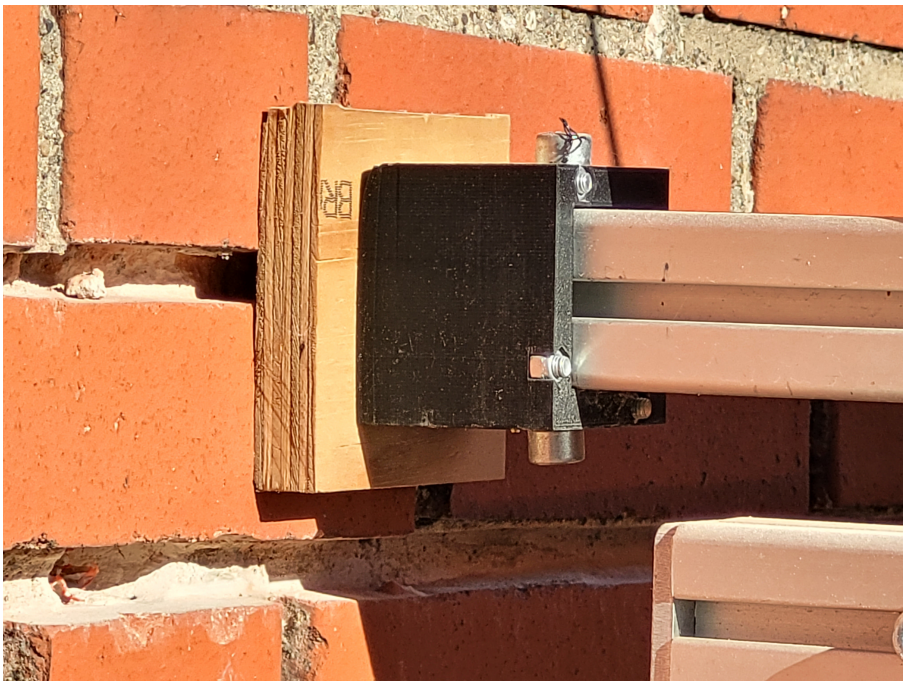


Figure 2.7: Image of the wooden block used to give a valid surface for static friction

When the systems need to be re-positioned, the frame is pushed away from the wall by activating four linear actuators positioned on the sides of the frame, as visualised in Figure 2.8. When the actuator is activated, the wheel is pressed on the wall and the force applied through the wooden block is removed.

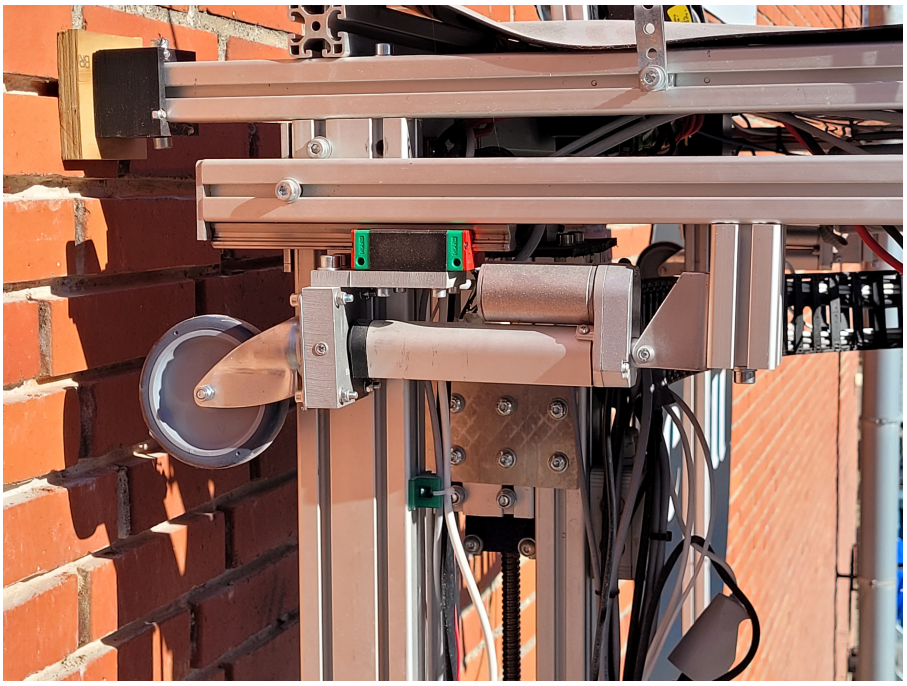


Figure 2.8: Linear actuator for pushing the system off the wall

2.5 Control system

After the gantry system has removed all reachable mortar in its workspace, the setup is then manually moved to the next area, because the current system does not have a control system implemented for automatically moving the robot. A handheld controller, as seen in Figure 2.9, is currently being used to move the robot. The controller is used to give inputs to the 5 motors which can loosen and tighten the wires depending on where the robot needs to be. This can be a slow process due to the manual task of adjusting the wires individually. This is not only less precise due to the employee spending a lot of time manoeuvring the robot.

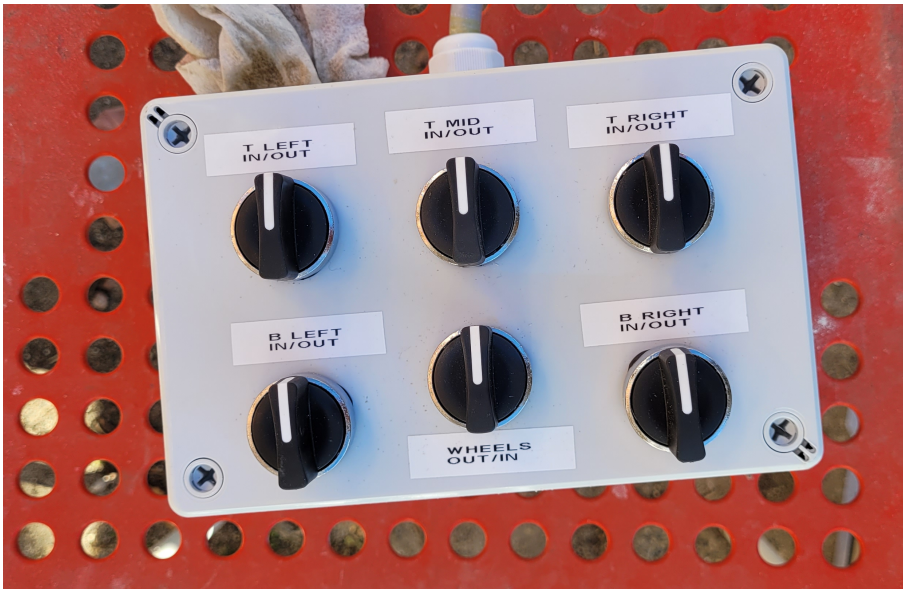


Figure 2.9: The 6-button handheld controller

2.6 Current system limitations

The current setup has its own limitations arising from some challenges on the different parts of their system. Some of these challenges and limitations will be mentioned and addressed in the following paragraphs.

The current system is currently manually controlled by a worker who has the necessary information to adjust the frame. Automating Site-Tech's solution would require information about the system in order to control the orientation and position of the frame. This can be done through different sensors, which allows the system to move automatically to a desired position. This must be done using as cost-effective sensors as possible, where durability is important due to the work environment the robot is operating in. It must be able to withstand all weather conditions and different hazards, such as dust, due to its working outside for multiple weeks at a time.

Wire system

The wires are currently controlled using a simple controller as seen in Figure 2.9 without sensors for registering the angle and torque of the winch motors. Therefore Site-Tech want a system that enables the user to control the rotation and the position of the system on the wall using as few sensors as possible. Over time the wires, motors and the component used for moving the system on the wall will be worn, and the damaged part must be identified and replaced. Together with this, Site-Tech would plausible solution to detect a faulty wire.

Batteries

Site-Tech has integrated batteries into their solution, the wire motors are connected to the batteries, making them function both online and offline. The other parts of the solution require concurrent current in order to function, so the solution is not made to function offline, but the goal is to make sure the wire motors do not trip a fuse because they have the ability to draw too much current from the circuit. The batteries ensure this does not happen. The batteries recharge whenever the wire system motors are inactive, which means they will have enough charge whenever the motors activate again.

2.7 Conclusion

As described in Section 2, the full system is able to remove mortar in a limited space, there the possibility for a more autonomous method for moving the system on the wall is limited and require more manual work and manual computation than needed. For a working solution to be made, some requirement must be made for making the solution usefully and testable. These requirement will be presented in the following chapter.

3 Requirements

To address the issues facing the mobile robotic platform, it is important to establish some requirements for the design and implementation. The requirements are derived from an analysis of the Site-Tech setup and the problem statement mentioned in Chapter 1, some of the identified problems and limitations mentioned in Section 2.6. The requirements section's goal is to showcase the relevant criteria that must be solved or met in order to reach a fulfilling solution.

Table 3.1 describe the requirements for the physical system, and its sensors.

Requirements			
	Requirement	**Measurement**	**Section**
1.1	The robot must accurately move to its target on the wall, deviating at most 0.1m when it completes its motion. Based on the height of 2 bricks, a line mortar is always visible.	m	2.3;2.6
1.2	When the robot reaches the goal location, the frame orientation must not deviate more than 7.5deg from the horizontal level.	deg(°)	2.2;2.6
1.3	The robot's trajectory should follow a predictable pattern, moving towards the designated position.		2.3;2.6
1.4	The robot must be able to Calculate its current location/position with an accuracy of 0.1m and rotation with an accuracy of 7.5deg, relative to placed anchor points.	m & deg(°)	2.2;2.6

Table 3.1: Requirements

3.1 Delimitation

The scope and limitations of the project will be outlined in this section, recognising it is not possible to address all of the issues facing Site-Tech within the time constraints of one semester. To ensure a focused and attainable outcome, the project will specifically concentrate on developing the autonomous movement capabilities of the prototype mortar removal robot to reach desired positions within the workspace.

3.1.1 Brick-wall for Test

There are some limitations on the wall itself, the wall where the tests will be performed is a fixed size, with fixed anchor points, which means it is only possible to gather data from this specific wall, with

the specific anchor points placements. This means, if the system works on this wall, and it passes all the testing, it is still not guaranteed it will work on other walls, therefore Site-Tech would have to do some testing on different-sized walls, with different anchor points to make sure it functions as expected. The wall itself is 3.35x4m with anchor points at predefined positions as can be seen in Figure 3.1.

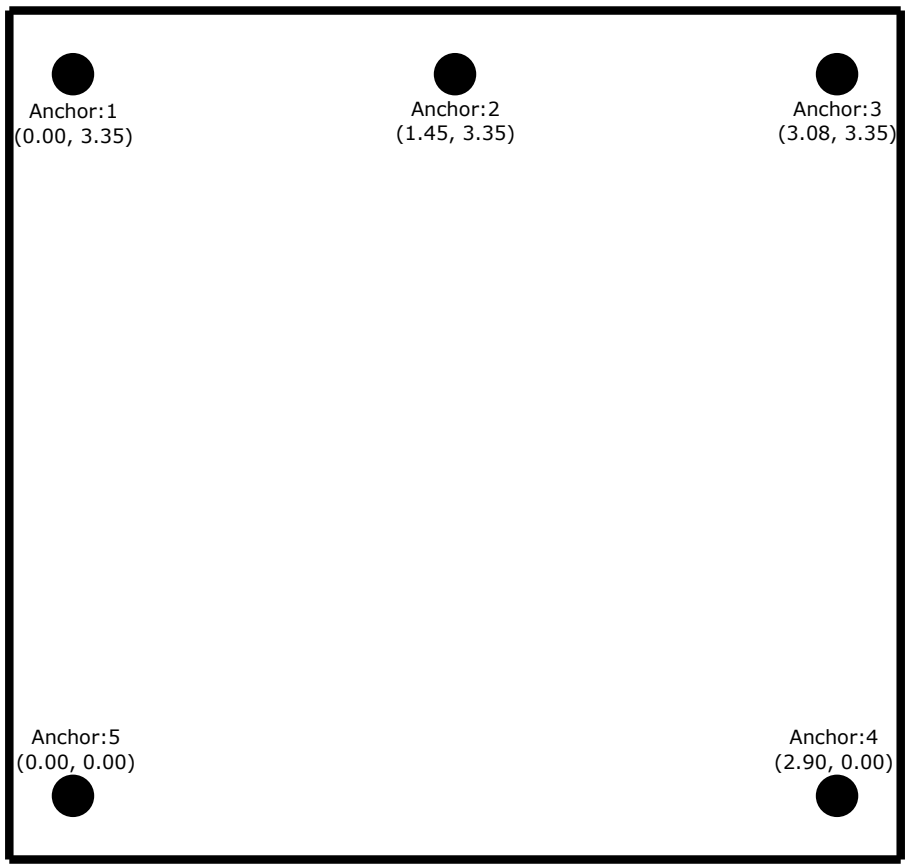


Figure 3.1: Image of the wall with anchor potions from Site-Tech

3.2 Acceptance tests

Testing the requirements is essential to ensure the solution meets the desired functionality and quality standards. The requirements are the foundation for the acceptance tests, they are already described in detail in Section 3, and there will be an acceptance test associated with each of the presented requirements. The results will determine if the requirement has been fulfilled or not, and it will give an insight into the functionality and quality of the solution.

3.2.1 Pose estimation

This test will be made with the robotic system mounted on the test wall at different positions. There will be 10 different points marked on the wall it is not important where they are placed as long they

are in the centre of the workspace as seen in Figure 3.3. Each different point in the X and Y positions together with the rotation around Z, will be measured from the robot position estimation using wire directions, and then physically using measuring tools. During measurement, the robot is holding itself with straight wires and locked motor positions. The physical position on the X and Y axis will be measured from the lower left anchor point and the rotation will be measured relative to a horizontal line. The robot estimated- and physical measurements in X, Y, and Z rotation offsets will be compared to each other. The offset will be defined as the distance for X and Y positions, and as the offset in degrees between the physical and the robot position estimate, as visualised in Figure 3.2.

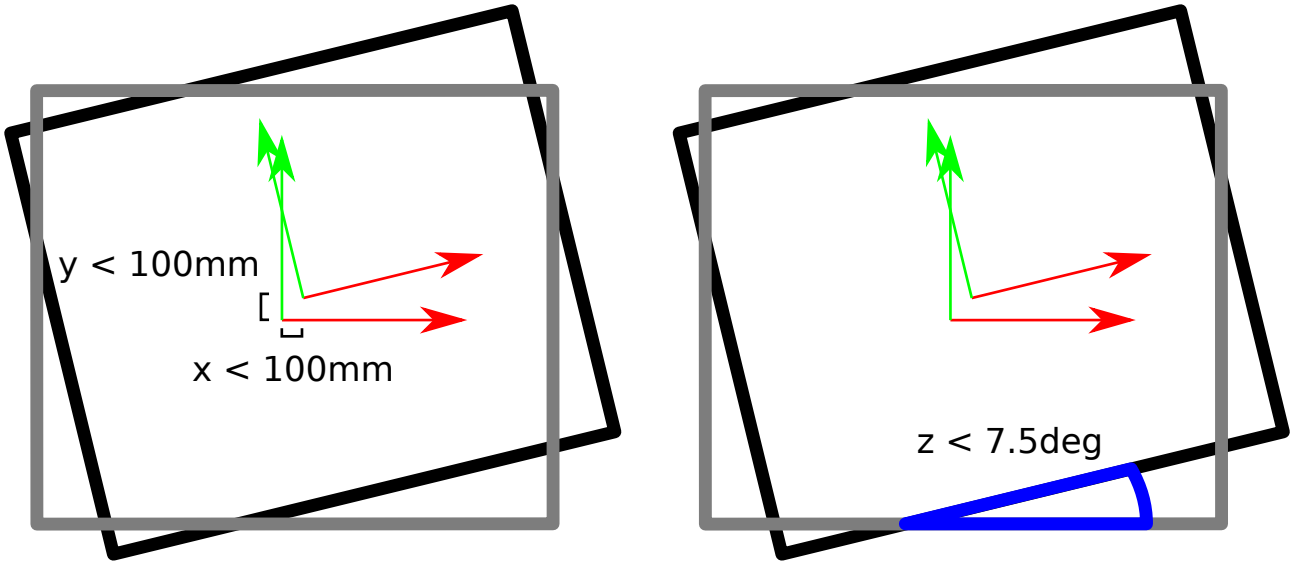


Figure 3.2: Illustration of the Position and rotation estimation The red arrows indicate the X-axis and the green arrow indicate the Y-axis from the robots COM inside the robots frame marked as a gray square for Estimated position, and black square for physical position. 3.2a Illustrates the deviation in X and Y positions, and 3.2b illustrates the deviation in Rotation around Z

When measuring the difference in the position, the offset between the physical position of the Center of Mass (COM) and the estimated position of the COM from the robot the in the X and Y axis will be measured, as visualised in Figure 3.2a. For the rotation around the Z axis, the angle between the COMof the physical robot will be measured and compared to the estimated rotation of the COM of the robot as seen in Figure 3.2b. The points chosen for this test, are scattered over the centre area of the area spanned between the anchor point as seen in Figure 3.3. The exact position of these points are not important, as long as the test is distributed evenly throughout the workspace.

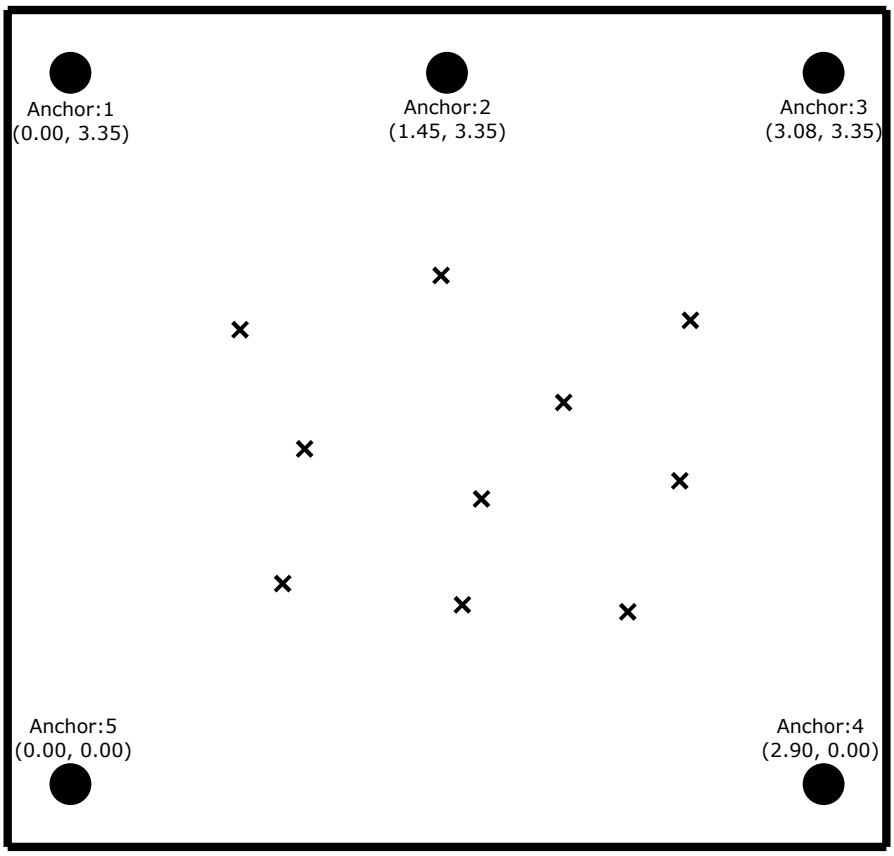


Figure 3.3: Illustration of 10 different points marked on the wall, distributed over the workspace

The test will measure the positional deviation along the X and Y axis, and the angular deviation around the Z axis for each test. If the X and Y deviations are less than 0.1m, and the rotation is less than 7.5deg for a point, the estimation for that specific point will be seen as passed.

3.2.2 Reach goal

The purpose of this test is to measure the robot's ability to reach a given point on the workspace from multiple directions. This test is based on the robot's view of where it and the goal are located, this is done because the purpose is to test the robot's ability to reach the goal. This test will be made at four different points marked around the centre of the wall between the anchor points. The robot will be given a goal from one of the four points to one of the other. This test will be repeated 3 times for each marked point, giving a total of 12 goals. Before testing the robot will be moved to one of the marked points and the robot position and rotation estimation will be noted. After notation, the robot moves to one of the other three other marked points, as seen in Figure 3.4.

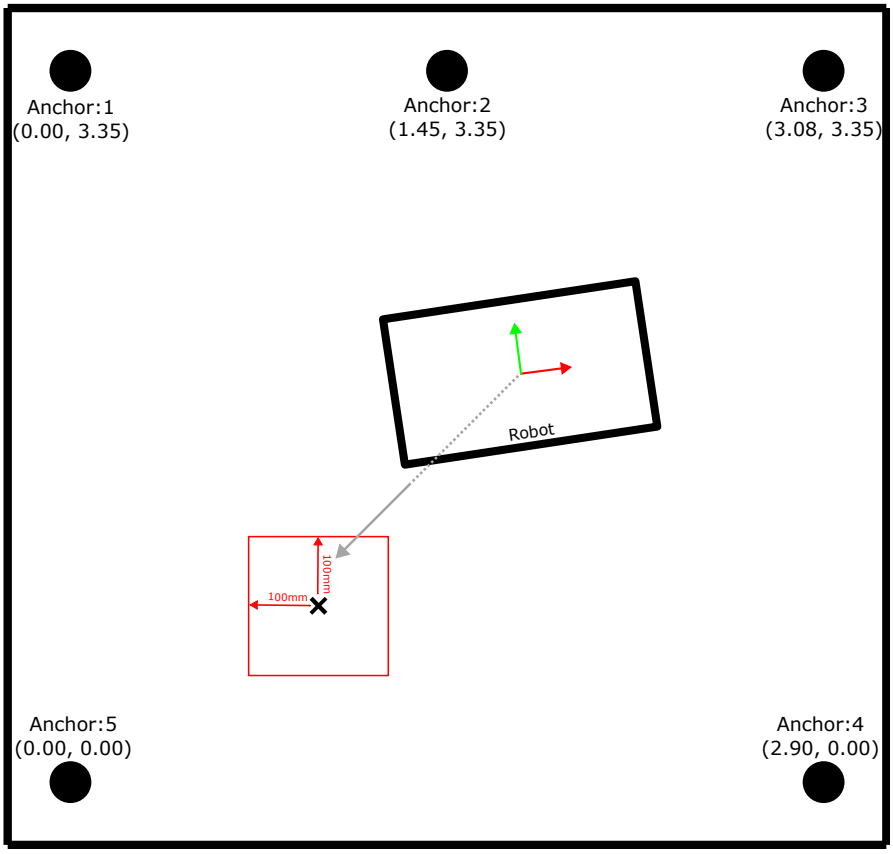


Figure 3.4: The illustration shows the robot moving towards the goal, which is marked on the wall

This test will measure the X and Y distances for the position and the rotation around the Z-axis. if the robot is less than 0.1m in the X and Y distances and less than 7.5deg in rotation around the Z axis from the wanted goal position, the movement is seen as passed.

3.2.3 Follow trajectory

The robot will move between 2 points in a predictable line by following a trajectory path. The robot will be given a straight path from one of the four points to one of the other. This test will be repeated 3 times for each marked point, giving a total of 12 trajectory paths.

This test employs a subjective approach for validating the robot's ability to follow a trajectory and its capability to correct its planned trajectory. The following trajectory will be visually validated, for how the robot corrects offsets to the goal and if the robot follows a straight line to the goal.

As seen in Figure 3.5, a trajectory where the robot is able to correct itself and then reach the goal is preferred over a trajectory where the goal is not reached. If the robot is able to follow the path, but not reach the exact goal the test is still considered passed, as seen in Figure 3.5a.

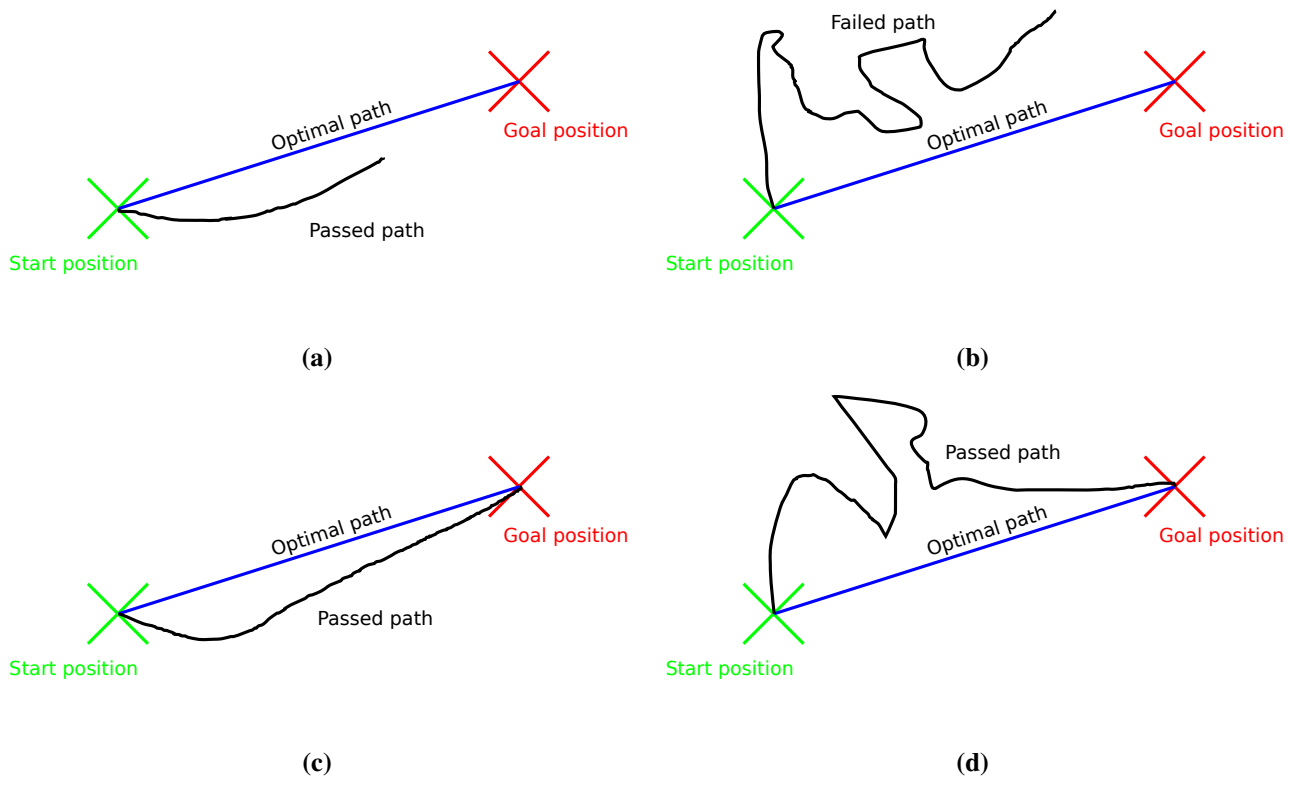


Figure 3.5: Illustration of 1 failed trajectory path in **(b)**, and 3 passed trajectory paths in **(a)**, **(c)** and **(d)**

4 Design

The following chapter describes the design of the robotic system. The goal of the system is to control the motion of the robot towards a desired position on a wall, thereby allowing the robot to reposition itself as shown in Figure 4.1. This motion is accomplished by pulling each of the five wires which are connected to the robot, the anchor points are then fastened to the wall.

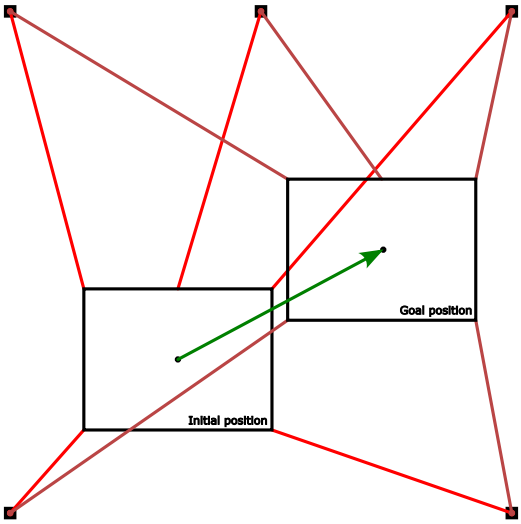


Figure 4.1: Illustration of a motion of the robot frame

The developed robotic system solves this task by computing the wire forces required to follow a path toward the goal. The robot will keep track of its position as it moves and will attempt to correct deviations from its intended trajectory. The system will send the required forces to a motor controller, which will make the motors pull the desired amount. Figure 4.2 shows an overview of the different parts of the system, all of which will be explained in greater detail throughout this chapter.

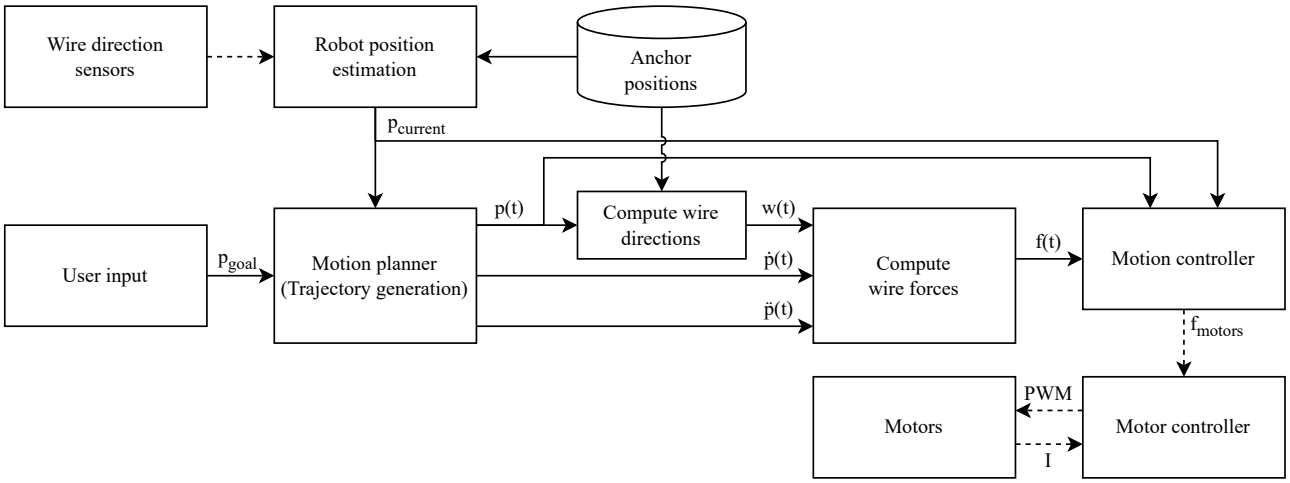


Figure 4.2: Diagram of the developed developed to control the motion of the robot

4.1 Robot representation

The robot is capable of moving along the wall with three degrees of freedom in Cartesian space, which include translation along the X and Y axes, as well as rotation about the Z axis. Its position in the static coordinate frame can be represented by the XYZ coordinates of its COM and its angle of rotation about the Z axis, as seen in Figure 4.3.

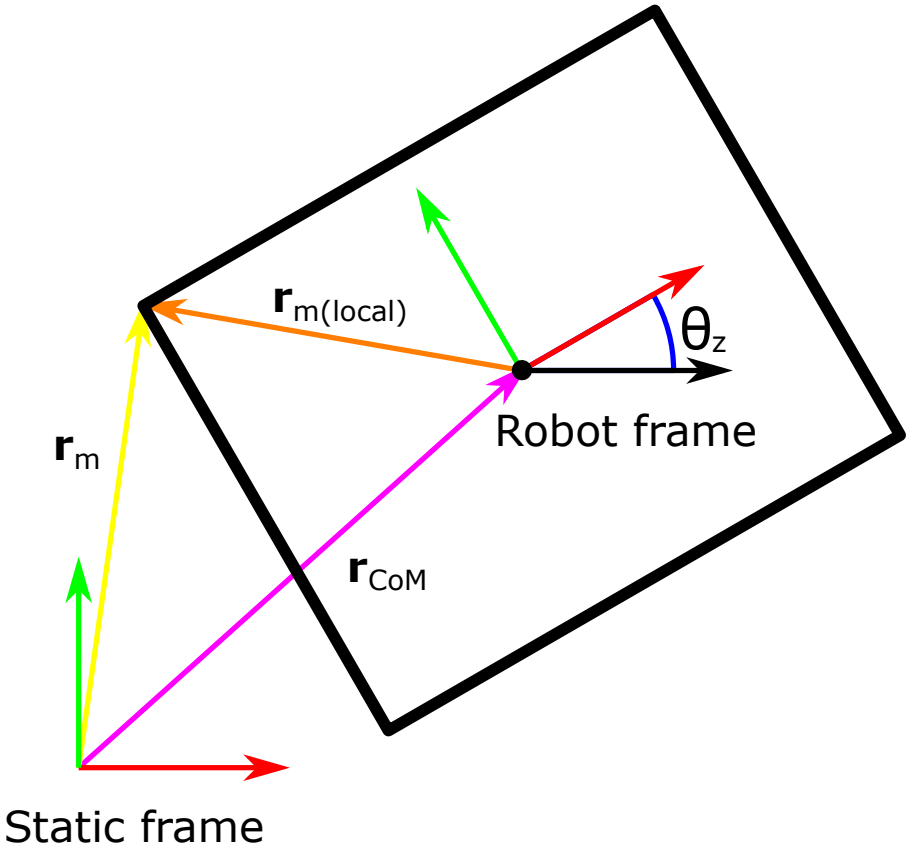


Figure 4.3: Visual representation of the orientation and rotation of the robotic frame relative to the static frame

The anchor points are defined by their XYZ coordinates in the static coordinate frame. The points where the wires are connected to the robot are located with respect to the COM of the robot. To obtain these positions in the static coordinate frame, they must be rotated by the frame angle and displaced by the COM position, as seen in Equation 4.1.

$$\mathbf{r}_m = \begin{bmatrix} cos(\theta_z) & -sin(\theta_z) & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{r}_{m(local)} + \mathbf{r}_{CoM} \tag{4.1}$$

Where:
$\mathbf{r}_m$ is the position where the wire connects to the robot.
θ_z is the angle of the robot about the Z axis.
$\mathbf{r}_{m(local)}$ are the coordinates where the wires connect to the frame, relative to the COM.
$\mathbf{r}_{CoM}$ are the coordinates of the frame COM.

4.2 Robot position estimation

The position estimate is necessary to determine the initial position of the robot motion planner. This estimate is acquired through the geometry of the wires. The relation between the robot's COM and the location where the wires connect to the frame is used to determine the position of the COM from the wire orientations and lengths. To determine the orientation of the wire, Inertial Measurement Unit (IMU)'s are used. On the robot, 3 IMU's can be used in order to find the exact length of the wires, and thereby the robot COM' position.

The direction of the wire based on the position data can then be found, the assumption is, the wire is a straight line, so the direction of the wire can be determined by calculating the angle between the position vector and a reference direction.

By using the accelerometer in the IMU, the orientation can be extracted from gravity. Once the coordinate frame of the IMU is established, it can be reoriented in relation to the gravity vector. This reorientation subsequently allows for the calculation of the wire directions. The arc-tangent function 4.2 can be utilised to find the unit vector from the accelerometer readings. This function uses the pitch and roll angles to calculate a unit vector, which represents the orientation of the device with respect to gravity.

$$\theta = atan2(y,x) \tag{4.2}$$

Where θ is the angle between the position vector and the axis.

The robot orientation around the Z axis to the wall can be found with an IMU having its unit vector parallel to the robot frame. The IMU's on the two wires can give the wire angles, and with the Z-axis IMU the angle to the robot can be found for both wires. Figure 4.4 shows how the position and orientation of the robot frame can be determined by two wires.

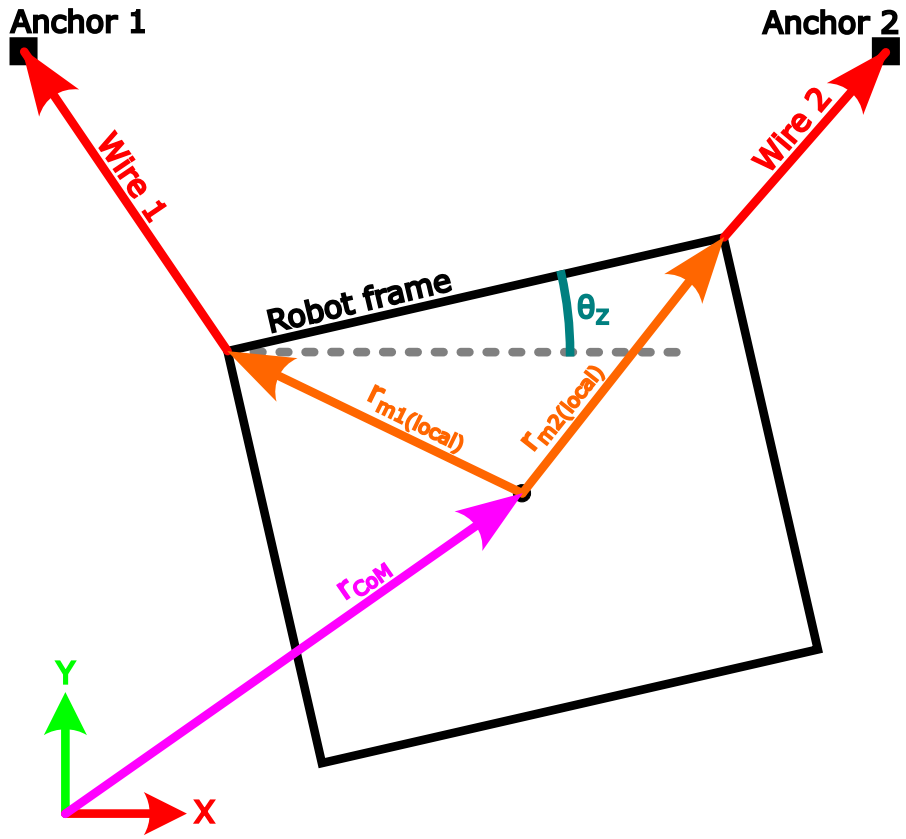


Figure 4.4: Illustration of the geometry used to determine the position of the robot COM. Two wires are sufficient to describe both the position and orientation of the frame

From the wire angles, an estimation of the two wire lengths can be found if the 2 wire anchor-points are known.

With the location of two such points the angle of the frame can be determined as the angle of the line between two points where the wires connect to the frame, as shown in Figure 4.5. With the orientation of the frame, the positions of the COM can be determined from Equation 4.1 shown in the Robot representation section.

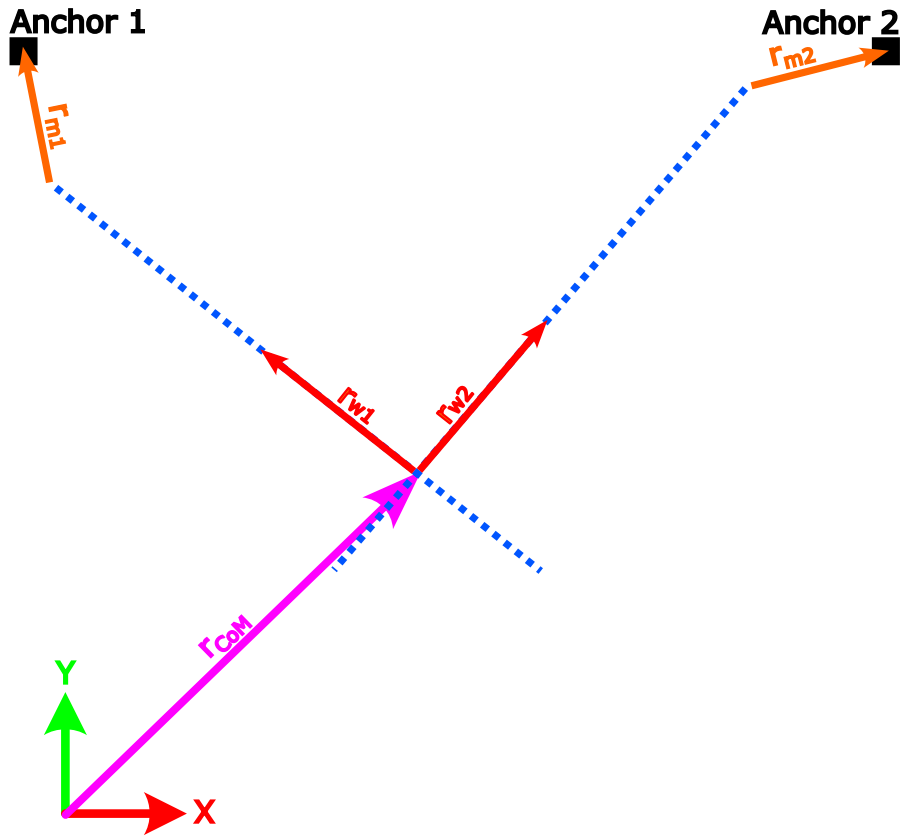


Figure 4.5: Illustration of how wire lengths are found, from known anchor-points and robot orientation

As seen in Figure 4.5, r_{m1} & r_{m2} are the COM to wire vectors, which convey the robot orientation. If r_{m1} & r_{m2} are placed on each anchor-point, representing the orientation and displacement of COM, a blue vector consisting of the known wire direction can drawn. The crossing point between the 2 blue vectors is the exact position of the robot COM. As long as the 2 wire IMU's are pointed directly at their respective anchor-points, the exact robot X-, Y-coordinates, and Z orientation on the wall are known.

4.3 Motion planning

In order for the robot to move to a new position, it needs a path to follow. This path represents the positions the robot will move through during its motion. Three separate paths are generated for the robot's X, Y, and rotational movements to allow for linear movement toward the destination. The trajectories are represented as 5^{th}-degree polynomials to ensure the motion is acceleration continuous. By differentiating the robot's position trajectories, its velocity, and acceleration can be found, as shown in Equation 4.3.

$$\begin{aligned} p(t) &= a_0 + a_1 \cdot t + a_2 \cdot t^2 + a_3 \cdot t^3 + a_4 \cdot t^4 + a_5 \cdot t^5 \\ \dot{p}(t) &= a_1 \cdot t + 2 \cdot a_2 \cdot t + 3 \cdot a_3 \cdot t^2 + 4 \cdot a_4 \cdot t^3 + 5 \cdot a_5 \cdot t^4 \\ \ddot{p}(t) &= 2 \cdot a_2 + 6 \cdot a_3 \cdot t + 12 \cdot a_4 \cdot t^2 + 20 \cdot a_5 \cdot t^3 \end{aligned} \tag{4.3}$$

Where:
p, $\dot{p}$, and $\ddot{p}$ are the position trajectory, velocity, and acceleration polynomials.
a_0, a_1, a_2, a_3, a_4, and a_5 are the polynomial coefficients.

The polynomial coefficients are calculated for every movement by constraining the polynomial to reach the initial and goal state of the motion, as seen in Equation 4.4. By enforcing these constraints the polynomial coefficients can be computed as seen in Equation 4.5.

$$\begin{aligned} p(0) &= x_s \quad p(T) = x_e \\ \dot{p}(0) &= \dot{x}_s \quad \dot{p}(T) = \dot{x}_e \\ \ddot{p}(0) &= \ddot{x}_s \quad \ddot{p}(T) = \ddot{x}_e \end{aligned} \tag{4.4}$$

Where:
p, $\dot{p}$, and $\ddot{p}$ are the position trajectory, velocity, and acceleration polynomials.
T is the time the motion will take to complete.
x_s, $\dot{x}_s$, and $\ddot{x}_s$ are the position, velocity, and acceleration at the beginning of the motion.
x_e, $\dot{x}_e$, and $\ddot{x}_e$ are the position, velocity, and acceleration at the end of the motion.

$$\begin{aligned} a_0 &= x_s \\ a_1 &= \dot{x}_s \\ a_2 &= \frac{\ddot{x}_s}{2} \\ a_3 &= \frac{-20 \cdot x_s + 20 \cdot x_e - 8 \cdot T \cdot \dot{x}_e - 12 \cdot T \cdot \dot{x}_s + T^2 \cdot \ddot{x}_e - 3 \cdot T^2 \cdot \ddot{x}_s}{2 \cdot T^3} \\ a_4 &= \frac{30 \cdot x_s - 30 \cdot x_e + 14 \cdot T \cdot \dot{x}_e + 16 \cdot T \cdot \dot{x}_s - 2 \cdot T^2 \cdot \ddot{x}_e + 3 \cdot T^2 \cdot \ddot{x}_s}{2 \cdot T^4} \\ a_5 &= \frac{12 \cdot x_s + 12 \cdot x_e - 6 \cdot T \cdot \dot{x}_e - 6 \cdot T \cdot \dot{x}_s + T^2 \cdot \ddot{x}_e - T^2 \cdot \ddot{x}_s}{2 \cdot T^5} \end{aligned} \tag{4.5}$$

Where:
a_0, a_1, a_2, a_3, a_4, and a_5 are the polynomial coefficients.
T is the time the motion will take to complete.
x_s, $\dot{x}_s$, and $\ddot{x}_s$ are the position, velocity, and acceleration at the beginning of the motion.
x_e, $\dot{x}_e$, and $\ddot{x}_e$ are the position, velocity, and acceleration at the end of the motion.

An example of the required motion trajectories to move between two points can be seen in Figure 4.6. The motion will adjust the velocities and accelerations to ensure the motion completes within the defined duration.

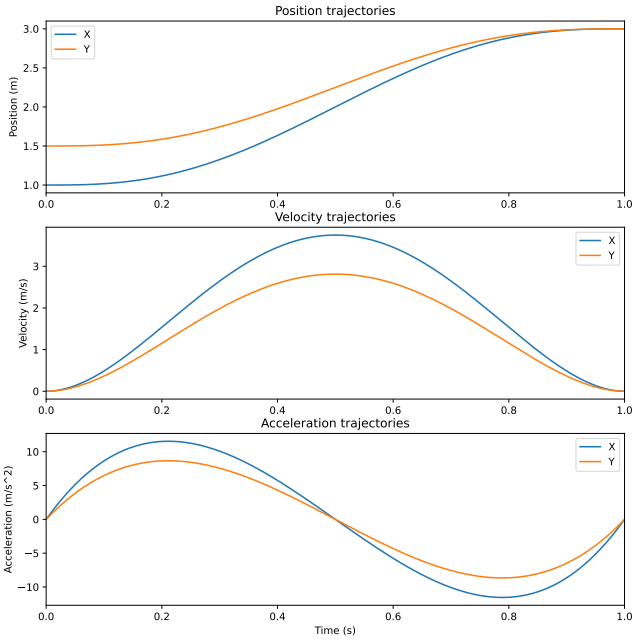


Figure 4.6: Position, velocity, and acceleration trajectories for the movement along the X and Y direction. A movement from (1.0, 1.5) to (3.0, 3.0)

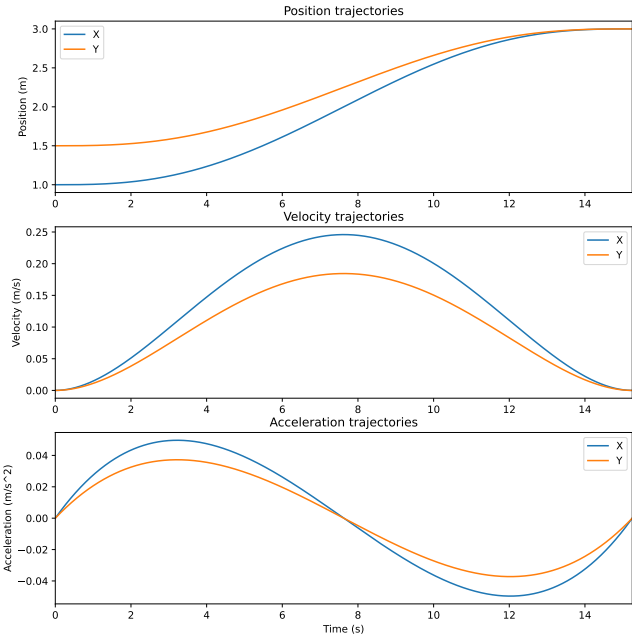


Figure 4.7: Position, velocity, and acceleration trajectories. Movement duration is adjusted to bring the acceleration peaks down

The available motors have limitations in their ability to accelerate the frame. Therefore the duration of the movement must be set high enough to limit the maximum acceleration. To keep the motion linear the motion of all degrees of freedom should have the same duration, as shown in Figure 4.6.

4.3.1 Acceleration limited motion duration

To determine the duration of the motion in seconds a constraint on the maximum absolute acceleration is set. This constraint exists to ensure the motors will not need to deliver excessive force to complete a trajectory which demands high acceleration. To determine a suitable duration for the trajectory a modified binary search algorithm is used[1]. The modified algorithm can be seen in Figure 4.8. The algorithm is divided into two loops. Initially it will continue to double the duration until the acceleration constraint is fulfilled, this exponential growth quickly will quickly find a suitable upper and lower bound for the duration.

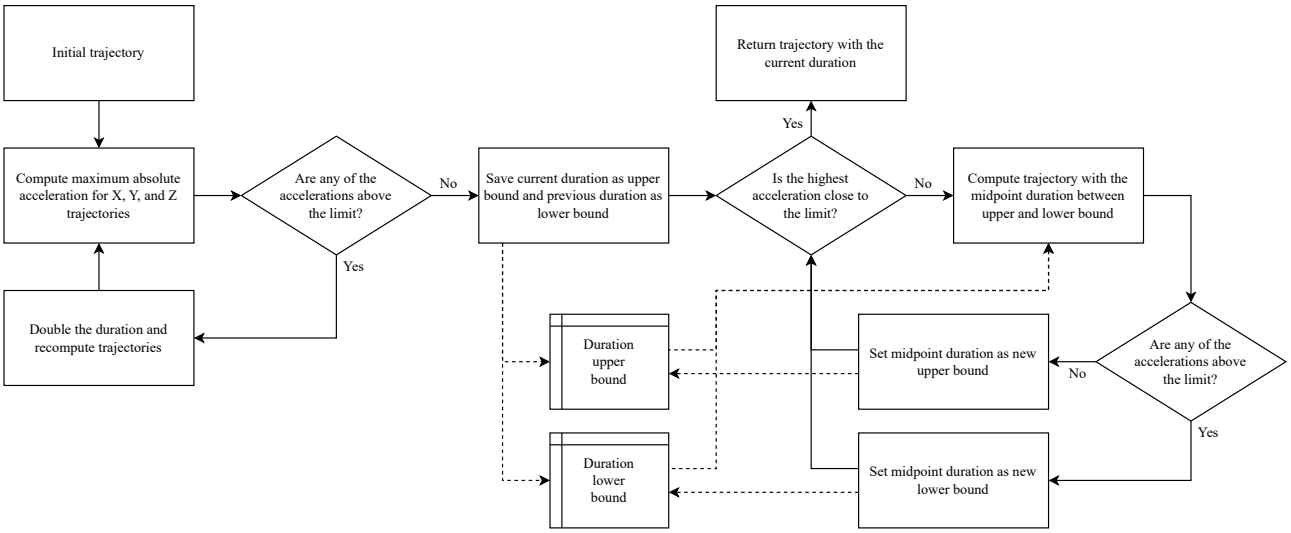


Figure 4.8: Algorithm used to refine the duration of the trajectory to limit the maximum acceleration of the robot

Once an upper and lower bound have been determined the algorithm will perform binary search to further refine the duration until the acceleration which is closest to its constraint is within a defined tolerance of 99% of the constraint. The refinement process can be seen in Figure 4.9.

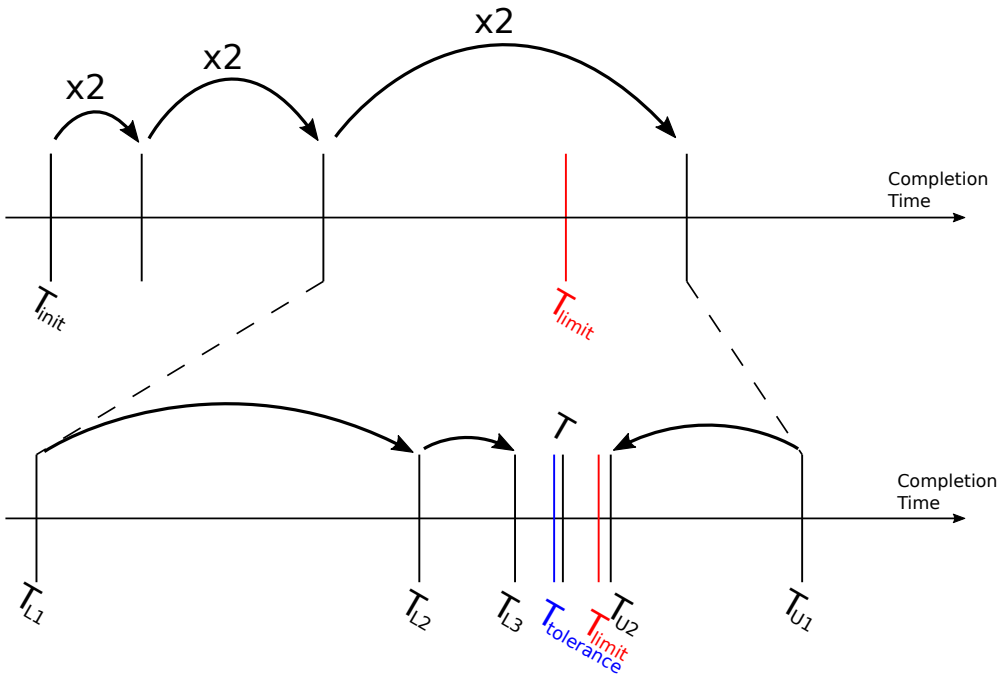


Figure 4.9: Illustration of the trajectory duration refinement by the binary search algorithm. The top part of the figure show the initial exponential growth while the bottom part shows the refinement process. T_L and T_U shows how the new lower and upper bounds close in until a result T is found

4.4 Computing wire directions

The wire directions describe the direction of the force applied by the motor which is connected to the corresponding wire. The directions are formulated as a set of cartesian vectors. The kinematics are calculated as the normalised vectors from where the wires connect with the frame to the anchor points, as seen in Equation 4.6.

$$\hat{\mathbf{w}} = \frac{\mathbf{r}_{anchor} - \mathbf{r}_{motor}}{|\mathbf{r}_{anchor} - \mathbf{r}_{motor}|} \tag{4.6}$$

Where:
$\hat{\mathbf{w}}$ is a unit vector describing the direction of the wire.
$\mathbf{r}_{anchor}$ is the position of the anchor points on the wall.
$\mathbf{r}_{motor}$ is the position where the wire connects to the robot in the static frame.

4.5 Wire forces

The robot dynamics are formulated to determine how the motors should exert force to achieve the desired movement described by the generated trajectory. The dynamics of the robot in this project are formulated using Newtonian mechanics. Each wire connecting the motors to the wall exerts a force on the frame. The directions of the forces are determined through inverse kinematics as is the wire directions, while the magnitudes of the forces are equal to the force exerted by the motors.

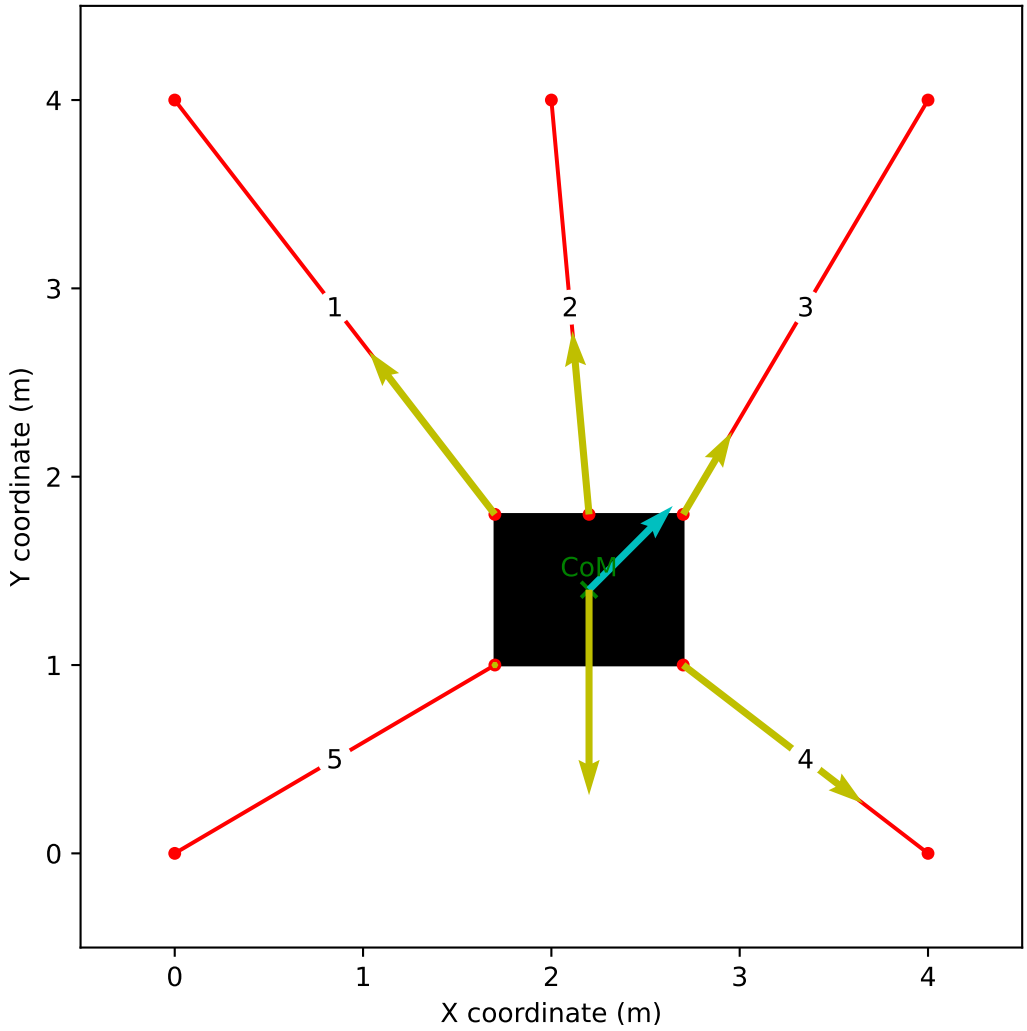


Figure 4.10: Necessary relative force magnitudes are required along the wires to apply a diagonal acceleration to the robot.

By summing up all the forces which are acting on the frame, the acceleration of the robots COM can be determined, see Equation 4.7. Likewise, the sum of all moments informs the angular acceleration of the robot as shown in Equation 4.8.

$$\mathbf{a} = \frac{\Sigma \mathbf{F}}{m} \tag{4.7}$$

$$\alpha_z = \frac{\Sigma M_z}{I_z} \tag{4.8}$$

Where:
a is the acceleration of the robot COM.
$\Sigma \mathbf{F}$ is the sum of all forces acting on the robot frame.
m is the mass of the robot.
α_z is the angular acceleration about the Z axis.
ΣM_z is the sum of all moments, about the Z axis, acting on the robot frame.
I_z is the inertia about the Z axis of the robot.

In addition to the forces from the wires, the frame will also be subject to gravity and dampening forces such as friction. The wall will also exert a reactive force on the robot, equal and opposite to any force in the negative Z direction. This makes the forces in the Z axis sum to zero as the robot cannot push itself away from the wall.

The moment arms are equivalent to the displacement from the COM to where the wires connect to the frame, in the static coordinate frame. In this project, it is assumed that the angular acceleration about the X and Y axes will always be cancelled out by the normal force from the wall.

Through this formulation, three equations arise which describe the forward dynamics of the robot, see in Equation 4.9. The forward dynamics describe acceleration and angular acceleration, for a given set of motor forces. The dynamic equations are position dependent as the wire directions vary throughout the workspace.

$$\begin{aligned} a &= \frac{\sum_{i=1}^{5}(\mathbf{F}_{wi} \cdot \hat{\mathbf{w}}_i) + \mathbf{F}_g + \mathbf{F}_f}{m} \\ \alpha &= \frac{\sum_{i=1}^{5}(\mathbf{r}_{wi} \times (\mathbf{F}_{wi} \cdot \hat{\mathbf{w}}_i)) + \mathbf{M}_f}{I} \end{aligned} \tag{4.9}$$

Where:
a is the acceleration of the robot.
α is the angular acceleration of the robot.
$\mathbf{F}_w$ are the five wire forces.
$\hat{\mathbf{w}}$ are the fire unit vectors describing the directions of the wires.
$\mathbf{r}_w$ are the positions where the wires connect to the robot in the static frame.
$\mathbf{F}_g$ is the gravitational force experienced by the robot.
$\mathbf{F}_f$ and $\mathbf{M}_f$ describe the dampening forces and moments caused by friction.
m and I denotes robot mass and inertia tensor.

The inverse dynamics are needed to determine the motor forces required for a defined acceleration. Formulating an inverse expression is not possible, as the sum of wire forces lacks a unique inverse. This makes the process of finding suitable motor forces non-trivial. The algorithm implemented to determine suitable motor forces will be explained in the implementation chapter.

The wires are unable to push on the frame, meaning they cannot deliver negative forces. For the inverse problem, this means the robot cannot create all moments. Dependent on the placement in the workspace it is only capable of delivering positive or negative moments.

Since the motors used to tighten the wires for moving the frame are not made to push the wires, there are some problems when the motors need to give a negative force in the wire direction. This has an effect on the inverse dynamics, meaning the robot cannot create all movements needed for total control of the motion inside the given workspace. Depending on the placement in the workspace and the given movement, some of the motors must affect the wires with a negative force resulting in a long line.

The lines must make a negative force mostly at the opposite side of the positive, and the side which the system is moving towards. In a vertical movement, the lower motors loosen the wire in an up-warding movement as the top motors are reeling the wires in, and the opposite when the movement

is downwards.

4.6 Motion control system

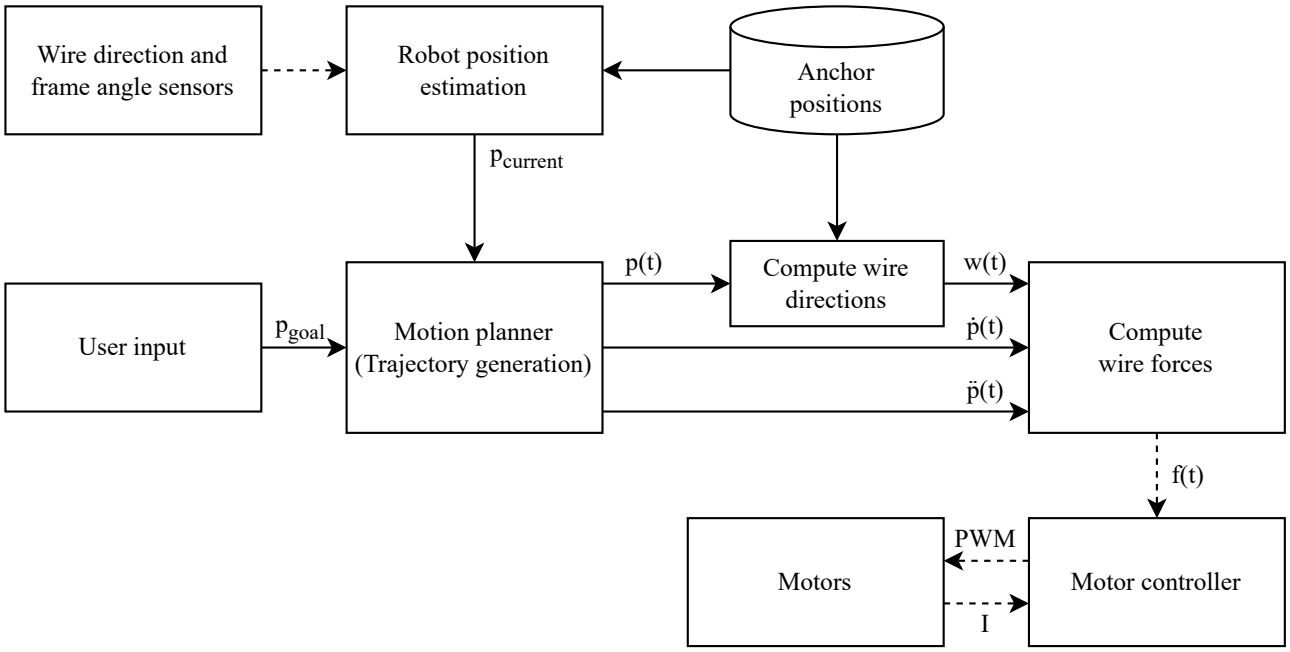


Figure 4.11: Flowchart of the motion control system

The motion control system continuously updates the trajectory to the desired end goal. With each update, the control system calculates a new trajectory from the current estimated position, velocity, and acceleration toward the desired end position. This recalculation and the error-handling process take place at regular intervals, enabling the robot to reach its end position while compensating for external forces and system errors. By utilising the iterative capabilities of the short updated trajectories, this control loop effectively adapts to larger non-modelled errors and inequalities. Once a trajectory is determined, the required forces to follow the trajectory are computed as discussed in the previous section. The resultant force trajectories are sent to the motor controller.

4.7 Motor control

The motor control system is responsible for controlling the force of the 5 motors, which are essential for the proper functioning of the robot. This section will discuss the motor control system and its key components. The motor control receives 5 timed wire force arrays for each motor's trajectory, the controllers' task is to ensure constant desired motor forces exerted at the cables on the planned trajectory times.
The 5 motors are supplied with a 12V DC, which cannot feasibly be changed independently between

motors to achieve the exact desired forces on the wires. The control scheme for controlling the individual power given to the motors is through Pulse-width Modulation (PWM) of the static voltage. Each motor controller has to ensure the motor reaches and holds the desired force on the wire at that specific time. When a trajectory is generated and the timed forces are passed on to the motor controller, it has to ensure the motors follow the exact timing and force on the wires to minimise errors.

4.7.1 Motor modelling

The controllers have to adjust the effective wire forces with a PWM Duty Cycle (DC), which through varying signal frequencies between 0% and 100% controls the effective current delivered to the winches.

$$I_{effective} = I_{max} * PWM_{DC} \tag{4.10}$$

Where:
I_{max} is the maximum current delivered.
PWM_{DC} is the DC fraction.
$I_{effective}$ is the proportion of current effectively delivered to the winch.

In order for the controller to precisely control the desired forces with a PWM DC-fraction, a transfer function from the DC percentage to a wire force has to be modelled. Since the winches are able to exert the necessary forces and are not intended for precision control, limited data is available and the motor specifications has to be found in laboratory setups at AAU. The amount of current drawn and the force exerted by the winches make it hard to test the motor constants even in laboratory environments. This means only the lower force exertions of the motor, can be effectively modelled, as it is not possible to model the higher forces with the current test setup. Different motor constants have to be tested and found for modelling, since the motors mostly will be in somewhat static states or low velocities at most times, the motor velocity constant might be negligible and the back Electromotive Force (EMF) low enough to be ignored. The motor torque constant describes how the torque exerted by motor, is affected by the current in the motor's coils.
This relation can be setup with the formula:

$$K_\tau = \frac{\tau}{I_a} \tag{4.11}$$

Where:
I_a is the armature current going through the motor coils.
τ is the angular moment exerted by the motor drive shaft.
K_τ is the motor-torque constant describing the motor force.

The motor constant relation is plotted over various currents and torques, mapping the motor behaviour. It is assumed the relation, especially in the lower forces, is close to linear, which makes it possible to model the relation as the best-fitted linear equation on the plot. The y-intercept of the fitted equation is the Stall torque of the motor, and the gradient is the motor torque constant, as seen

in Figure 4.12. Here each dot represents a measurement of Torque at a different current and the red line represents the assumed linear relation between Torque and current.

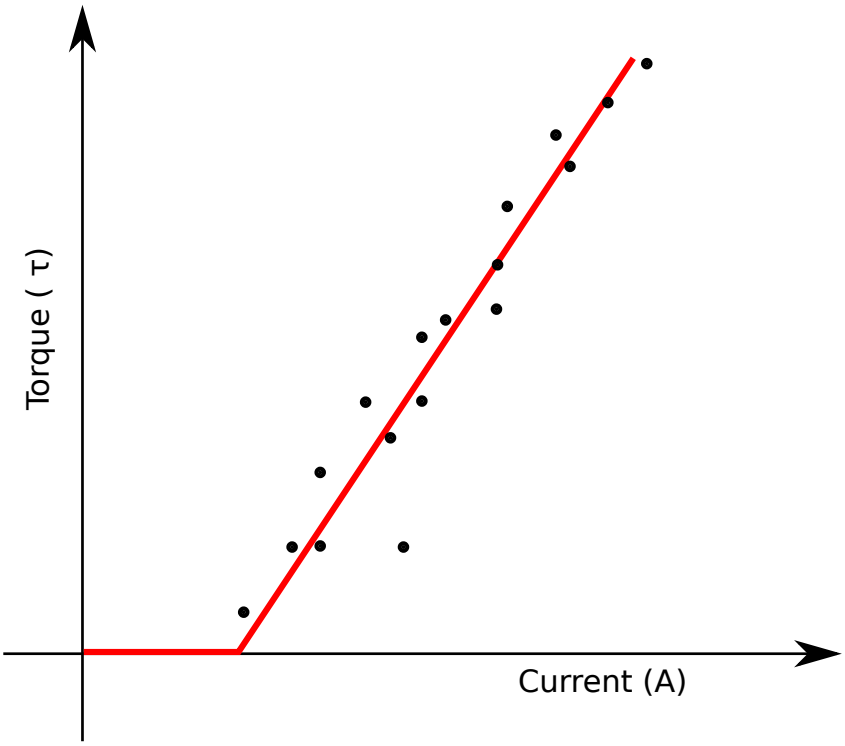


Figure 4.12: Torque to current relations linearised, example how a simple model could be made

4.7.2 Control loop

The simplified model is not able to take in factors such as motor disturbances, PWM generation- and motor resistances, and motor back-EMF into account. To make up for uncertainties in the model, closed-loop feedback is introduced to the motor controller. Since no encoder is available, the motor current is used as feedback to the controller. Any deviations from the desired torque can be compensated by introducing a Proportional–Integral–Derivative (PID) controller. The block-diagram for the motor controller can be seen in Figure 4.13.

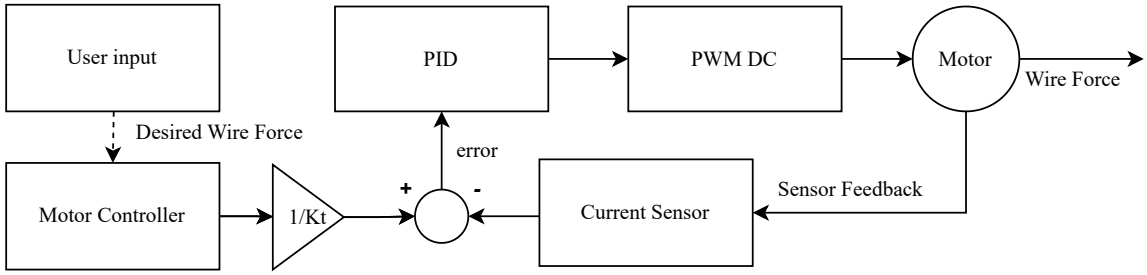


Figure 4.13: Block-diagram of proposed motor controller

The proposed block diagram for a motor current controller, which can take a wire force as an input. The force is converted to a current using the motor-torque constant K_τ. A PID controller regulates the current and requires tuning for the entire controller. The PID outputs a current which can be converted to a PWM Duty Cycle. The current to PWM Duty Cycle might need to be modelled. The desired force will then be exerted on the wire, and the PID-controller receives feedback from a current sensor model.

With all the necessary elements in place, from the general overview of the workflow, computations between motors and anchor points, trajectory planning and more, the system can now be implemented.

5 Implementation

When implementing the various components, both the physical components and the relevant software, the interaction and collaboration must be discussed. Here the physical components include a frame, motors, sensors and microcontrollers. The implementation of the software is divided into two main parts, the software on a nearby computer and the software on the microcontrollers. The development and function of each part will be described in detail in the following sections.

5.1 Mock-up model

A Mock-up model is essential for this project, not only to do tests but also in order to analyse the behaviour of the system and to see if it acts as expected.

Adjustments have to be made to the actual setup due to a limited testing area and the limitation of available components. One of the flaws with the components is, they cannot tolerate high amounts of current, unlike the motors. This becomes more problematic in a small work environment, as the forces required to move around the workspace edges are high due to more of the wire forces being directed at the wall.

The required currents exceed the capacity of most components to move around in the workspace. Due to this, the depth of the Mock-up frame has to be decreased in order to have meaningful tests. The reduction of the Mock-up means it needs to use less force and current to move around the workspace compared to the model built by Site-tech.

The goal is to test the current control pipeline, where every aspect will be thoroughly tested and adjusted based on the results, the end goal of the testing phase is to have a functioning control system, which would allow the robot to automatically move around the workspace to desired locations given from computer inputs. An illustration of the Mock-up model can be seen in Figure 5.1.

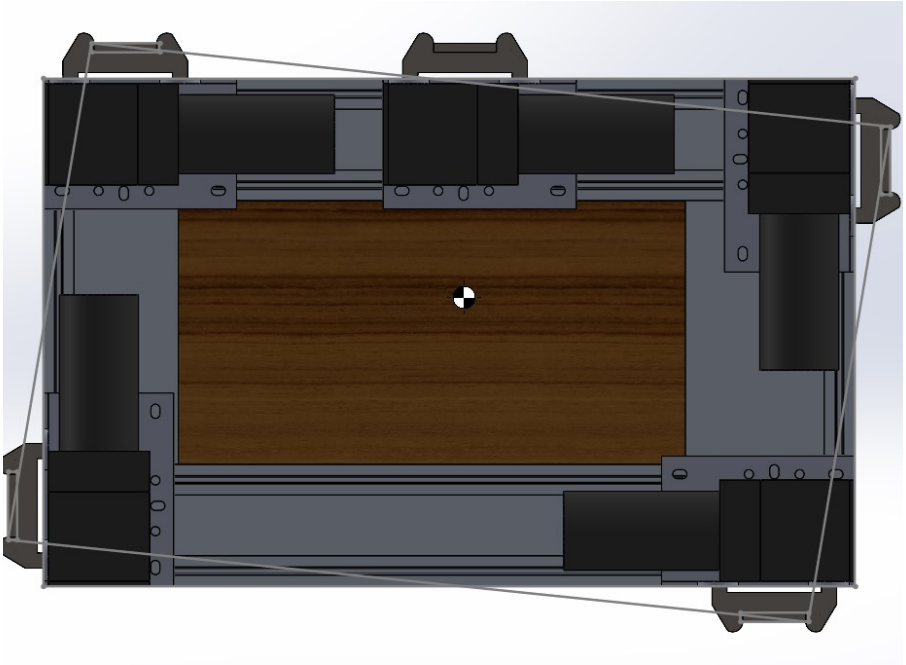


Figure 5.1: Digital image of the Mock-up

When building the Mock-up model, it is essential to incorporate as many components which are directly applicable to the existing system, as possible. This approach simplifies Site-Tech's transition from the Mock-up to the actual model. Many components are not readily available, which requires their substitution.

The Mock-up is made in such a way, the VidaXL motors, see Appendix A.1, used by Site-Tech can be directly tuned and tested on the Mock-up, and the entire control system can be applied to Site-Tech's existing solution.

The Mock-up model is designed to ensure wide accessibility of parts, meaning if components break, replacements can typically be procured within a short time frame of 2-3 working days. This proves to be important, given the forces being tested, as some of the more fragile components may get damaged during testing and exceed their current limit. The design needs to incorporate IMUs into the wires in a way where it would be possible to gather useful data from them. To secure the IMU to the model, a soft body is installed between the IMU and the wire-feeder, as visualised in Figure 5.2. The Mock-up model is limited to the frame, motors, wires, and required sensors, for generating the trajectory and controlling the motors. Consequently, the solution isn't directly transferable to Site-Tech but requires adjustment.

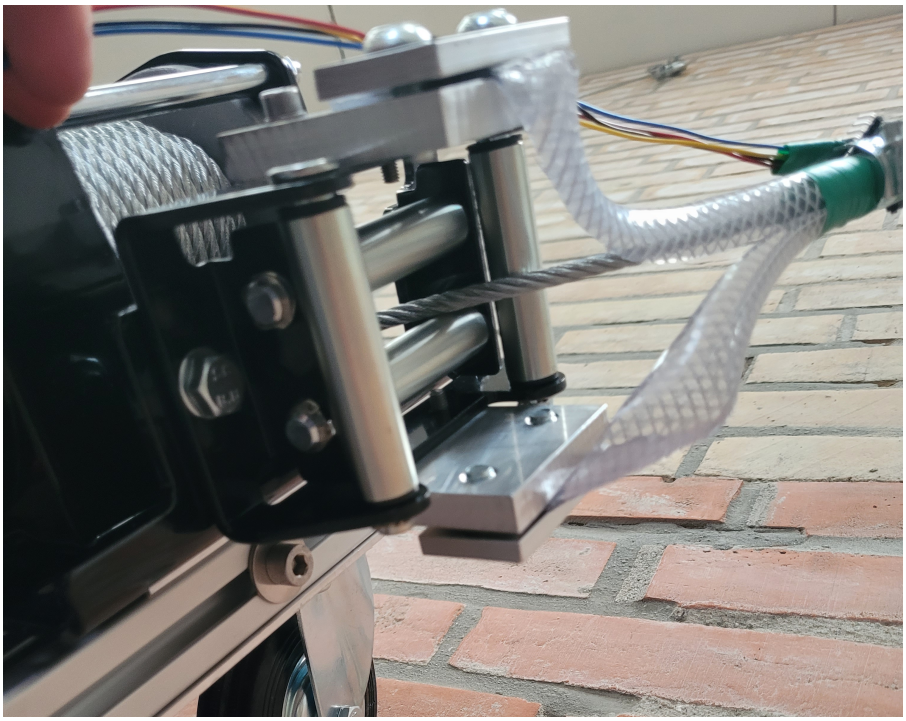


Figure 5.2: The installation of the soft body IMU holder mounted to the wire-feeder

In conclusion, the Mock-up model is designed and developed to overcome the challenges posed by the testing environment and the components. This is done to provide a platform for testing the current control pipeline. To address these challenges, the depth of the setup is drastically decreased, which enables the setup to move around with less force, hence requiring less current. The design process focuses on using accessible parts which could be easily replaced in case of damage or failure during testing. Soft body holders are installed to secure the IMUs and gather data from them.

5.1.1 The physical system

The physical system consists of six 30x30mm Sigma profiles[2], four at the length of 0.740m in the horizontal plane, and two at the length of 0.500m in the vertical plane. These sigma profiles are used for holding the robot together, and making the general build of the frame as seen in Figure 5.3.

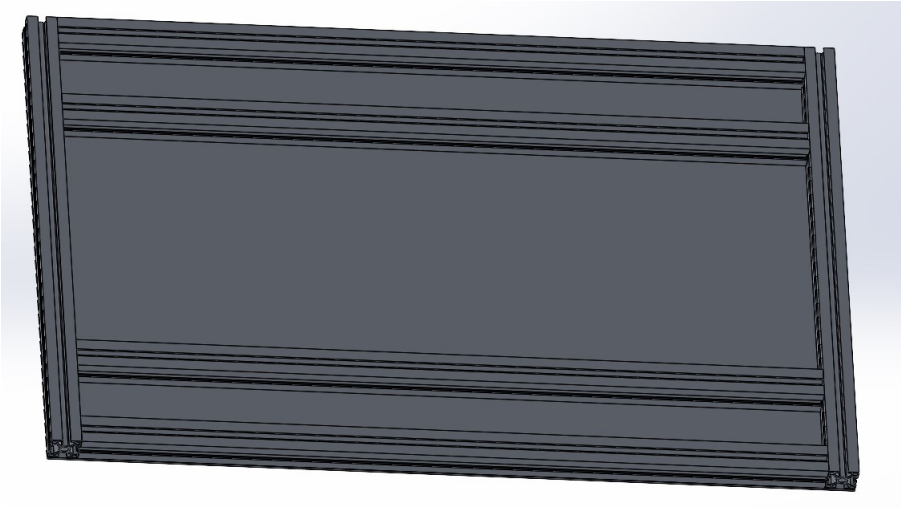


Figure 5.3: Digital image of the frame and aluminium back plate

For support, an aluminium plate with the dimensions 0.500x0.800x0.003m is placed behind the Sigma profiles. The purpose of the aluminium plate is to give the Mock-up frame a higher resistance against bending and twisting around itself due to the exerted forces from the motors.

For mounting the motors to the frame, five Aluminium plates with the size of 0.150x0.120x0.010m are made and placed on top of Sigma profiles in designated spots. For connecting the parts, four holes are made for connecting the plate to the frame, and two for holding the motor. The motor frame with the wire feeder, and the Aluminium plate together. This gives a rigid/solid interconnection throughout all parts, which gives a final setup as seen in Figure 5.4, where the frame is placed in the bottom followed by a small aluminium plate, motor frame, and the motor at the top.

Figure 5.4: Image of the final setup of the motors on the frame using connectors

The placement of the motors ensures a strong junction between the components but affects the angle which is used for calculating the position of the robotic frame. The placement of the motors

affects the plane stretching between the motors compared to the plane of the frame made from the back-plate and sigma frames as seen in Figure 5.5. This angle between the two planes does not affect the general efficacy of the system but must be kept in mind when observing during testing.

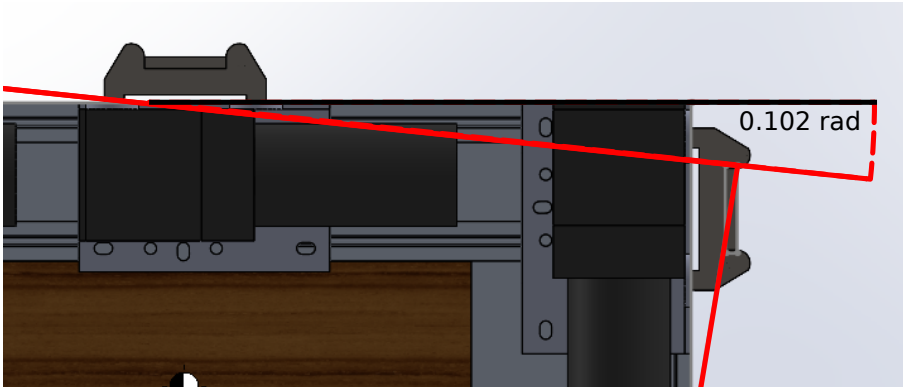


Figure 5.5: The angle offset between the bag plate and the motor plane

5.1.1.1 Electronics

To house the necessary electronics, a wooden plate is affixed to the centre of the frame, as shown in Figure 5.1. This arrangement facilitates the secure mounting of modules on the frame and simplifies the drilling of holes for the components. Given the wooden board is only required to support the control components and modules, with no force from motors to contend with, a wooden platform is considered sufficiently robust.

The motor control system employs five motor drivers and five current sensors, all linked to a single ESP32 unit as visualised in the electric circuit for the Mock-up in Figure 5.6. Each motor driver is connected to a current sensor to measure the current drawn from the corresponding motor. For communication with the nearby computer, the ESP32 unit managing the motors is connected to another ESP32 unit via a serial connection. This additional ESP32 unit ensures a stable connection to the computer using Transmission Control Protocol (TCP) over a WiFi connection.

To determine the current orientation and rotation of the frame, three IMUs are utilised. These are located at the two upper corner motors and one in the centre of the frame and each is connected to an ESP unit as visualised in Figure 5.6. The ESP units transmit data to the nearby computer over a WiFi connection.

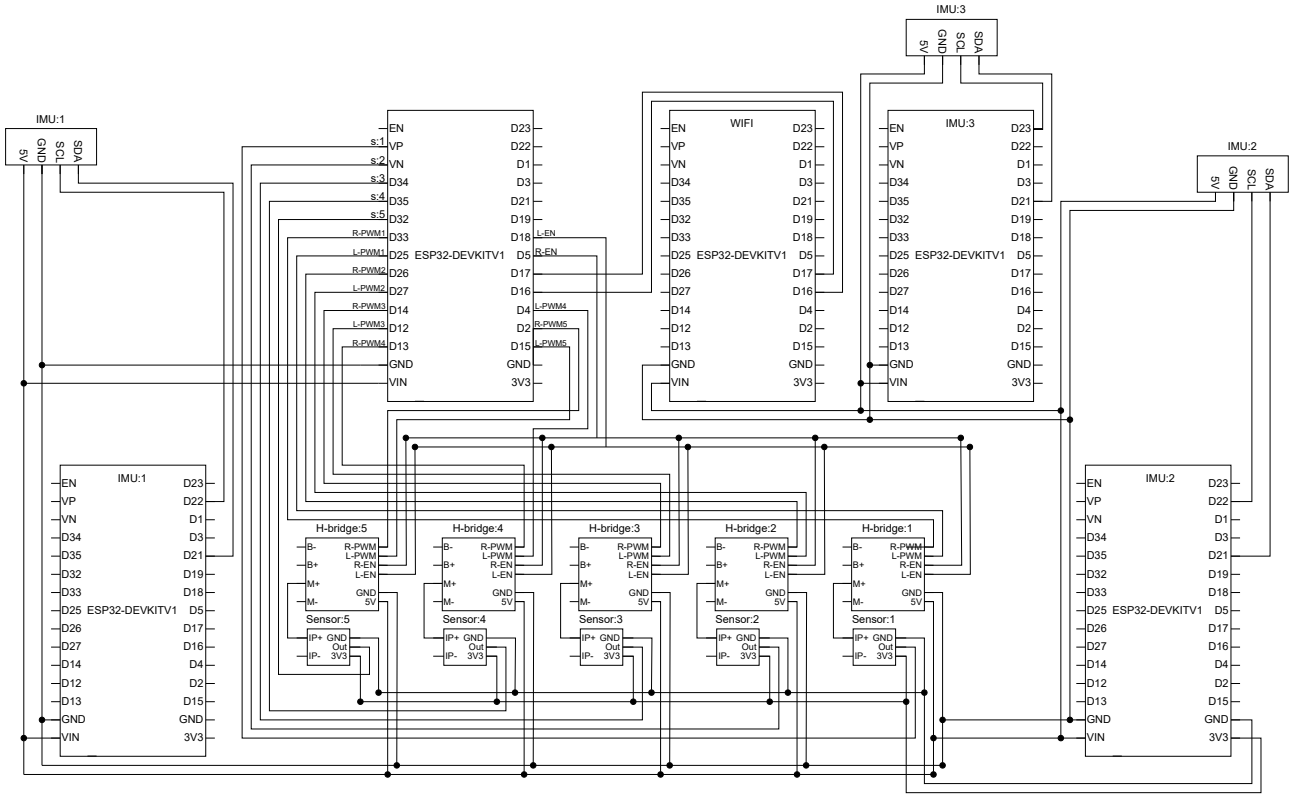


Figure 5.6: Over-view of the electric circuit of the Mock-up setup.

For powering and monitoring the motors, each motor is connected to an H-bridge and a current sensor. For powering the motors, an external power source is connected to the B+ and B- and the M+ and M- terminals are connected to the motor. A current sensor is incorporated between the motor and the H-bridge, as seen in Figure 5.6. In order to vary the power exerted by the motor, a PWM DC signal can be altered to the H-bridge and the turn-direction of the motor can be changed by altering a digital sign. The current drawn by the motor can be registered by the current sensor.

After addressing the hardware components and their functioning for the Mock-up, the focus can be turned towards the software aspects of the system. While the hardware ensures the movement and physical functioning of the system, the software handles computation, trajectory control, and other essential tasks.

5.2 Software framework

The system consists of several programs spread out onto the different devices, as shown in Figure 5.7. The main computation and trajectory control is handled on a laptop running the *Motion Controller.py* Python program. This program handles generating the position estimate, creating a trajectory towards the goal, and calculating the required motor forces to reach the goal.

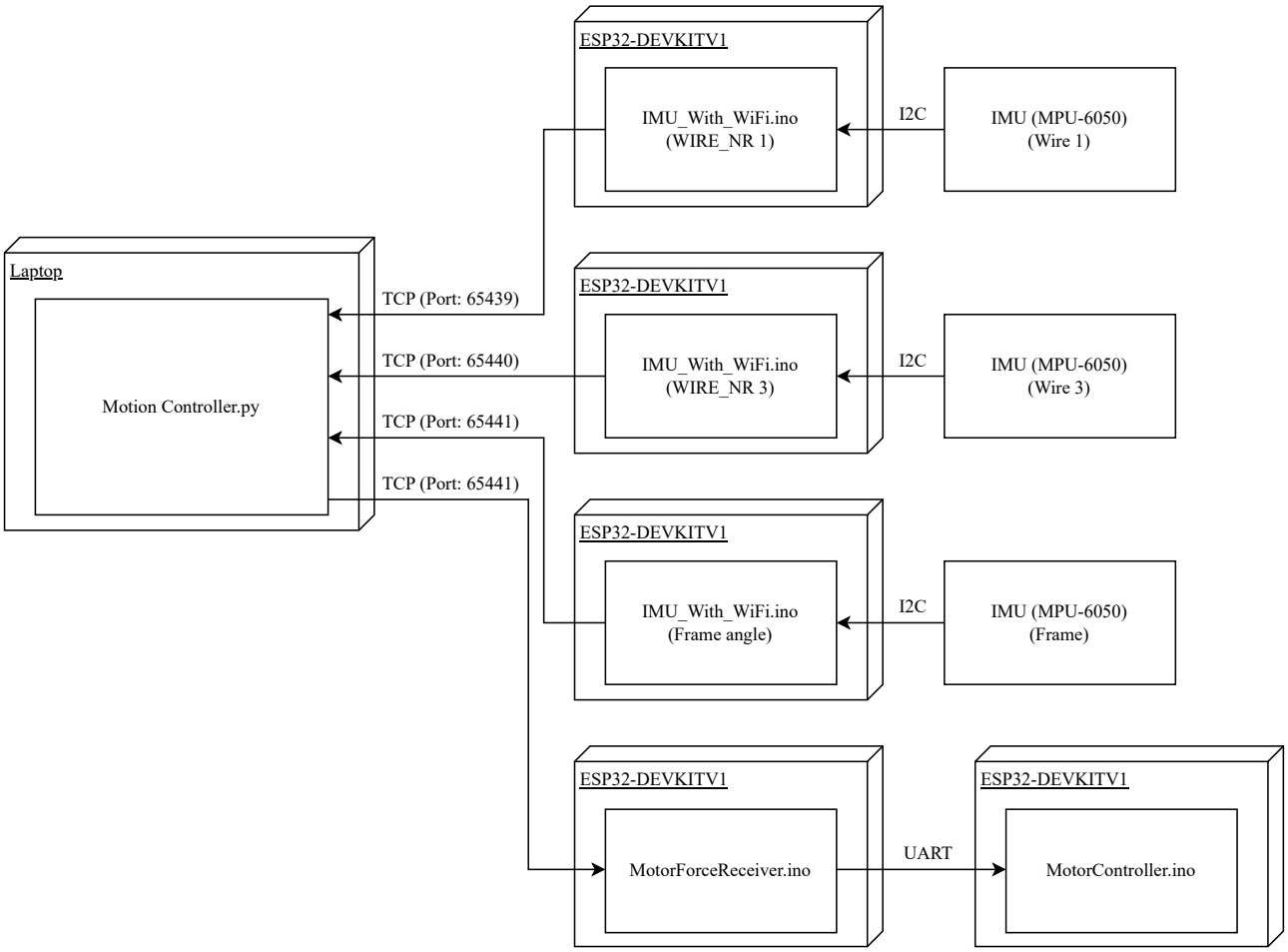


Figure 5.7: Deployment diagram for the developed system.

The ESP32 microcontrollers are tasked with collecting sensor data and operating the motor control loops. All communication between the Python program and components mounted on the frame is done via wireless Transmission Control Protocol (TCP) connections. The three ESP32s which handle collecting wire orientation data for the position estimate are running the *IMU_With_WiFi.ino* program. Each of these only interface with one IMU via I2C communication. The two remaining ESP32s are used to respectively receive the desired motor forces from the motion planner, and operation of the motor control loop for all 5 motors. The motor controller, running the *MotorController.ino* program, handles PWM generation and current sensor readings. The receiver module, running the *MotorForceReceiver.ino* program, acts as a secondary WiFi module for the motor controller which does not have enough available pins to have its own WiFi module active.

The *Motion Controller.py* Python program handles the role of TCP server. As shown in Figure 5.7, each ESP32 client connects to a different port. The TCP connection is established by using the python *socket* library, and the *WiFi.h* and *WiFiClient.h* libraries for ESP32.

After establishing the roles and communication protocols between the microcontrollers and the Python program, attention turns to the crucial aspect of data collection, sensor mounting on the mock-up, and how these sensors facilitate the localization and movement of the robot. While the microcontrollers and the Python program ensure the proper functioning and communication within the system,

sensors provide the necessary data for real-time operation and control.

5.3 Sensor data collection

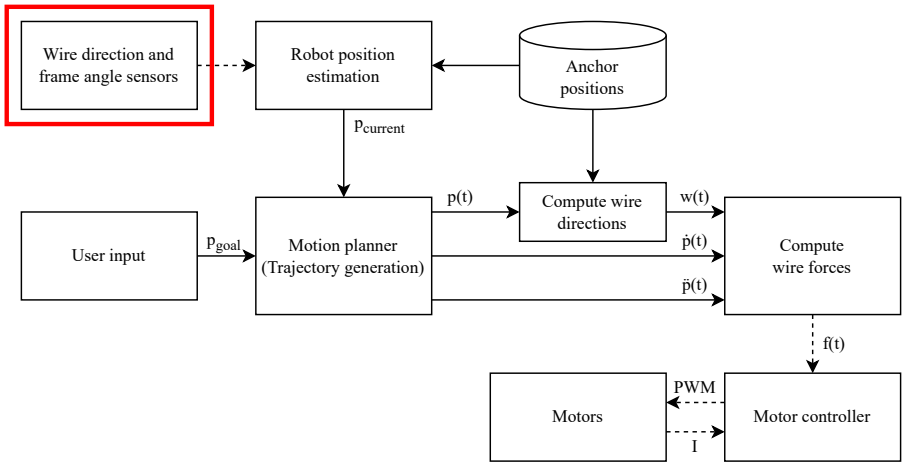


Figure 5.8: Sensor data collection in the pipeline

This section will cover data collection, mounting of the sensors on the Mock-up and how it transmits data which can be used to localise and move the robot.

As discussed in Section 4.2, the IMU must be mounted on a flexible body that aligns with the wire's direction. To create this flexible body, a garden hose is cut to an appropriate size, and small metal tubes are added to each end of the hose for support. This design reduces friction between the hose and the metal wire and is placed as illustrated in Figure 5.9. A flexible body with an IMU sensor is mounted on each of the 2 top corners for determining the position of the robotic frames COM, as described in Section 4.2.

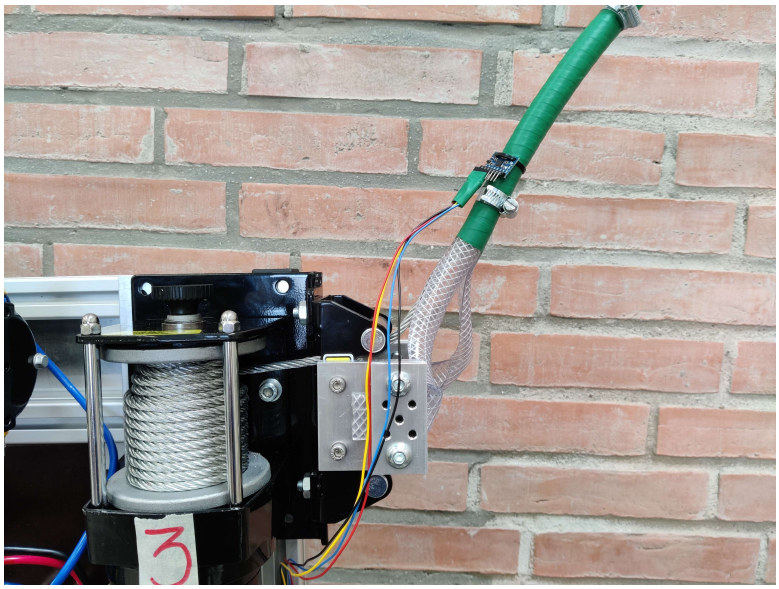


Figure 5.9: A detailed close-up image of the IMU mount connected to a motor

The flowchart shown in Figure 5.10 showcases how the program: *IMU_With_WiFI.ino*[a], which is how IMUs communicate and transmit their data. The IMU data traverses through a program where gravity directions are extracted and then transformed into wire direction coordinates for the X, Y, and Z axes. The processed data is transmitted to the server via a TCP connection, where the server hosts the position estimation program for the robot frame.
The third IMU is placed in the centre of the robot frame, the data is transformed by using inverse tangents, as the goal is to get the rotation of the robot frame, which is sent to the server using TCP.

[a]https://github.com/Gsvend20/P6_Site_tech_removing_Robot/blob/7d02df17acef740fd110a05e7ac8730fe841292e/ESP/IMU_With_WiFI/IMU_With_WiFI.ino

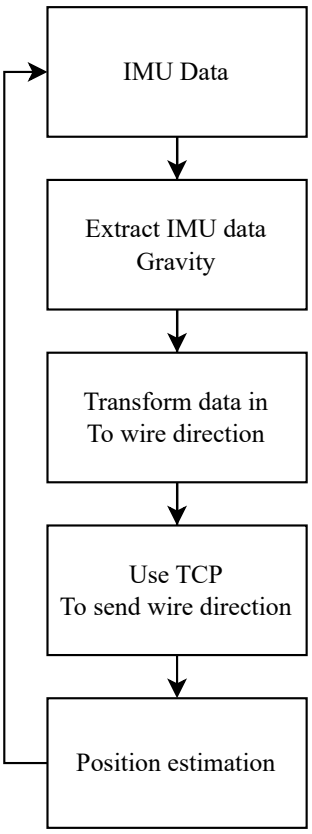


Figure 5.10: Flowchart representing the IMU program

5.4 Motion controller

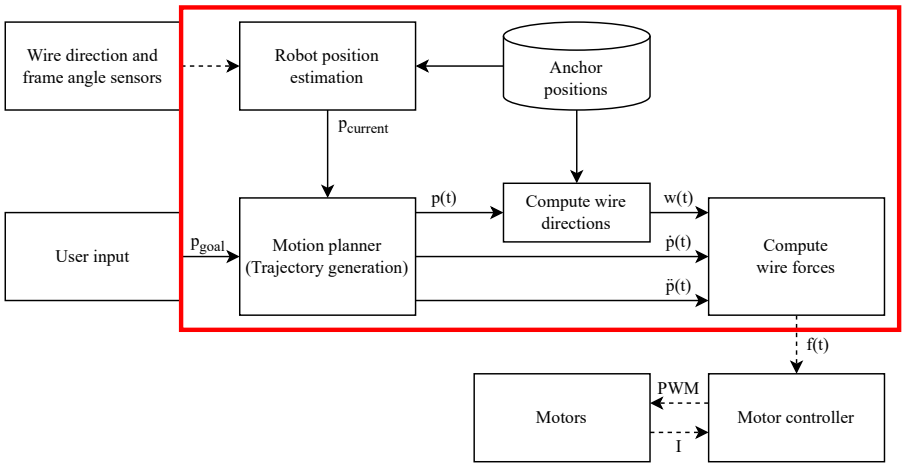


Figure 5.11: The trajectory control in the motion controller pipeline

The following section will cover how the trajectory generation was implemented into the *Motion Controller.py*[1] program. The trajectory is generated from the position estimate towards a specified goal.

A new position for the robot is received as an array containing a XY position together with a rotation around the Z axis as seen in Equation 5.1, where the function trajectory_generation()[2] calculates a trajectory path as a 5^{th}-degree polynomial. As mentioned in Section 4.3 this is done for all 3 movement-axis', namely the X-axis, Y-axis and the rotational movement around the Z-axis. This trajectory is generated from the current position, derived as described in Section 4.2 from the sensor data described in Section 5.3.

$$r = \begin{bmatrix} X_{Position} \\ Y_{Position} \\ Z_{Rotation} \end{bmatrix} \tag{5.1}$$

When this path is generated the required wire forces for the first 100 points in the trajectory are computed as described in the following section, and sent to the motor controller through the TCP connection. Each batch of 100 points contains 100 wire forces for each motor, which the motor controller should follow to move along the trajectory. A new batch is sent to the motor controller at a frequency of 1Hz, since this is the highest frequency with which the force computation can guarantee completion.

[1] https://github.com/Gsvend20/P6_Site_tech_removing_Robot/blob/d65a47f2b7f5972eb14ba05051553358b79310e0/Python/Motion%20Controller.py

[2] https://github.com/Gsvend20/P6_Site_tech_removing_Robot/blob/d65a47f2b7f5972eb14ba05051553358b79310e0/Python/Python_Functions/TrajectoryGenerationFunctions.py

5.4.1 Compute wire forces

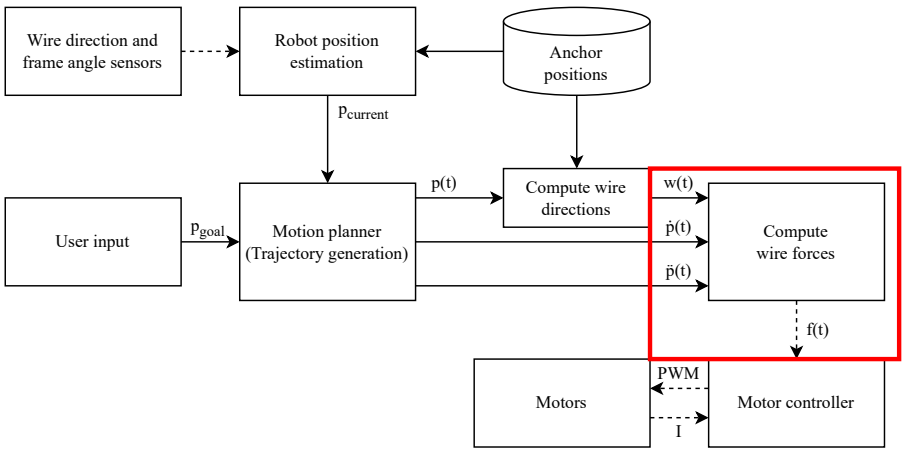


Figure 5.12: Wire force computation in motion controller pipeline

This section explains the algorithm that was used to compute the wire forces needed to achieve the desired acceleration of the frame. As mentioned in Section 4.5 of the Design chapter, finding the suitable wire forces is not a straightforward task as the force and direction of each wire impacts the total forces and moment experienced by the frame. To control the motion of the robot frame, it is necessary to manage the forces along the X and Y axes, and the moment around the Z axis. Additionally, as described in the design chapter, the force in the Z direction, and the moments about X and Y will be nullified by the reaction forces from the wall, if all wires are under tension.

To determine the required forces for each wire, it is necessary to find a solution for $\mathbf{f}_w$ in the systems of equations shown in Equation 5.2 and 5.3, which satisfies the desired force in X and Y, and the desired moment about Z. Balancing the forces of each wire is necessary to achieve the desired total force and moment on the frame. However, there are an infinite number of solutions to this under-constrained problem.

$$\mathbf{F}_{desired} - \mathbf{F}_g = \sum_{i=1}^{5} (\mathbf{f}_{wi} \cdot \hat{\mathbf{w}}_i) \tag{5.2}$$

$$\mathbf{M}_{desired} = \sum_{i=1}^{5} (\mathbf{r}_{wi} \times (\mathbf{f}_{wi} \cdot \hat{\mathbf{w}}_i)) \tag{5.3}$$

Where:
$\mathbf{F}_{desired}$ and $\mathbf{M}_{desired}$ are the forces and moments needed to induce the desired motion in the robot.
$\mathbf{f}_w$ are the five unknown wire forces.
$\hat{\mathbf{w}}$ are the fire unit vectors describing the directions of the wires.
$\mathbf{r}_w$ are the positions where the wires connect to the robot in the static frame.
F_g is the gravitational force experienced by the robot.

A gradient descent algorithm is implemented to find one solution to the under-constrained problem. With the goal of finding a suitable $\mathbf{f}_w$ for any desired set of forces and moments affecting the

frame, an overview of this algorithm can be seen in Figure 5.13.

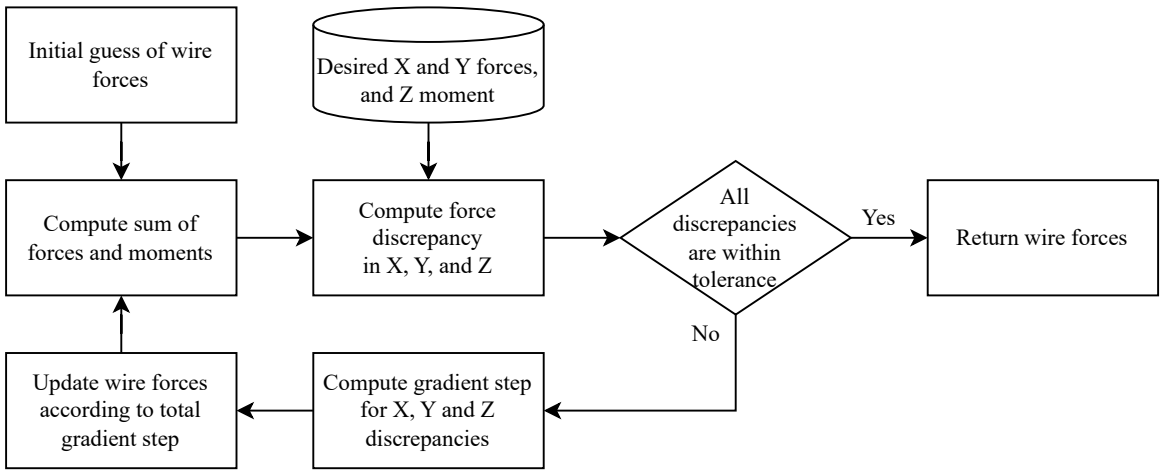


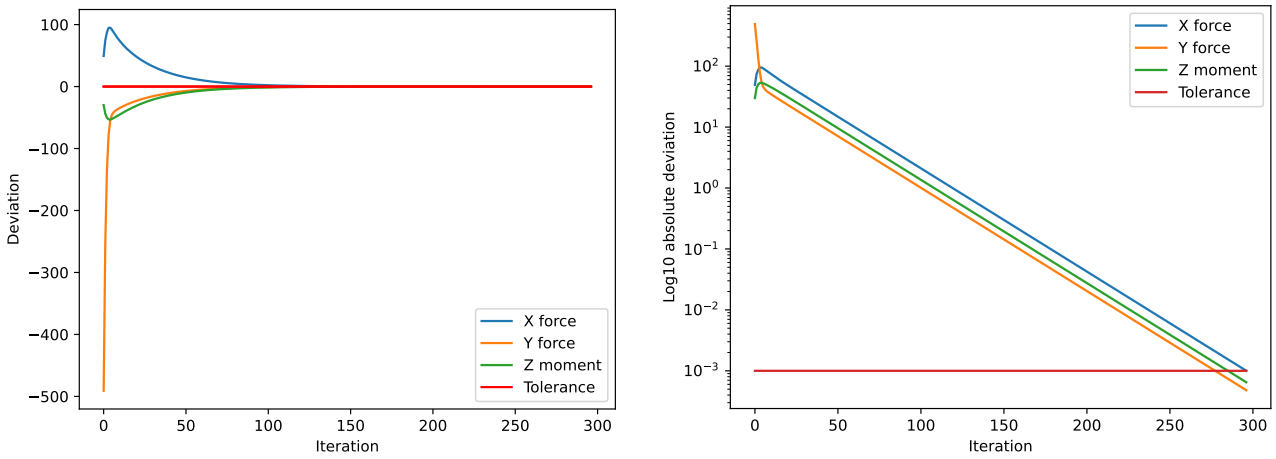
Figure 5.13: Gradient descent algorithm used to find suitable wire forces.

Each iteration of the algorithm the sum of forces and moments acting upon the robot frame are computed for a set of wire forces. The discrepancy between these totals and the desired forces and moments are computed, as shown in Equation 5.4.

$$\varepsilon_{Fx} = Fx_d - Fx_S \tag{5.4}$$

Where:
ε_{Fx} is the frame force discrepancy in the x direction.
Fx_d is the desired frame force in the x direction.
Fx_S is the computed frame force in the x direction, with the current wire forces.

If the discrepancy is acceptable the current wire forces are considered a suitable solution. If it is too large, the algorithm will attempt to update the wire forces to decrease the discrepancy. This process continues until all the discrepancies are within the tolerable limit, as shown in Figure 5.14.



(a) Discrepancies per iteration shown on linear plot

(b) Absolute discrepancies per iteration shown on single-logarithmic plot.

Figure 5.14: The absolute discrepancies decrease each iteration until all are within tolerance. Smaller discrepancies decrease more slowly.

The algorithm determines how to update the wire forces by multiplying the discrepancies by the gradients for force and moment. The gradients are determined by partial differentiation of the force or moment sum equations with respect to the 5 motor forces, as shown in Equation 5.6.

$$F_x = \sum_{i=1}^{5} (\mathbf{f}_{wi} \cdot \hat{\mathbf{w}}_{xi}) \tag{5.5}$$

$$\nabla \mathbf{F}_x = \frac{\partial F_x}{\partial \mathbf{f}_w} = \hat{\mathbf{w}}_x \tag{5.6}$$

Where:
F_x is the x component sum of wire forces.
$\mathbf{f}_w$ are the 5 wire forces.
$\hat{\mathbf{w}}_x$ are the x components of the 5 wires direction vectors.
$\nabla \mathbf{F}_x$ is the gradient describing the relation between wire forces and frame forces in the x direction.

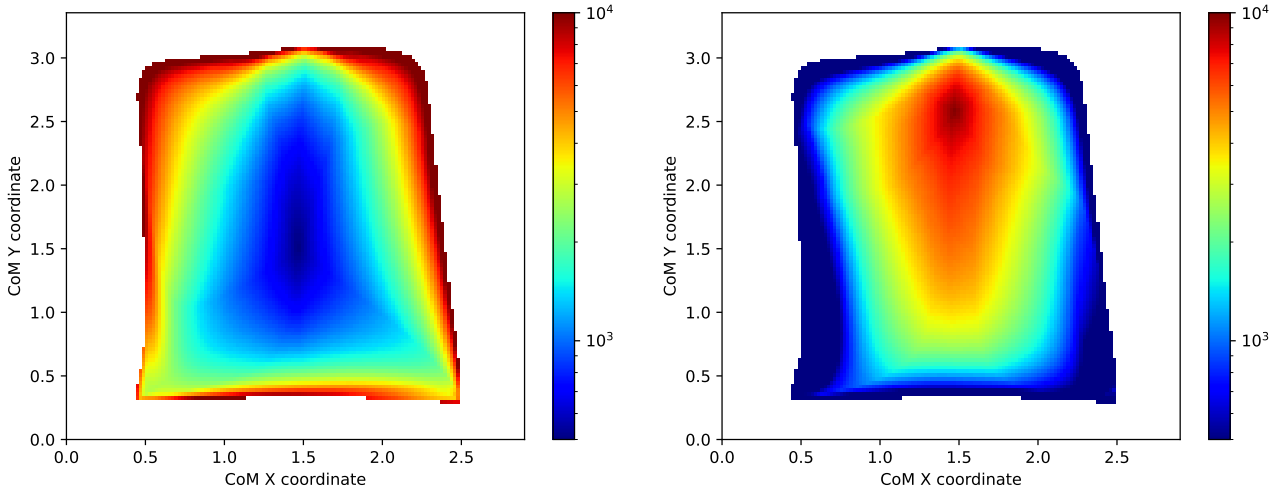
Each of the gradient steps for the X and Y forces, and Z moment are calculated separately and then added together to determine total change to the wire forces, as shown in Equation 5.8. Before this each of the three steps are multiplied by a fraction to prevent overshooting the target, which could cause the deviation to grow. After updating the wire forces according to the calculated step, each of the forces will be constrained within a specified range. This range should correspond to the minimum and maximum forces the motors can produce.

$$\nabla \mathbf{F} = \begin{bmatrix} \nabla \mathbf{F}_x & \nabla \mathbf{F}_y & \nabla \mathbf{M}_z \end{bmatrix}^T \tag{5.7}$$

$$\mathbf{f}_{new} = \mathbf{f}_{old} - \sum_{i=1}^{3} (\alpha_i \cdot \nabla \mathbf{F}_i \cdot \varepsilon_i) \tag{5.8}$$

Where:
$\mathbf{f}_{new}$ are the 5 updated wire forces.
$\mathbf{f}_{old}$ are the 5 previous wire forces.
α are the step sizes for the X and Y forces and Z moment steps.
$\nabla\mathbf{F}$ are the gradients for X and Y forces, and Z moments.
ε are the discrepancies for X and Y forces, and Z moments.

The algorithm starts with an initial guess for the wire forces, as shown in Figure 5.13. As there are many suitable solutions for wire forces to achieve suitable total forces and moments on the frame, these initial wire force values should be set to the lower bound of what the motors can deliver. This biases the algorithm towards solutions with a lower energy state as seen in Figure 5.15. For motion the lower energy state is preferred to minimise stress on the frame. It also requires lower currents to pass through the electrical components, reducing heat buildup.



(a) Maximum wire force when initial guess is 500N

(b) Minimum wire force when the initial guess is 10000N

Figure 5.15: Showcase of two different states the dynamics can compute dependent on the initial force guess. Both give the same resultant forces and moments on the frame. Figure a shows a low-energy state, whereas figure b shows a high-energy state. The white regions do not have a viable solution.

5.5 Motor controller

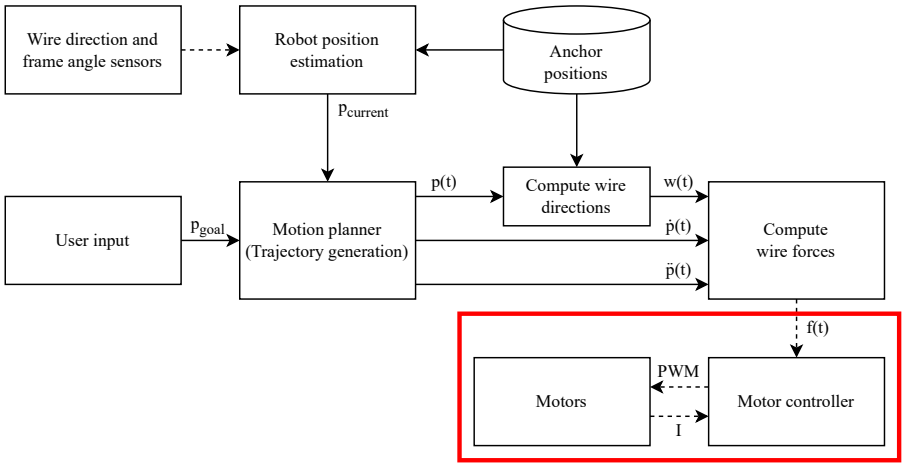


Figure 5.16: Motor control in the motion controller pipeline

The implemented motor control loop for reaching the desired wire forces consists of a central controller and a few key physical components. A single ESP32 micro-controller is used to run the control loops for all 5 motors.

The controller is designed to be directly applicable to the existing Site-Tech solution. The hardware framework related to motor control can, be directly exchanged with similar components rated for higher currents.

The program MotorController.ino[3] is used to control exerted wire forces on all 5 motors. It is crucial for the entire robot that each motor follows the designated force trajectory as precisely as possible. Since the ESP32 has 16 analogue ADC's all sensors and motor Input and Output (I/O) can be connected to a single ESP32. When the ESP32 has enabled wireless communication, 10 of the programmable ADC's are lost which is why a second ESP32 is used solely as a WiFi node, connected to the motor controller ESP32 through a serial connection.

From the WiFi node 100 desired forces are delivered at a frequency of 1Hz, meaning each motor has to update their forces 100 times a second before the next trajectory arrives. The motor controller has to adjust each of the 5 motors to their individual desired forces, with a 100Hz frequency before the desired forces in the trajectories, change to the next index of the 100 force trajectory. An Interrupt Service Routine is set up to precisely update the force trajectory with 0.01s intervals. During the interval, a PI controller ensures the correct current is delivered to the motor, by adjusting the PWM DC to the H-bridge. The controller runs at a set frequency of 100Hz, in which the output is adjusted based on the closed loop sensor feedback.

[3]https://github.com/Gsvend20/P6_Site_tech_removing_Robot/blob/cddf198a8dd8ec5fc30e9fdf5a59fdd54a209139/ESP/MotorController/MotorController.ino

5.5.1 Sensor feedback

The sensor feedback is detected by a Bi-directional Hall-effect current sensor, giving feedback to the motor ranging between 0 supply voltage to max supply voltage The calculations for sensor output to reference current can be seen in formula 5.9. .

$$I_{sensor} = \frac{ADC - ADC_{max} * 0.5}{\kappa} \tag{5.9}$$

Where:
κ is the sensor Sensitivity Factor at 40mV/A.
ADC_{max} is the maximum value detectable on the ESP32 Analogue to Digital Converter (ADC).
ADC is the ADC-value detected from the sensor-voltage output.

The ADC max resolution is divided with 2 since the Bi-directional sensor has its 0 point at exactly half the supply voltage. The ADC value has to be adjusted, since the ESP32 ADC has an imprecise non-linear relationship with supplied voltages, as can be seen in Figure 5.17.

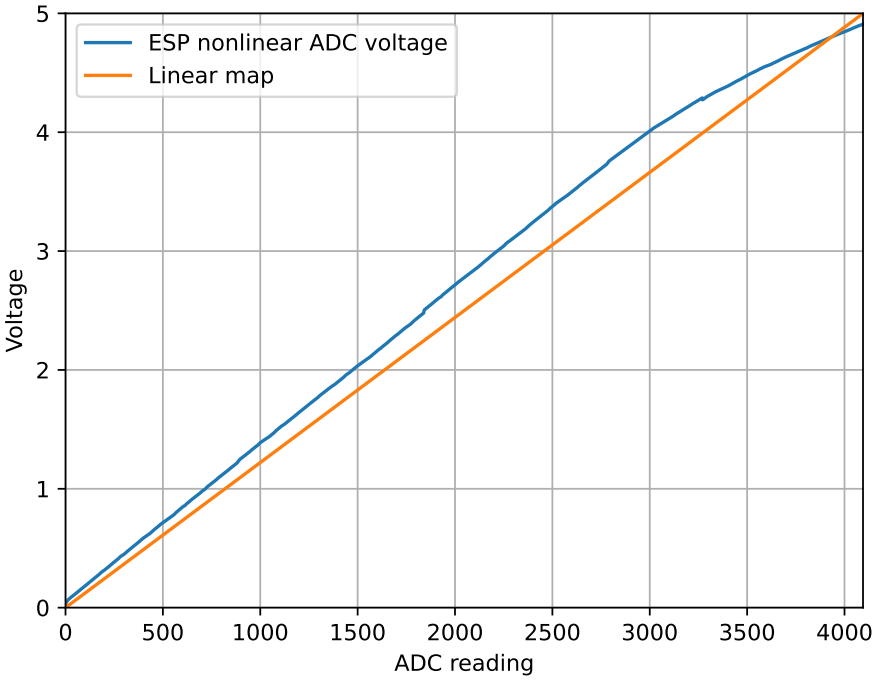


Figure 5.17: ESP32 ADC reading relative to actual voltage

The individual ESP32 ADC used can be calibrated by making a Look-up Table (LUT) consisting of the actual linear resolution, relative to the non-linear ADC resolution. The LUT is generated by connecting two ESP32 analogue pins, and generate known voltages with the ESP32 Digital to Analogue Converter (DAC) and matching them with the ADC resolutions. For stable sensor readings, a 100 index moving average filter is added, taking samples at 100Hz.

5.5.2 Motor Modelling

As explained in Section 4.7, the Back-EMF is considered negligible and the current to force relation can be found through the Motor Torque Constant. A test was setup mapping the relation of the lower bound forces to the drawn current, the test-journal can be found in Appendix B.1. Motor Torque constant of $K_\tau = 77$ is found by linearising the currents to their corresponding forces exerted, as seen in Figure 5.18.

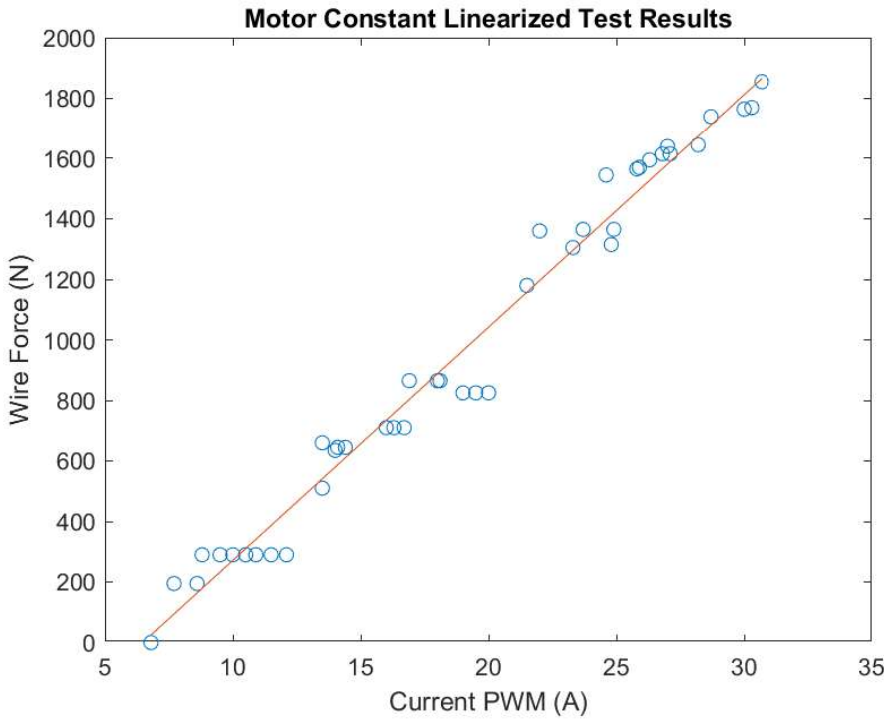


Figure 5.18: Torque to current relations linearised, Relation between Current adjusted through PWM and exerted Wire Force

The motor torque constant becomes close to linear as soon as the motor overcomes it's stall current. The force balance algorithm has a minimum force exertion on each wire on at least 500N, this means the worst non-linearity of the motor torque constant is completely avoided.

5.5.3 PWM modelling

The H-bridge controls the power by adjusting the PWM Duty Cycle, which works like a Voltage. This means the current detected behaves vastly differently than the desired current delivered by the controller. If the sensor data is used as feedback to the desired current, the H-bridge current transformation should be modelled. This is done by linearising the detected sensor current plotted to the PWM Duty Cycle steps spanning from 0 to 254, as seen in Figure 5.19.

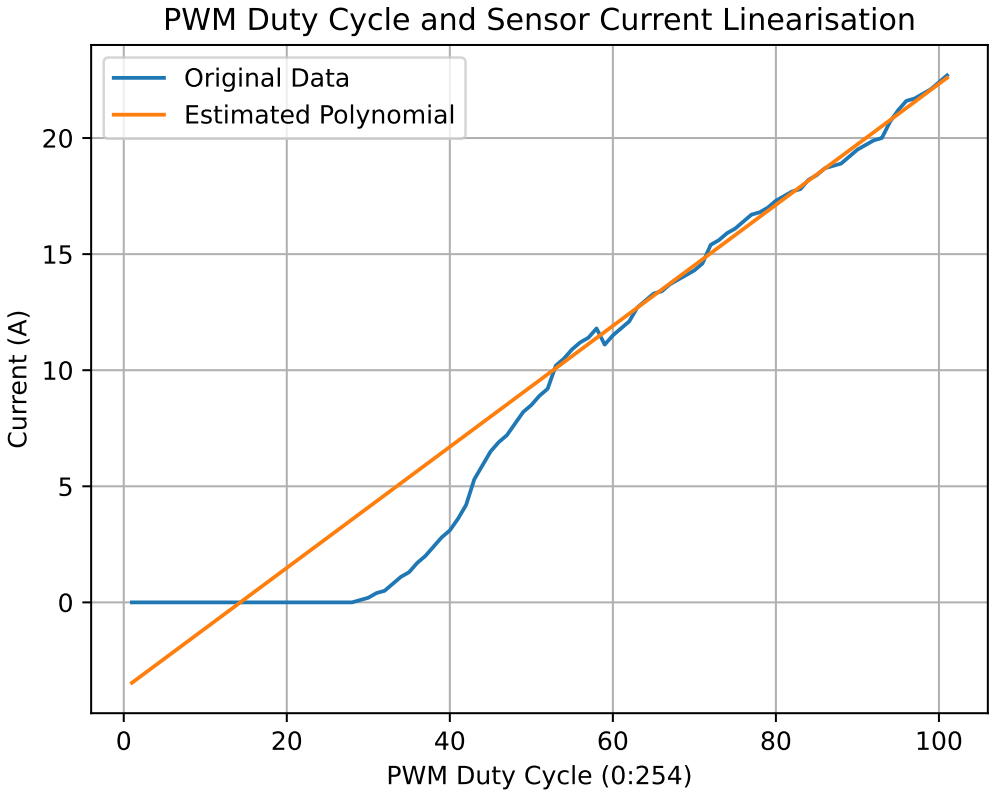


Figure 5.19: Linearisation used for the PWM constant, Duty Cycle ranges from 0 to 254

The slope coefficient of the line, is used as a simplified model for supplying current by adjusting the PWM Duty Cycle. The slope of the linearised model is $K_{PWM} = 0.2606$ and functions as the PWM transfer from Duty Cycle to actual currents. The PWM constant is used to generate the correct PWM for the motor, based on a desired current. The non-linear part is not modelled, as the lower-wire forces and thereby the lower currents, are under the force balance algorithm's minimum force.

5.5.4 PI controller

When implementing the PI controller, anti-windup limits at min- and max 100, are used for constraining the integral. The Kp and Ki value, used, are generated with the MATLAB auto-tuner, and the values: $Kp = 0.0065$ & $Ki = 1.3072$. The constants are tuned for no overshoot and stable rise time, the constants are found in discrete time.

This PWM-coefficient is multiplied by the desired current before being passed to the PI controller, so the desired current matches its sensor current equivalent. Before the adjusted PI output is output as a DC-fraction, the PWM constant is divided in order to achieve the actual desired current from the H-bridge to the motor.

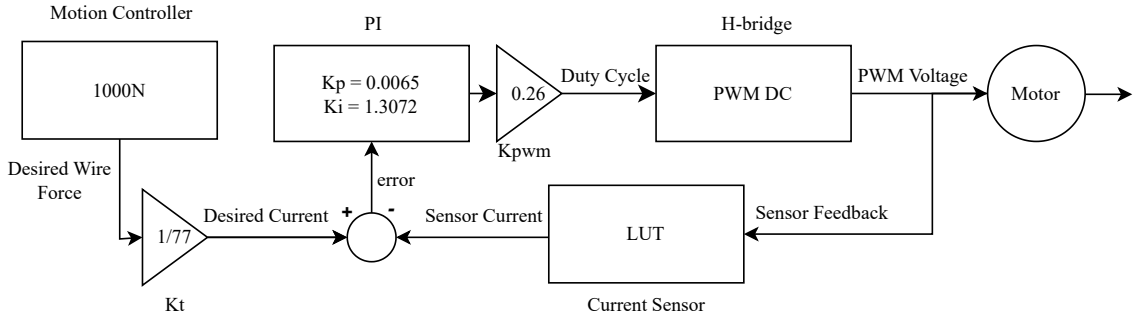


Figure 5.20: Implemented Motor Control Loop.

As can be seen in Figure 5.20, the Motor- and PWM constants are placed before and after the PID respectively. The PI-controller and it's Kp- and -Ki, constants adjusts the error, and the sensor model feeding back the used current. All of this controls the motor and ensures the exerted wire forces reaches the desired level.

After an examination of the components and the software implemented, from the Mock-up model to the relevant software, the full system is ready for testing.

6 System tests

This chapter will test the full system against the requirement described in Section 3. Each following section contains a test described in the acceptance test section 3.2. The results from these tests will help understand the system's reliability and performance.

6.1 Pose estimation

The pose estimation test is based on the acceptance-test found in Section 3.2.1. For each test, the position estimate measured at least 200 measurements to check the impact of sensor noise on the readings. The distance between the robot's mean pose estimation, and the physical pose of the robot, can be seen in Figure 6.1.

If the deviation for positions in X and Y, and around the Z component is less than 0.1 m and 7.5deg respectively, the test is considered passed. As seen in Table 6.1, the absolute positional deviation along Y is consistently too high to be accepted. The angular deviations are all within the requirement.

Position and rotation estimation test			
Test nr	Deviation X (m)	Deviation Y (m)	Deviation Z (Deg)
1	0.120	-0.187	-0.648
2	-0.014	-0.112	-0.674
3	0.072	-0.136	-0.641
4	0.214	-0.148	-0.689
5	-0.018	-0.197	-0.562
6	0.094	-0.162	-0.628
7	0.096	-0.298	-0.693
8	0.226	-0.167	-0.634
9	0.245	-0.257	-0.613
10	-0.006	-0.142	-0.584

Table 6.1: Test results for position and rotation estimation test

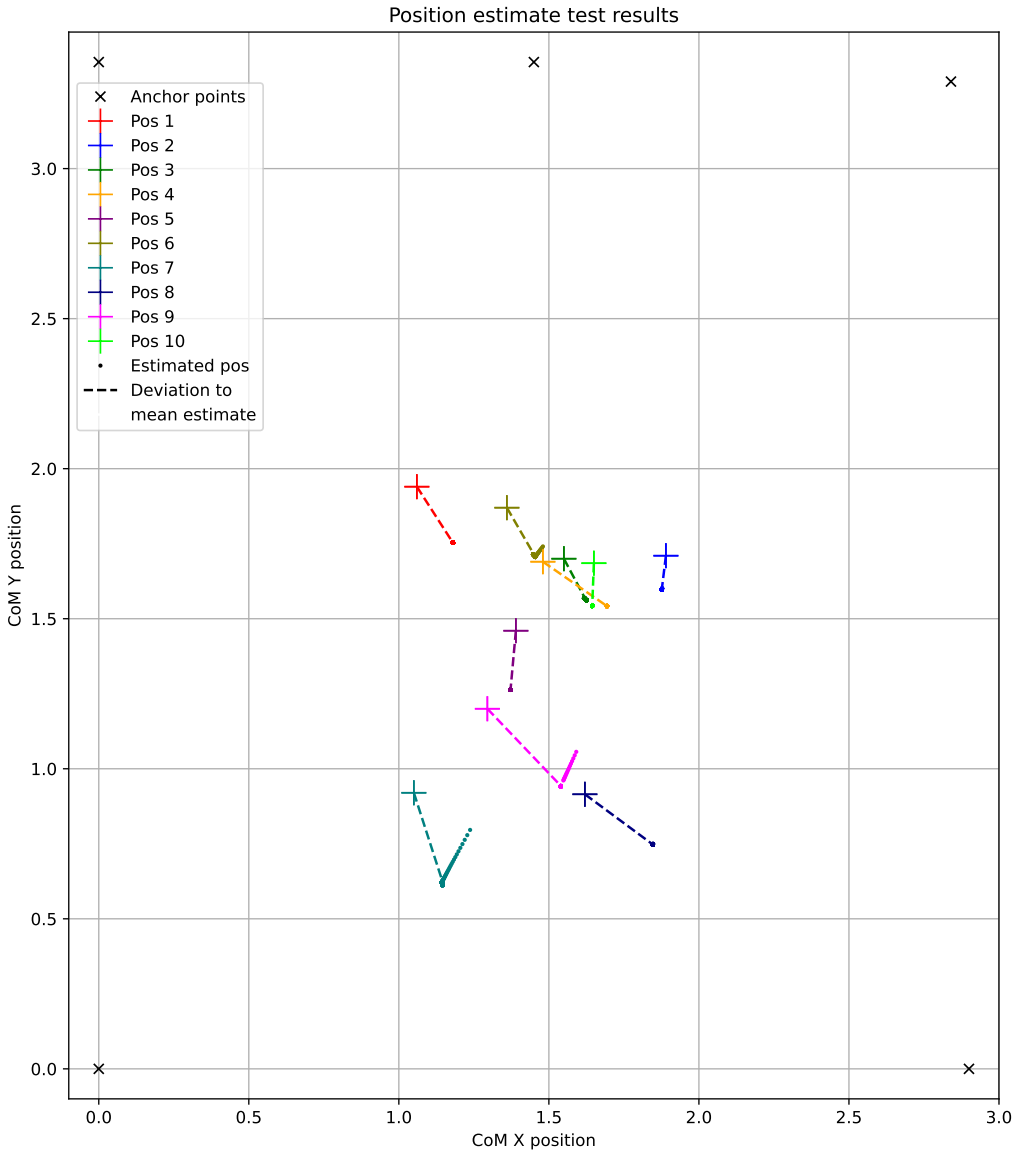


Figure 6.1: Illustration of the test, displaying mean deviation from goal

6.2 Reach goal

This test is based on the acceptance test described in Section 3.2.2. For this test the robot must be able to reach a given goal, within a margin of 0.100m deviation in X and Y distances, and less than 7.5deg deviation for rotation around the Z axis.

Reach goal test			
Test nr	Deviation X (m)	Deviation Y (m)	Deviation Z (Deg)
1	0.3594	-0.0923	-0.0406
2	-0.6111	-0.5411	0.1489
3	-0.4143	0.4307	-0.0573
4	-0.0166	0.4353	0.1792
5	0.0570	-0.1744	-0.0243
6	-0.0427	0.1201	0.1215
7	-0.1292	0.5134	-0.0229
8	-0.0146	-0.2290	-0.0711
9	0.2546	-0.1204	0.1081
10	-0.8163	1.0351	-0.0896
11	-0.1113	-0.5442	-0.1260
12	0.1470	-0.0692	0.0123

Table 6.2: Test results for reach goal test

As seen in Table 6.2, none of the goals were exactly reached, since all tests were stopped prematurely when the robot clearly was moving off course. This can be seen in Figure 6.2.

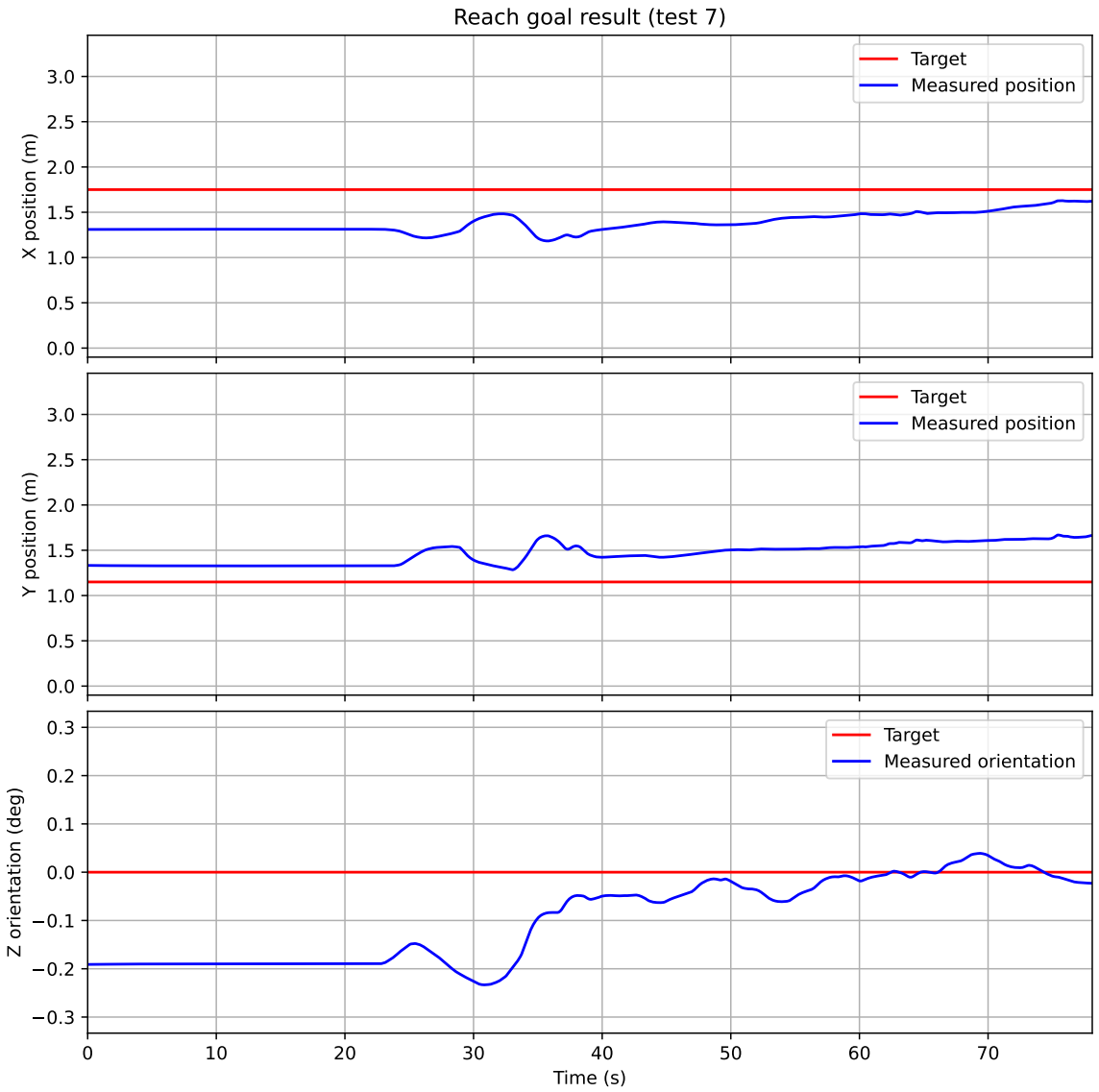


Figure 6.2: Illustration of the test, displaying reach goal for X, Y and Z.

The X and Y values indicate the measured position, and Z represents the measured orientation for the 7^{th} test, which can be seen in Appendix C.1. The X position and the Z orientation were steadily approaching the target values, whereas the Y position of the robot was moving away from the desired workspace.

6.3 Follow trajectory

For the Acceptance test described in Section 3.2.3, the robot must be able to follow a linear trajectory between 2 points. As seen in Figure 6.3, 4 of the 12 tests were evaluated to be passed following the criteria outlined in the acceptance test. The evaluation is based on the positional data recorded by the system during the motion, as seen in Figure 6.3. All trajectories can be found in Appendix C.3.

Follow trajectory test		
Test nr	section	results / meaning
1	C.14	Failed
2	C.15	Failed
3	C.16	Failed
4	C.17	Failed
5	C.18	Passed
6	C.19	Passed
7	C.20	Failed
8	C.21	Passed
9	C.22	Failed.
10	C.23	Failed.
11	C.24	Failed
12	C.25	Passed

Table 6.3: Test results for following trajectory test

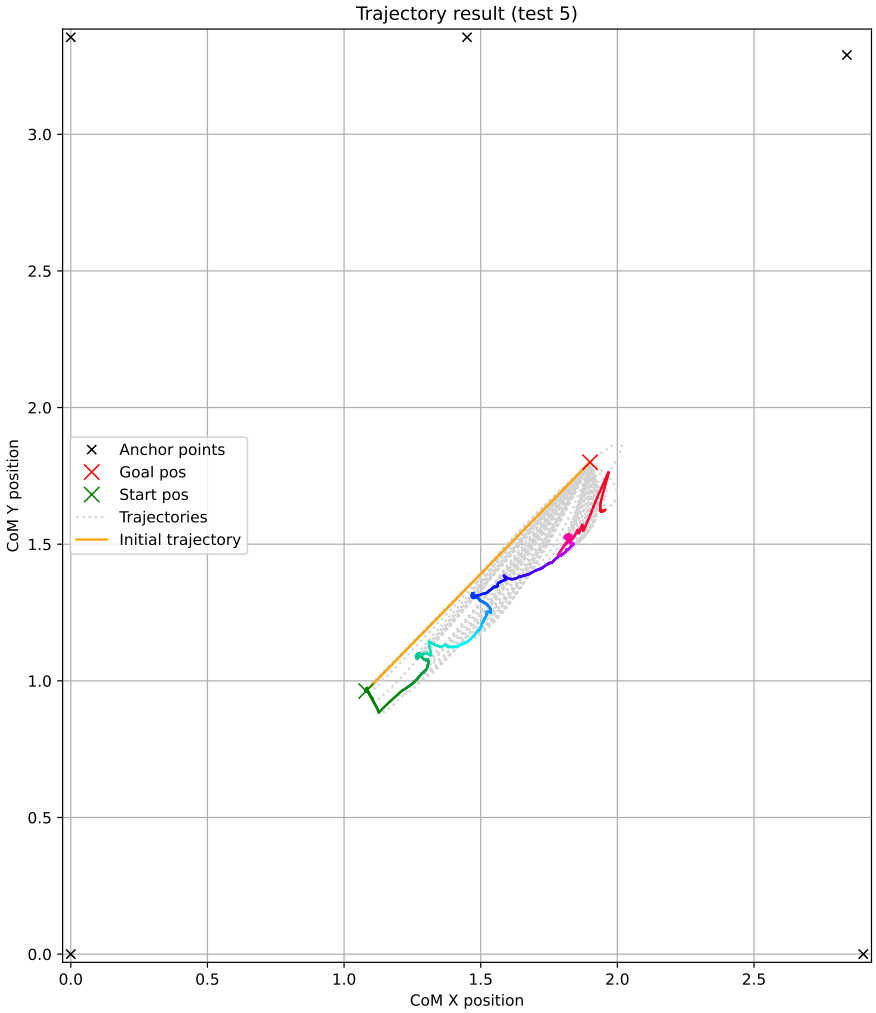


Figure 6.3: Visual representation of the test following the trajectory from the start position, marked green, to the goal position, marked red.

Figure 6.3, is an example of the plotted trajectory, the robot COM followed during tests. The axes indicate the robot position on the test wall, the yellow path is the initial trajectory the robot was tasked with. The coloured trajectory is the actual movement of the robots COM across the wall towards the goal position, where the colour range scaling from to green to red indicates the time used during test. The grey trajectories indicate the continuously updated trajectories to the goal.

7 Discussion

This chapter discusses the processes, findings, and results of the tests performed. And the goal of this project is to further develop Site-Tech's mortar removal robot system. The project has focused on the integration of a motion-controlling system, containing sensors for receiving positional data, a motor controller with PID control and more.

7.1 Tests

Due to the imprecise position estimation, the robot encountered difficulties in reaching the exact desired position and was stopped before the motion was done, to prevent the robot from damaging its components. The tests indicated additional issues where the robot struggles to move in the correct direction towards the goal position, even when considering the offset of the COM, as illustrated in Figure 7.1.

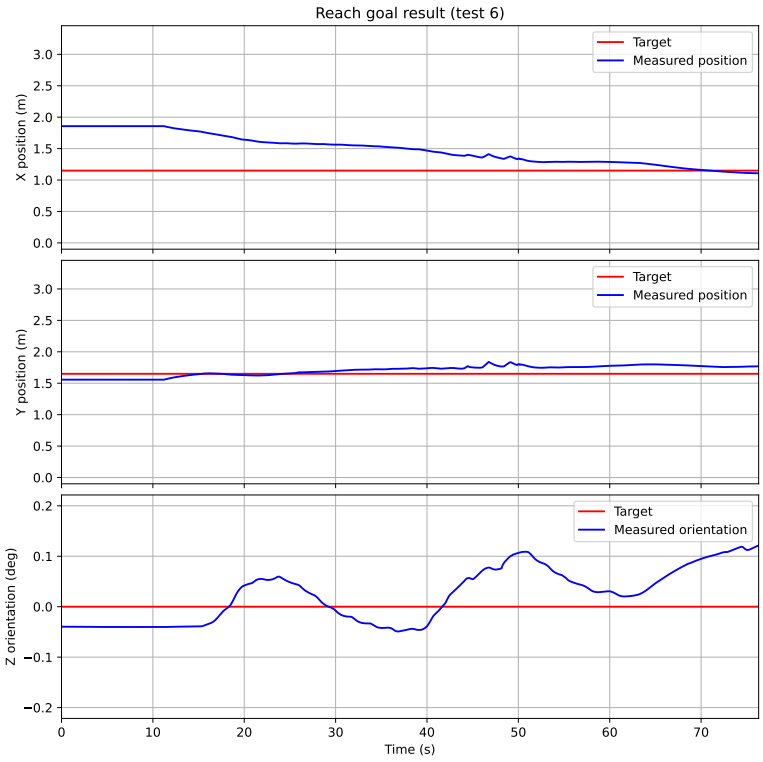


Figure 7.1: Results from test 6, showing the X, Y and Z deviation from target

Because all tests were stopped before the goal position was reached, it had an impact on the test results. In certain cases, tests even had to be stopped as the robot was trying to move off the workspace on the wall, while the robot's position estimation thought it was still moving towards the

goal. Even though the test failed, the test results show the motion controller is able to register the deviation of position for the movement and slowly compensate. As seen in Figure 7.1, the X and Y directions are encroaching on the goal, and could have reached the goal, if the test was not stopped, when the robot was moving toward the border of the workspace.

This observation can also be noted in the comparison between the robot's planned trajectory and the actual moved trajectory. As seen in the test results, 4 out of 12 tests are considered passed, suggesting a successful proof of concept for the motion controller and its error compensation abilities. This encompasses movements wherein the robot travels almost parallel to the planned trajectory as displayed in Figure 7.2, followed by an incorrect movement approximately 0.1m away from the goal position. In the actual trajectory of the robot, it is observed that the robot's position deviates from the target, but the overall movement compensates for these errors.

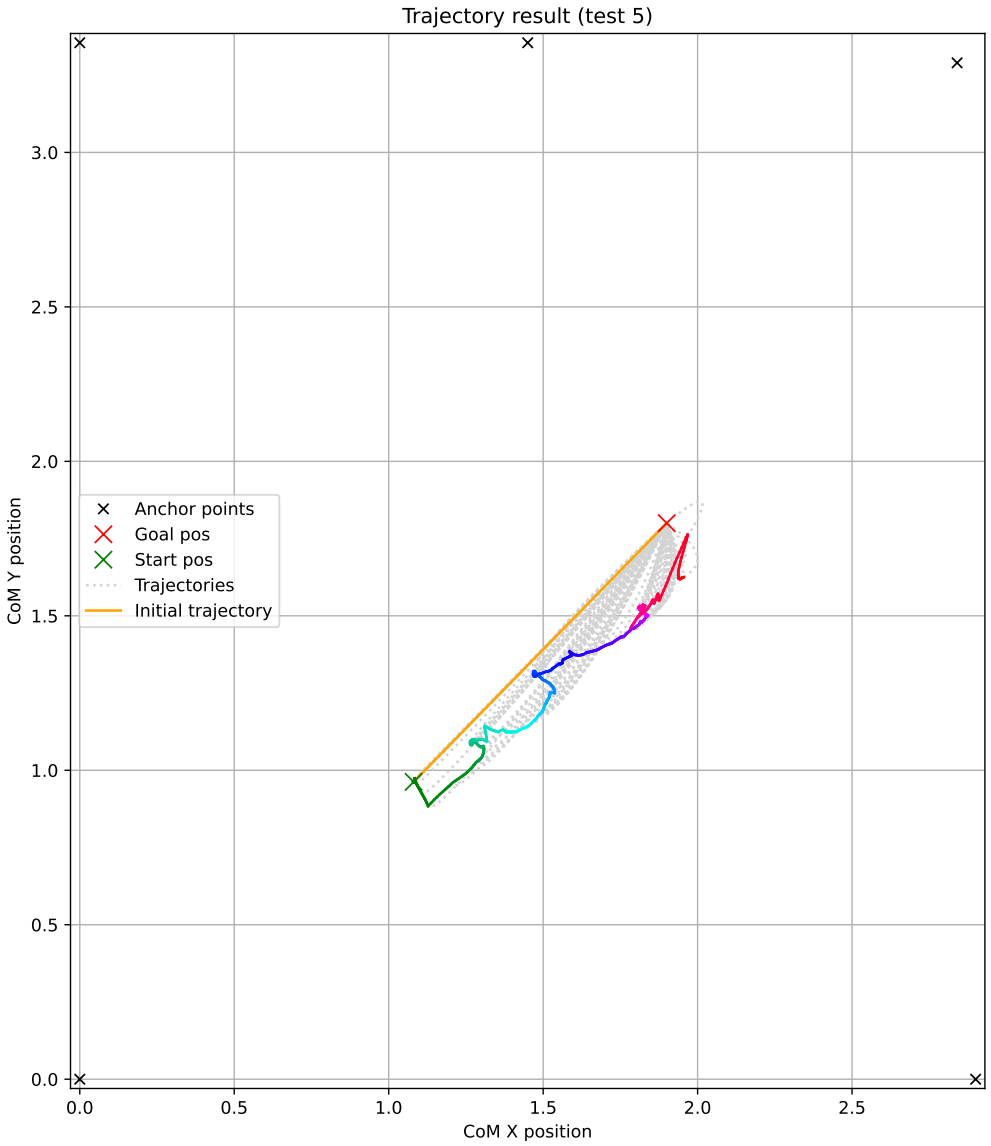


Figure 7.2: Results from test 5,showing the Trajectory estimations, as grey and multi-coloured, with the planned trajectory in orange

7.2 Hardware

7.2.1 Mock-up model

The mock-up model was made smaller, with the motors placed closer to the wall than Site-Tech's current solution. This effectively reduced the amount of force the motors needed to provide to move the robot, by reducing the size of the z-component in the wire directions. With the limited components available for the mock-up model this expanded the workspace the model could travel through, before the required forces demanded more current than the motors could deliver.

An error was made during the Mock-up building process, as the mock-up had to be as compact as possible. This made the motor placements tilt the frame by about 0.12 rad in an anti-clockwise direction Despite the issues with the tilting, it did not seem like an impactful issue during testing, as a simple offset is added to the COM which makes up for the error.

The motors which are used in the Mock-up, auto-locks, which means if the wire pulls with a certain force while the opposing wire pulls with a higher force, it will not be able to pull out the first wire. While making the force balance algorithm, this issue was not accounted for and the algorithm did not function. Therefore it became a significant issue that had to be solved.

The wires exiting the motors in the top corners are used for estimating the robot's position. The wires function as a rigid body whenever they are pulled. A soft body is attached to the wire, making the IMU point in the wire direction. The soft body makes the IMU follow the wire perfectly, as long as the wire is in a rigid state, which makes it effective for estimating the robot pose.

7.2.2 Pose estimation

The sensors' position and orientation estimation for the robot functions well due to the IMU placement. The pose estimation is closely aligned with the actual position of the robot if you account for the offset in X and Y, found in the pose estimation test 6.1. However, if the wire becomes loose, the estimation of the position is completely off. The pose estimation is very useful for determining the start position in a workspace with defined anchor points whenever the wires with the IMU are rigid. It may be possible to account for the Y offset, as it is always in the same direction. By accounting for offset, it also proves to be particularly beneficial in acquiring the robot's starting position. It also includes updating the robot's position after an anchor point moves to a new position.

The robot COM utilises measurements derived from SolidWorks, where a frame containing the motors is designed to mirror the real robot's dimensions. Due to the unavailability of the motor in SolidWorks, a metal block of equivalent weight and size is substituted, which is potentially leading to an inaccurate estimation of the COM. As observed from the results, the robot's estimation of position compared to the measured position, deviated with a constant of at-least -0.1m for the Y axis at all times.

Due to the weight estimation performed on the SolidWorks Mock-up model, the position of the

robot's COM differ from the physical robot. As a result, both physical and sensor-based position estimations may become inaccurate.

7.3 Software

7.3.1 Trajectory Generation

By using a fifth-order polynomial for trajectory generation, the acceleration becomes continuous and smooth. The algorithm which determines the required duration works well with the acceleration limit and can be directly applied in Site-Tech's current version. Although breaking the motion up into three parts: acceleration, constant speed, and deceleration, may be beneficial to also impose a velocity limit.

7.3.2 Force Balance

The force balance algorithm has trouble in certain regions of the workspace, while in other areas, it fails to find any feasible solutions mainly due to the inability to correct for moments.

The slow algorithm can be attributed to constraints in the acceleration calculations and the number of iterative processes involved. The iterative algorithm uses gradient descent for computation, with a maximum iteration limit of 5000 to prevent excessive processing time. While effective, this method is not the fastest, and alternative approaches may converge more quickly.

As a result, the force balance algorithm becomes the bottleneck of the entire workflow. However, it functions as intended and meets the requirements. One notable limitation is its inability to generate negative forces since the motors cannot unreel wires for pushing. To address this issue, wire velocity must be derived from the trajectory and used in the calculations.

7.3.3 Motion controller

The motion controller is able to generate trajectories based on the position estimates, it generates the full trajectory each time.

At first, the whole trajectory was used to generate the torques for the motors, but it later changed to only using the first 100 points to generate the torques for the motors which allowed the calculation speed to be reduced down to around 1Hz, the program can update faster, which reduces the error margins. The continuously updated trajectory, made the motion controller better at correcting errors from non-linear parts, such as the wheels and other system models. This can be seen in the test results, as the robot attempts to hone onto the target trajectory, but the motor modelling in the motor controller can give the motors inaccurate inputs, therefore sometimes making it impossible for the robot to follow the generated trajectory.

7.3.4 Motor Modelling

The Motor Controller which is responsible for applying the desired forces is experiencing malfunctions, leading to various system issues associated with Motor modelling. The accurate modelling of the motors presents challenges due to their size and power. The torque constant adequately represents forces above the stall torque, but it fails to consider higher currents.

Due to the nonlinear behaviour where the motor would lock itself at low currents, a static wire unreeling current had to be introduced. Wire unreeling was completely disregarded in the modelling process due to the excessive currents and motor powers, which exceeded the capabilities of the AAU facilities. Although line speed is not modelled, a static unreel is implemented for negative wire lengths, with any modelling errors left for the motion controller to fix.

The current-controlled model developed, proved to be imprecise, as it could produce varying forces for the same current input. This inconsistency can be attributed to factors such as internal motor resistances and wires becoming interlocked on the roll, resulting in different exerted forces. This inaccurate current feedback, stemming from the linearisation of the model, leads to unpredictable motor behaviour. In general, the current control model falls short and could be improved by incorporating motor speed feedback obtained from back EMF calculations. A more robust solution for future work could involve implementing servo-control to enhance the motor control system.

7.4 Further work

Motors
In order to improve the motion of the robot, the use of servo-control instead of force control should be investigated. The sensors may be more expensive, but accurately modelling the motors for force control is difficult due to the large nonlinear region at low motor currents. The encoders needed for servo-control could be attached to the wire in order to measure the change in wire length, with this information, it becomes possible to move around more accurately compared to current-controlled motors. Further testing is required to ascertain whether these components can enhance the precision of the robot's movements, and based on the results it will be possible to switch motor control systems.

Sensors
The implemented IMU solution, is generally successful and the robot position estimation is adequate. Since the IMU's are only used as gravity detectors, weighted potentiometers can be used with the proposed soft body IMU holder instead to avoid drift. This is due to some of the IMUs' information being redundant, and it makes sense to implement a cheaper and more simple option.

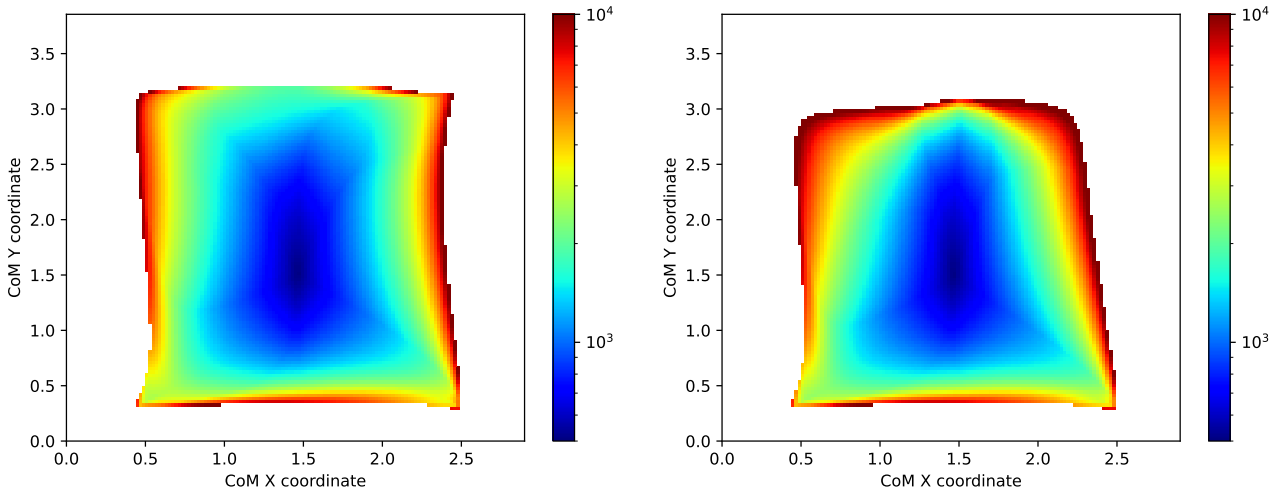
Sensor fusion
Servo-controlled motors and the implemented robot position estimation would work well with sensor fusion. Techniques, such as Kalman-filtering, would complement the trajectory planning and handle the estimated robot position challenges such as temporally slacked wires.

A fused solution with the robot's position estimate can assist a servo-controlled system, by estimating the robot's starting position. This is a useful feature since a servo-controlled robot should have its exact starting position known before use. A challenge that can appear on a wall is the anchor positions have to move to avoid an obstacle, by using the fused solution it will adapt to the new workspace.

7.4.1 Site-Tech's existing solution

Based on the results of the testing, and the findings of the report, there are some essential takeaways that can be useful for Site-Tech, they can utilise some of the systems as they are and incorporate them into their own systems. The first system to look into would be the estimate robot pose, the system functions well, and it can reliably output the correct position of the robot. The Motion controller also proves to be useful, Site-Tech should be able to implement most of the motion controller, but the force balance algorithm has to be amended to be compatible with the proposed servo controller.

Some other useful information for Site-Tech could be the findings about the anchor placements, moving the middle anchor position higher up, opens up a lot more of the workspace near the top of corners. Figure 7.3 shows how the workspace changes if the central anchor-point is moved 0.5m up, compared to if it is parallel to the top corner anchor-points. In Figure 7.3b there is not a lot of accessible workspace near the top corners, whereas Figure 7.3a shows more accessible workspace which is due to the gravity compensation being handled more by the central wire, because of the elevated anchor-point.



(a) The top centered anchor point is raised 0.5 meters above the 2 anchor points in top corners.

(b) The top centered anchor point is the same height as the 2 anchor points in the top corners

Figure 7.3: Illustration of the highest wire force required for positions throughout the workspace.

8 Conclusion

The periodic renewal of mortar between bricks is a labour-intensive task that poses physical challenges for the workers. In order to address this issue, Site-tech is actively developing a mobile platform that automates and enhances the mortar renewal process, but the robot has to be manually moved around the wall.

The objective of this project was to enhance Site-Tech's prototype by enabling the robot to autonomously reposition itself on the wall using the existing wiring system. To achieve this goal, a motion controller was introduced. This controller was capable of planning the required motion and calculating the necessary forces for each wire to ensure the robot followed the planned trajectory.

Despite sub-optimal sensor data, the motion controller demonstrated the ability to adapt partially to errors and plan a path to compensate for them. This project successfully showcased a proof of concept for the motion control of the robot while also identifying areas that require further improvement.

Overall, this research contributes to the advancement of the Site-Tech robotic solution for mortar removal, showcasing the potential for automating and streamlining the manual process of repositioning their robot. Further development and refinement are necessary to optimise the motion control system and overcome the identified areas of improvement.

Bibliography

[1] Luciano Manelli. *Introducing Algorithms in C: A Step by Step Guide to Algorithms in C*. Apress. ISBN: 978-1-4842-5622-0. (accessed: 31.03.2023).

[2] MB Makina. *30x30 Sigma Profile*.
URL: https://www.mbmakina.com.tr/urun/30x30-sigma-profil.html?dil=en. (accessed: 09.05.2023).

[3] Vidaxl.com. *Electric Winch 3000 lb with Plate Roller Fairlead*.
URL: https://www.vidaxl.com/e/electric-winch-3000-lb-with-plate-roller-fairlead/8718475863472.html. (accessed: 18.05.2023).

[4] Reichelt elektronik. *Knuth: Computers and Typesetting*.
URL: https://www.reichelt.com/de/en/closed-loop-stepper-motor-nema-23-1-8-4-a-3-6-v-act-23ssm2440ec-p237920.html. (accessed: 28.02.2023).

A Appendix

A.1 Wire motor

The motor used for the wires is the Electric Winch 3000lb with Plate Roller Fairlead[3]
Max. line pull: 3000 lb
Max. current: 145 Amp
Motor: 1.2 HP, DC 12 V permanent magnet
Gear ratio: 153:1
Gearing: Differential planetary
Wire rope length: 30’
Winch weight: 12 lb
Net weight (accessories included): 19.8 lb
Dimensions: 11” x 4” x 4” (L x W x H)
Drum diameter: 1”

A.2 Gantry motor

The motor Site-Tech uses for the grinder is the Closed-Loop Stepper Motor 23SSM.[4]
Step angle: 1.8.
Motor Length: 133mm.
Rated Voltage: 3.6V.
Rated Current: 4A.
Phase Resistance: 0.9Ω
Phase Inductance: 3.8 mH
Holding Torque: 2.8 N.cm
Detent Torque: 12 N.cm max
Rotor Torque: $800 g.cm^2$
Resolution: 1000

B Links

B.1 Google Drive

Link for Motor Torque Test:
https://docs.google.com/document/d/1sShOhcmaM6uyVXg0wnij71iaG1jYiULu0C2c2xF4aek/edit?usp=sharing

Link for Videos of Reach goal- & Follow trajectory Tests:
https://drive.google.com/drive/folders/11jhSJCBNL-74CEyVDXQAsbCczgMFXw4G?usp=sharing

B.2 Github

Link for Github:
https://github.com/Gsvend20/P6_Site_tech_removing_Robot

C Test results

C.1 Pose estimate accuracy

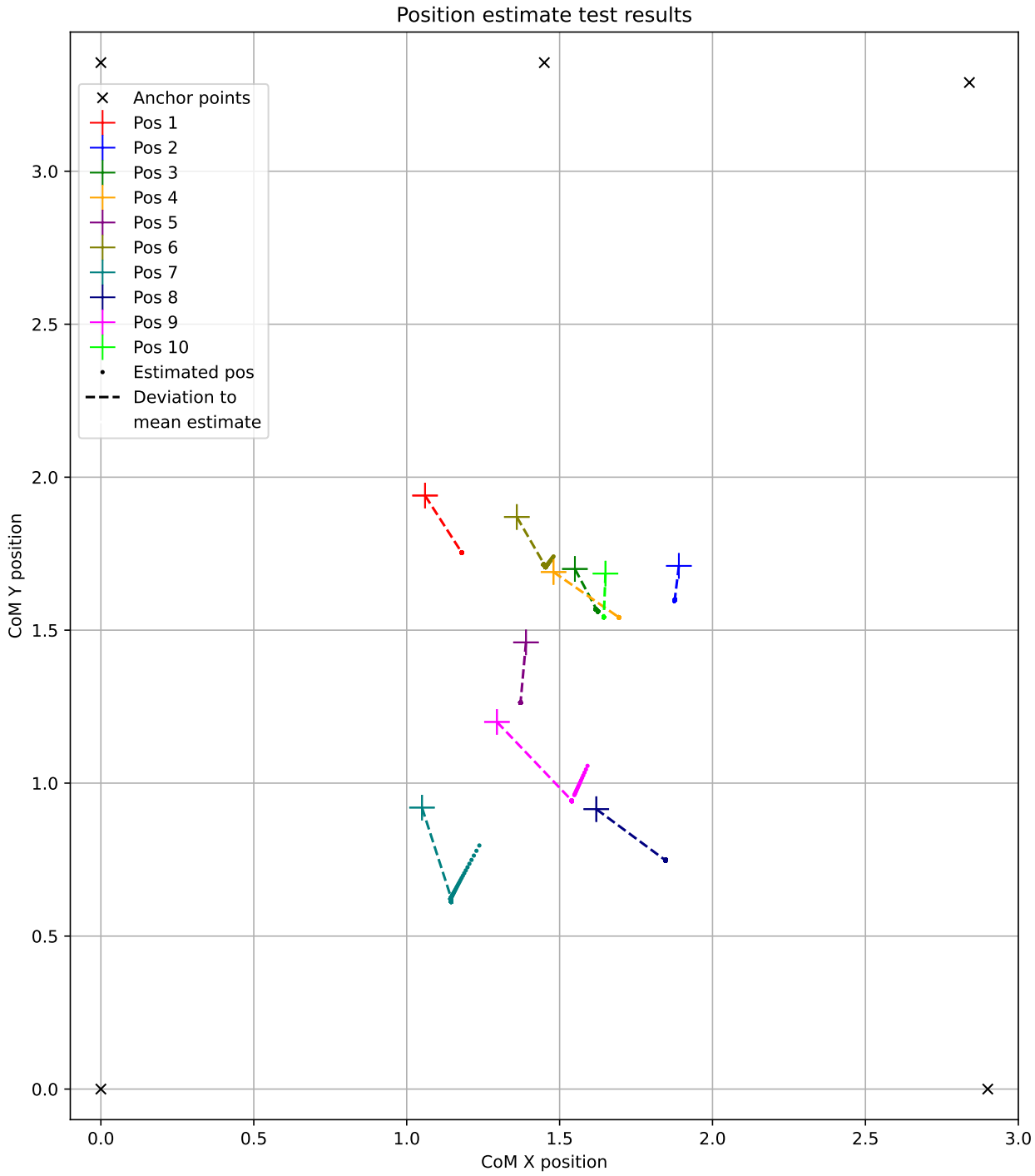


Figure C.1: Results from the position estimate accuracy test

C.2 Reach goal

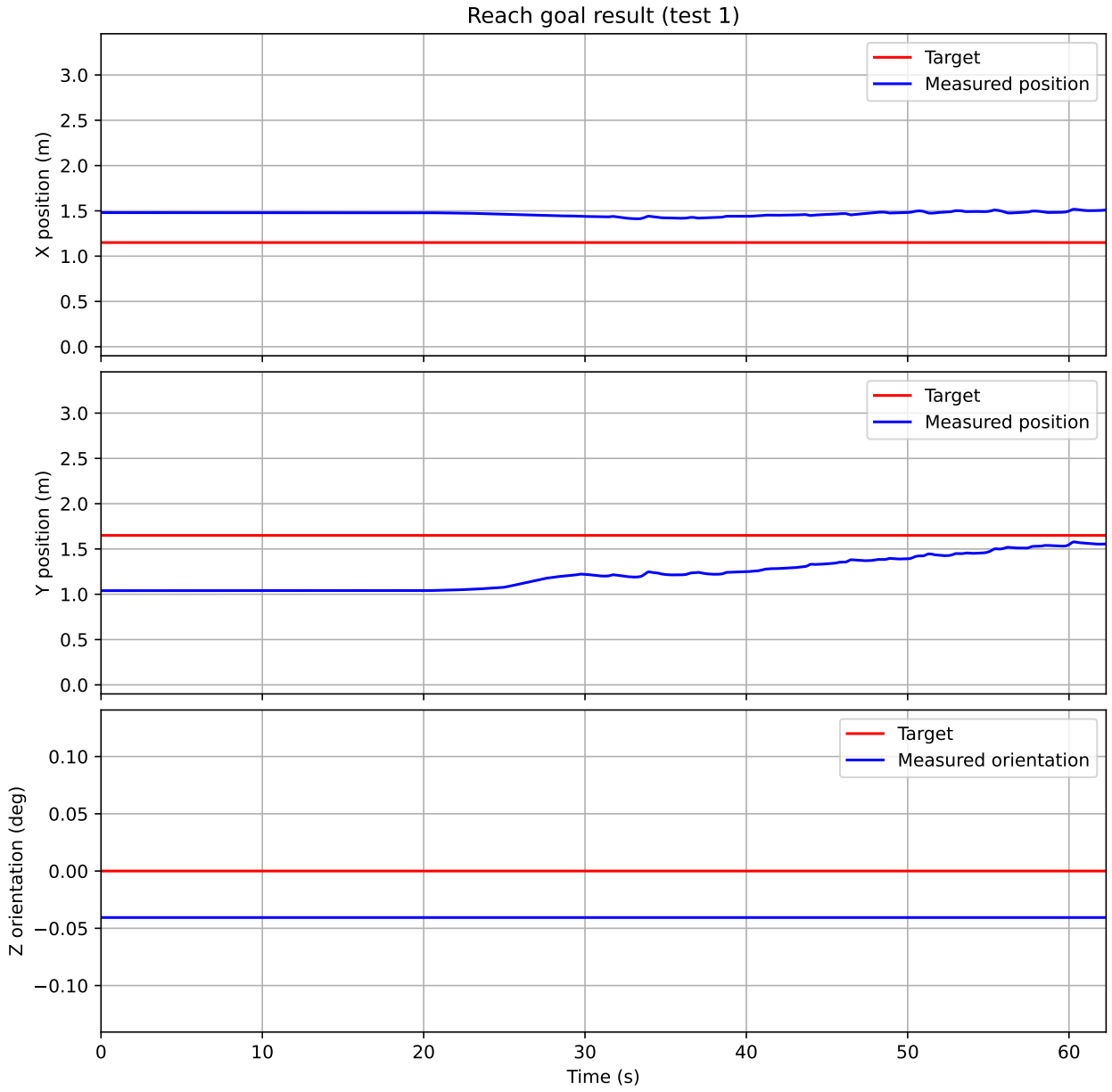


Figure C.2: Results from test 1

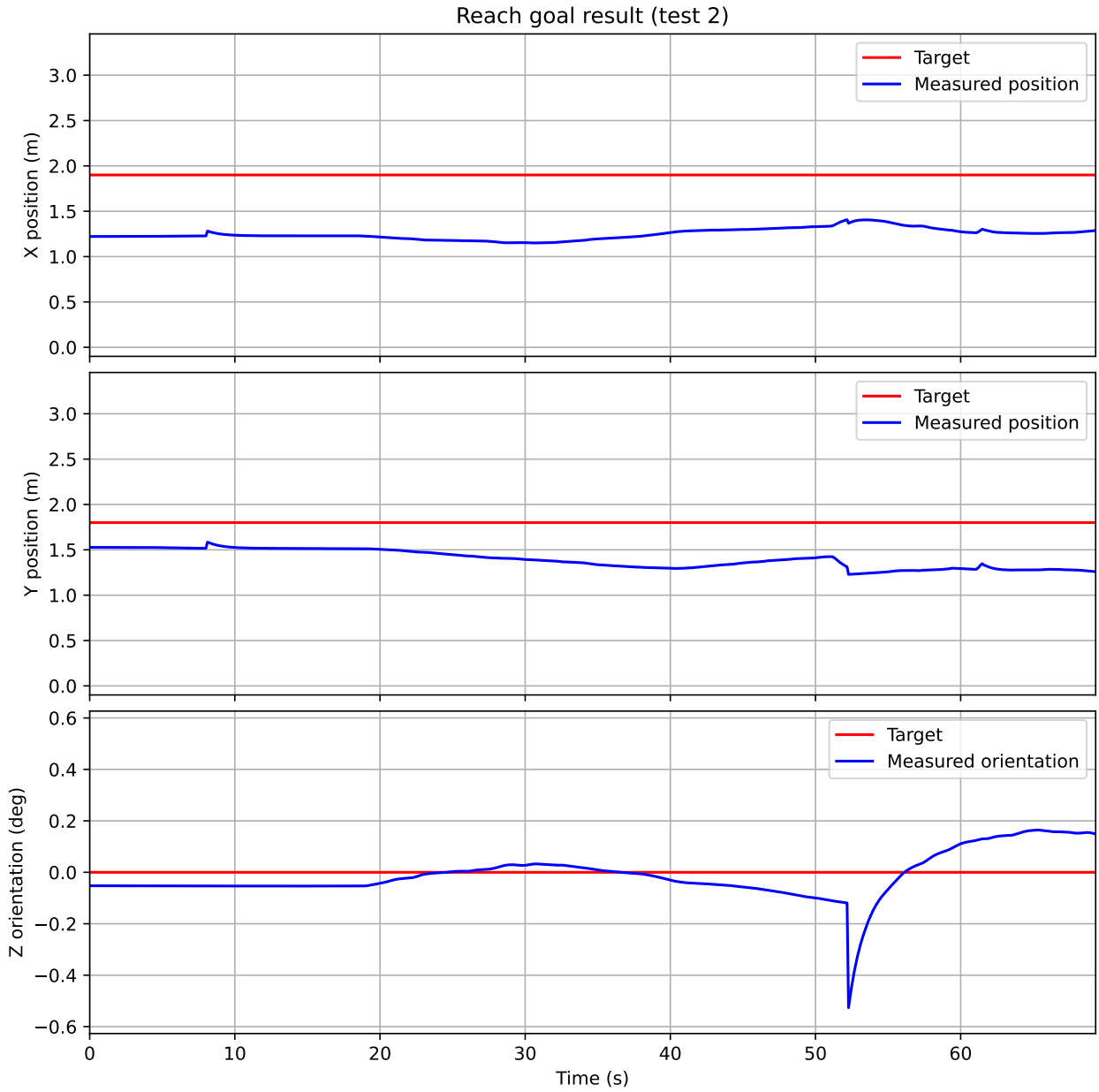


Figure C.3: Results from test 2

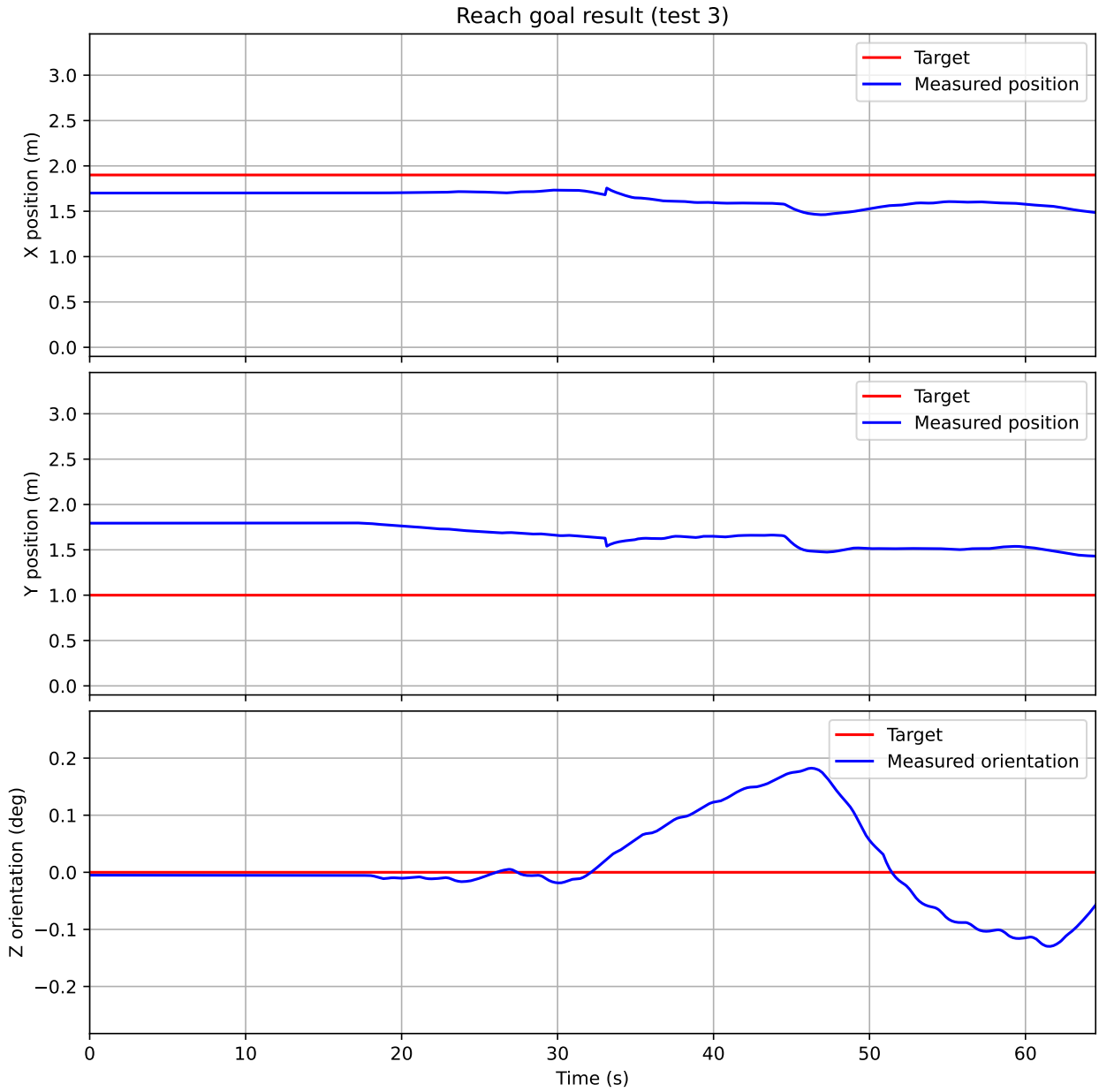


Figure C.4: Results from test 3

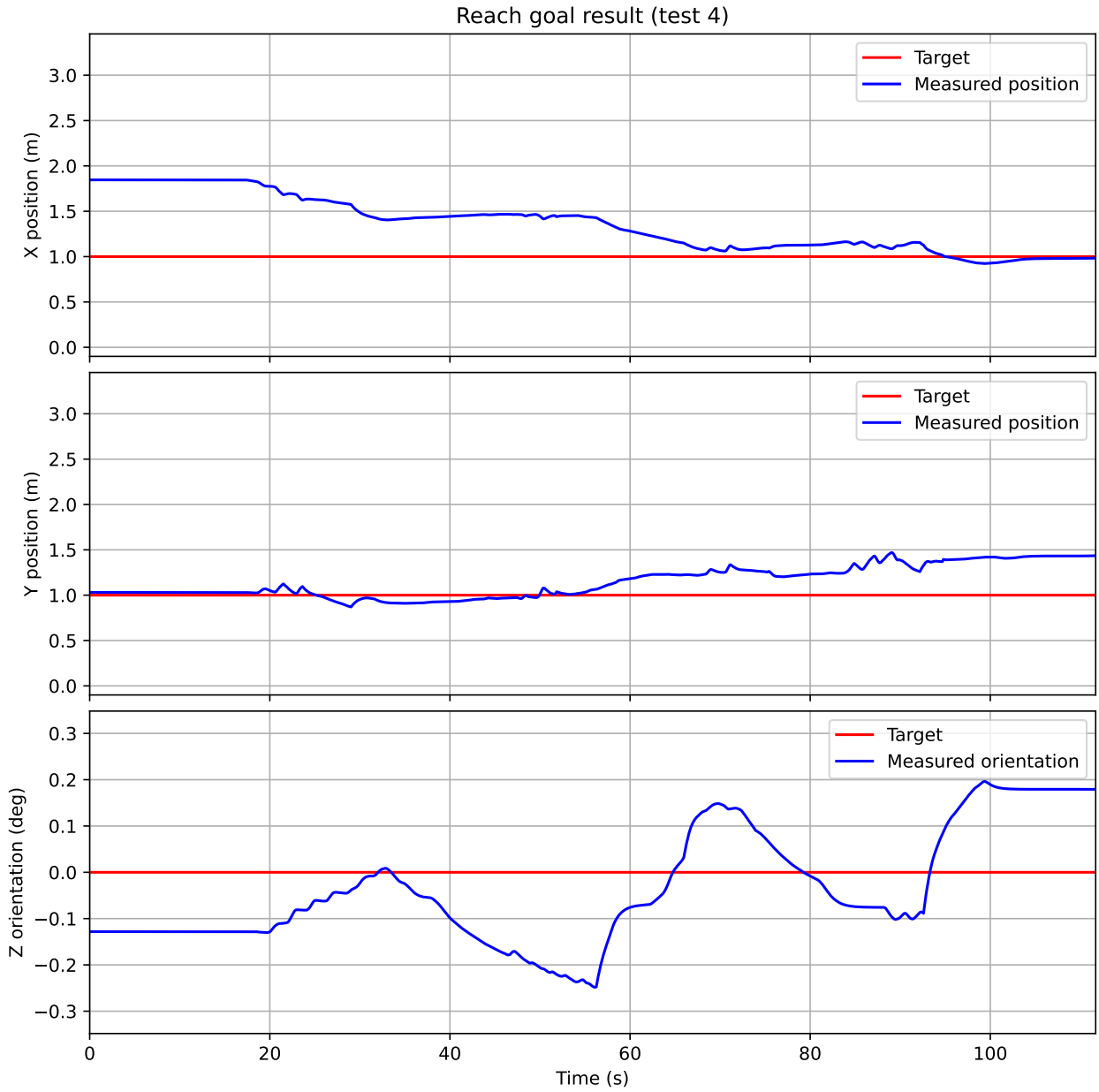


Figure C.5: Results from test 4

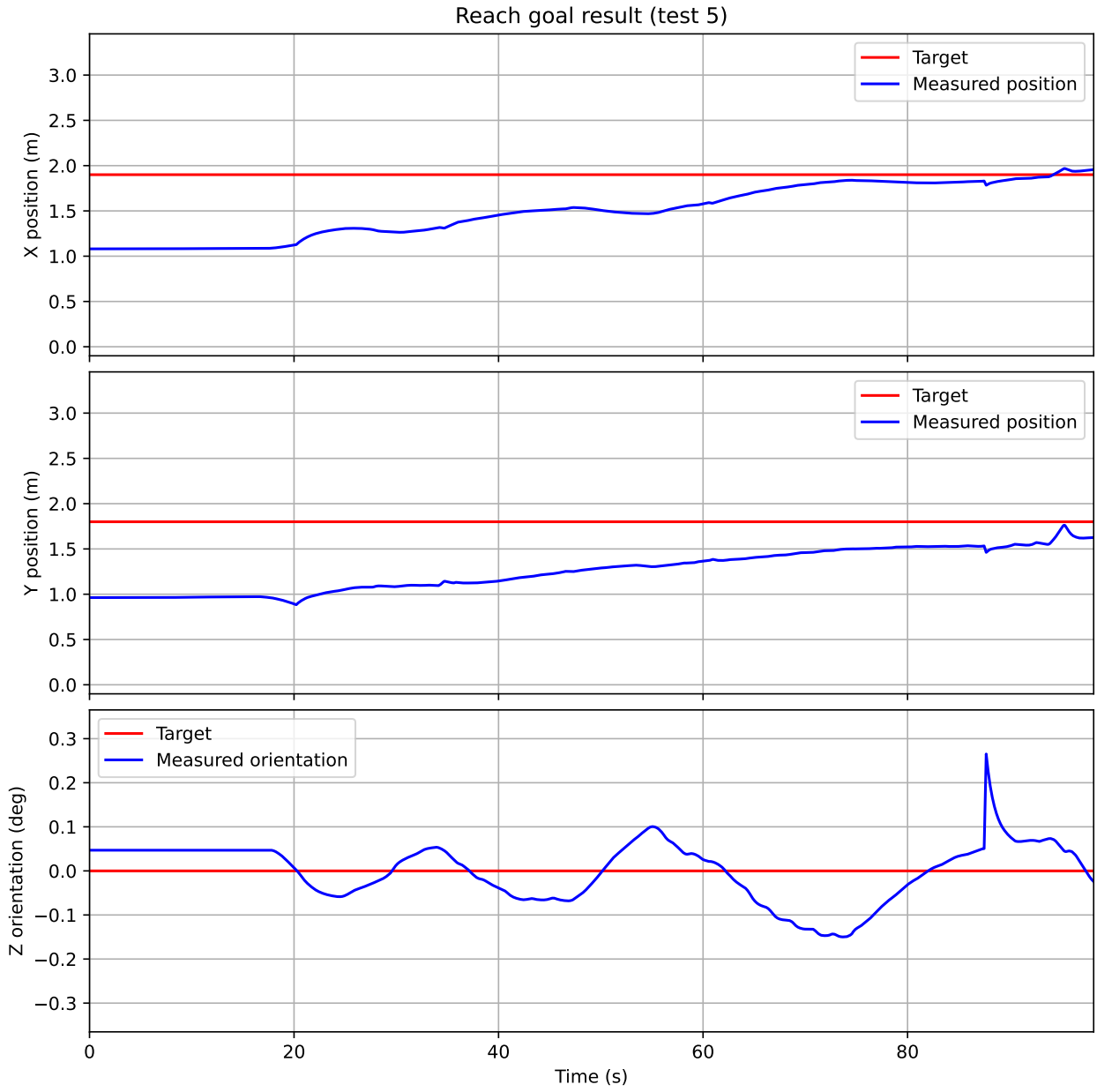


Figure C.6: Results from test 5

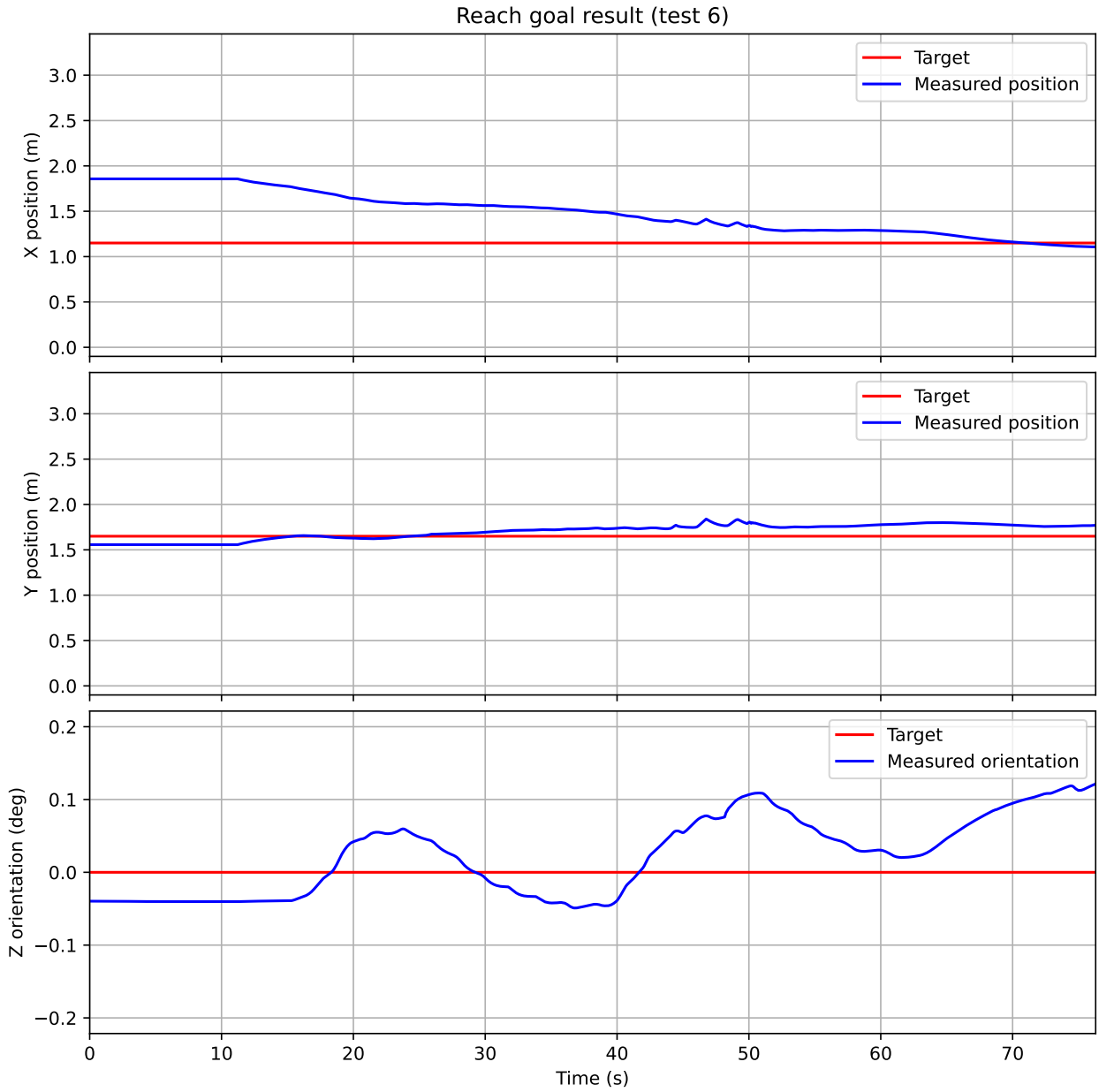


Figure C.7: Results from test 6

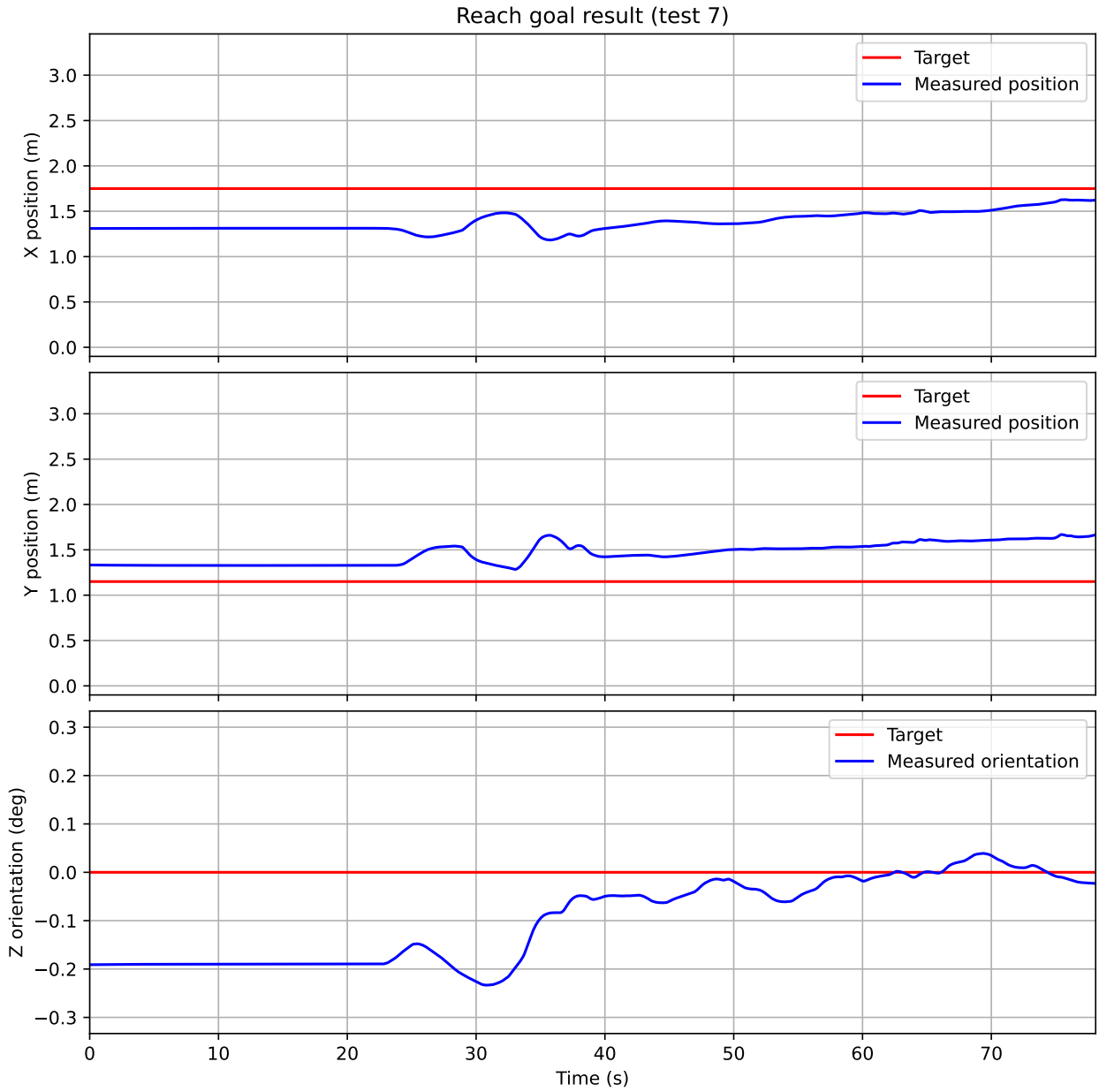


Figure C.8: Results from test 7

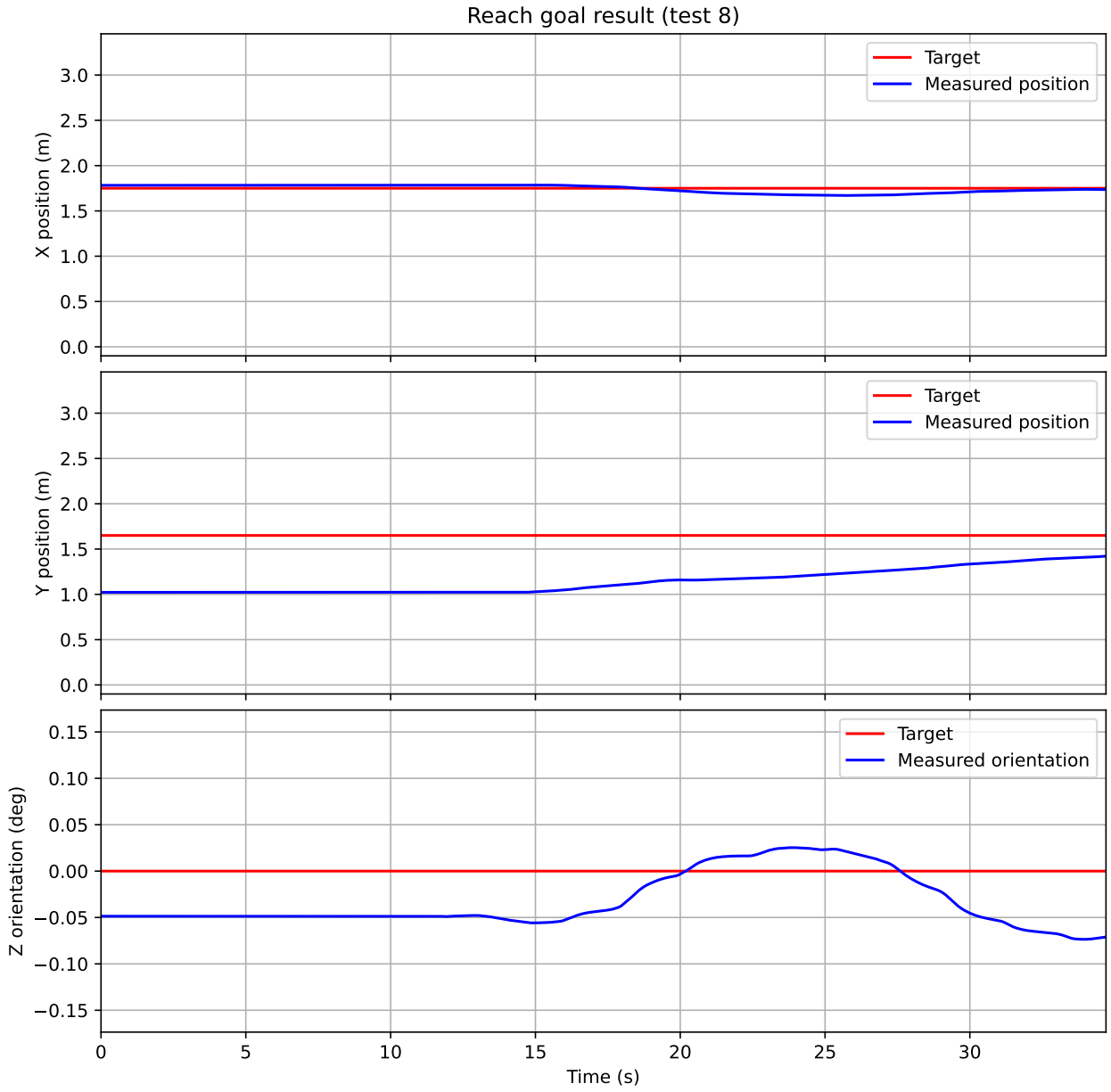


Figure C.9: Results from test 8

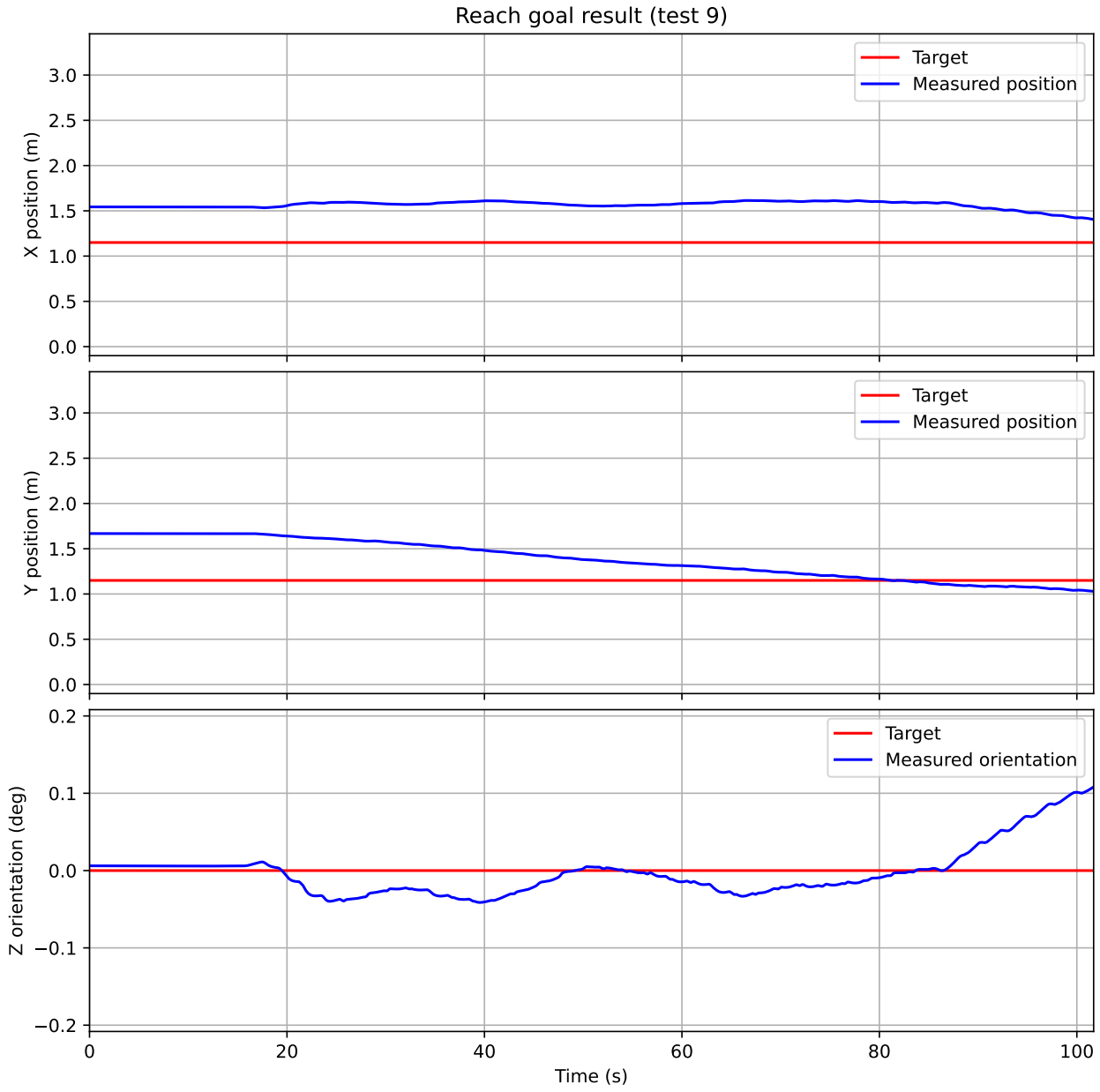


Figure C.10: Results from test 9

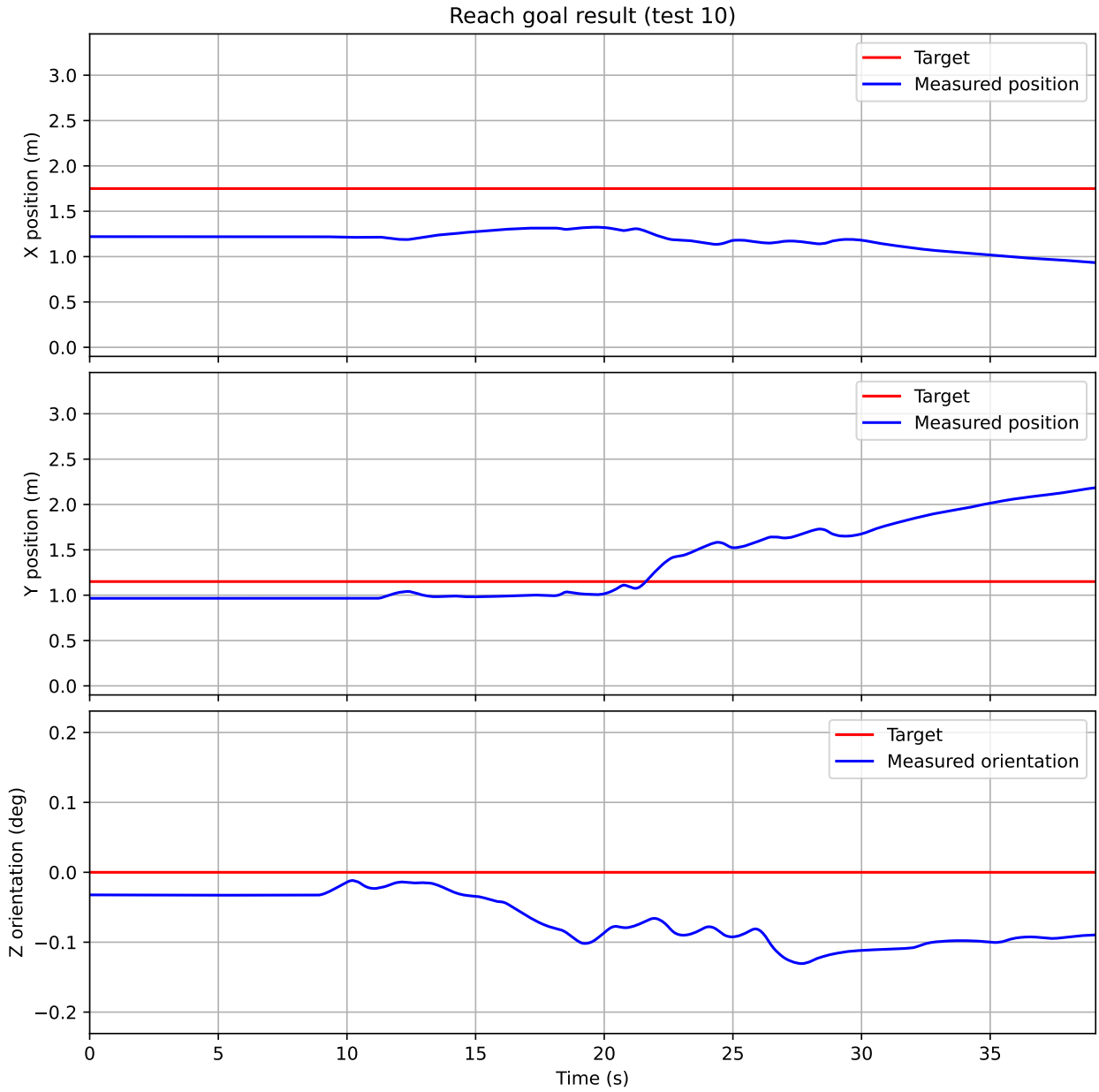


Figure C.11: Results from test 10

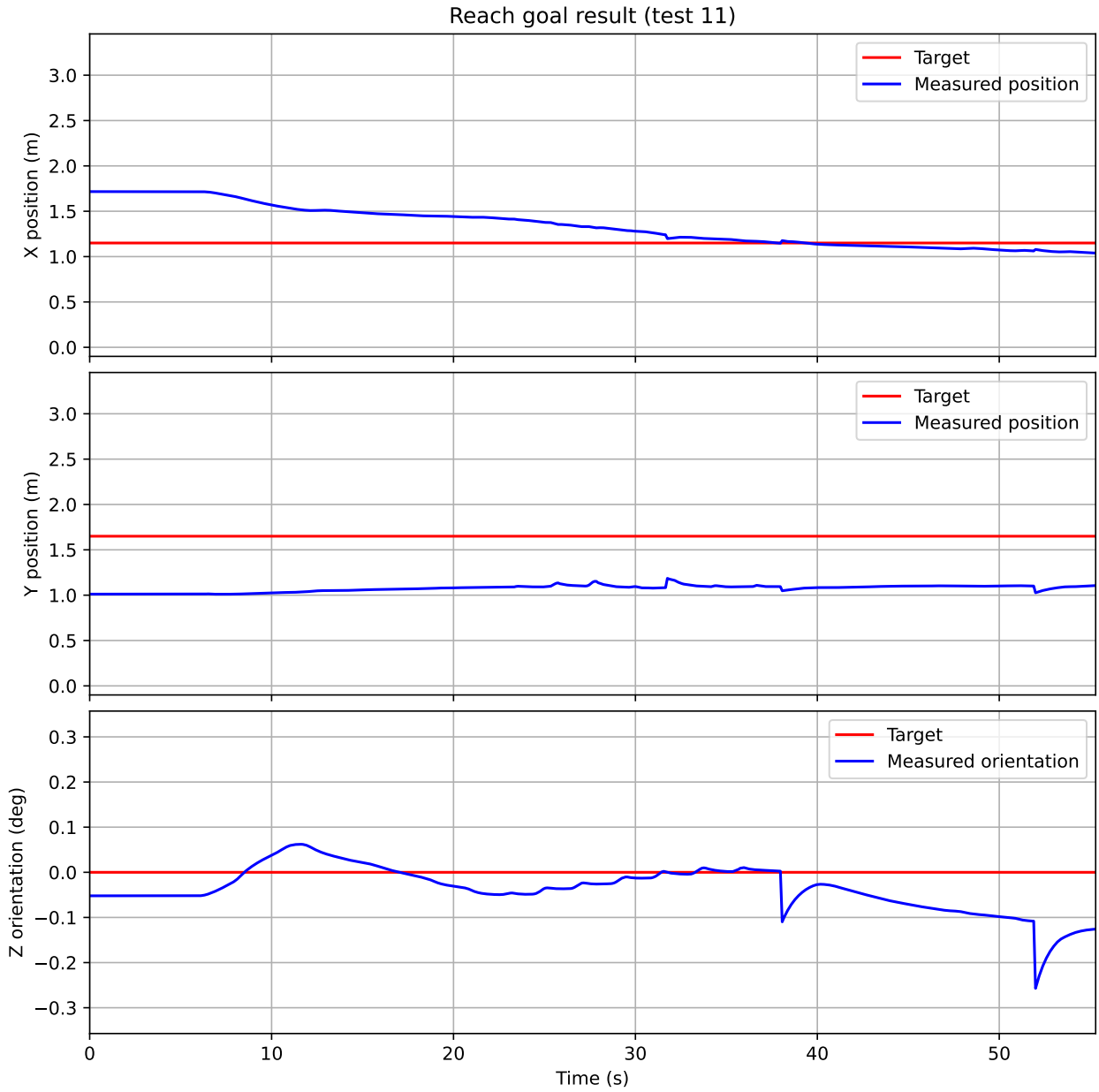


Figure C.12: Results from test 11

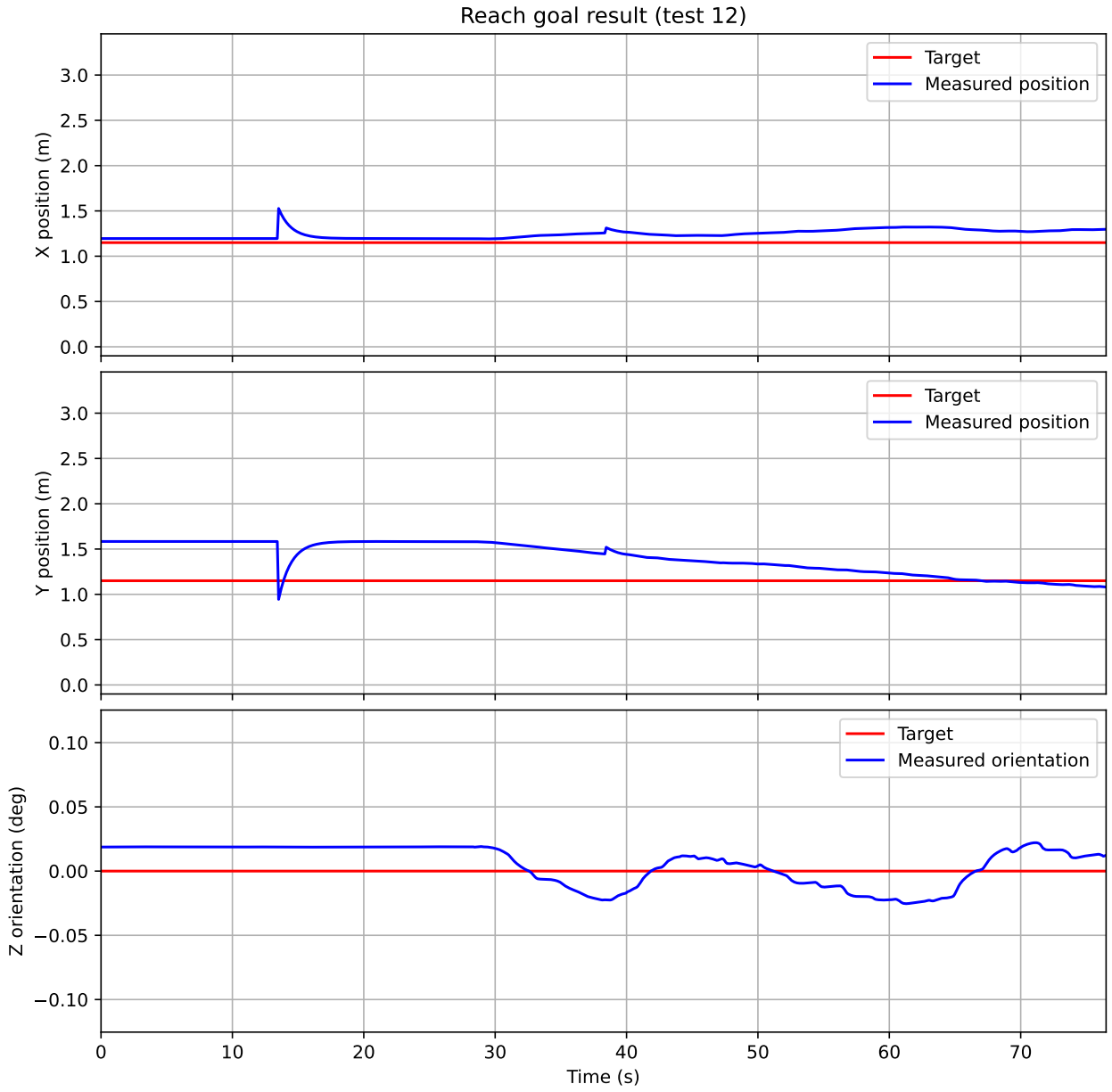


Figure C.13: Results from test 12

C.3 Follow trajectory

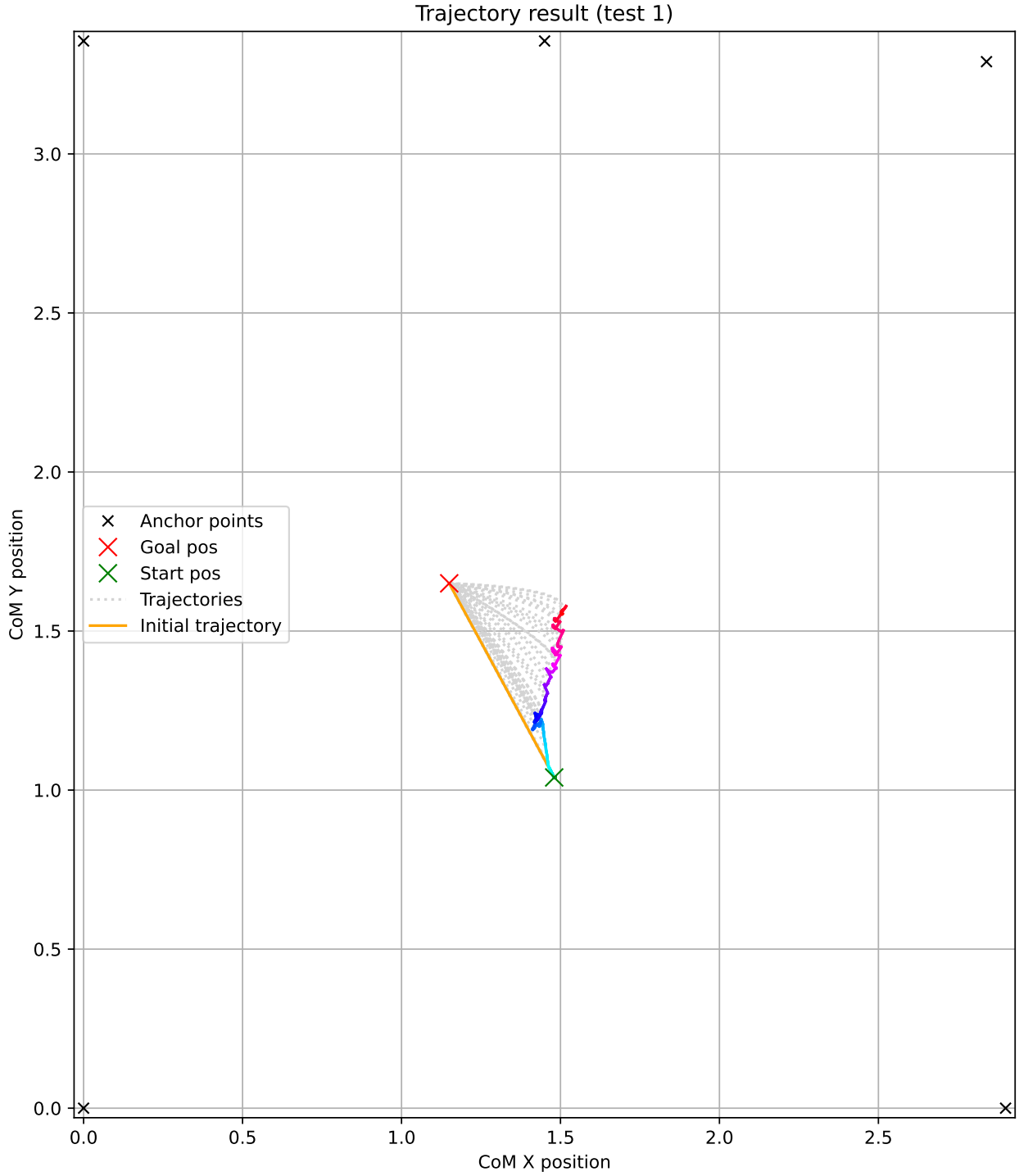


Figure C.14: Results from test 1

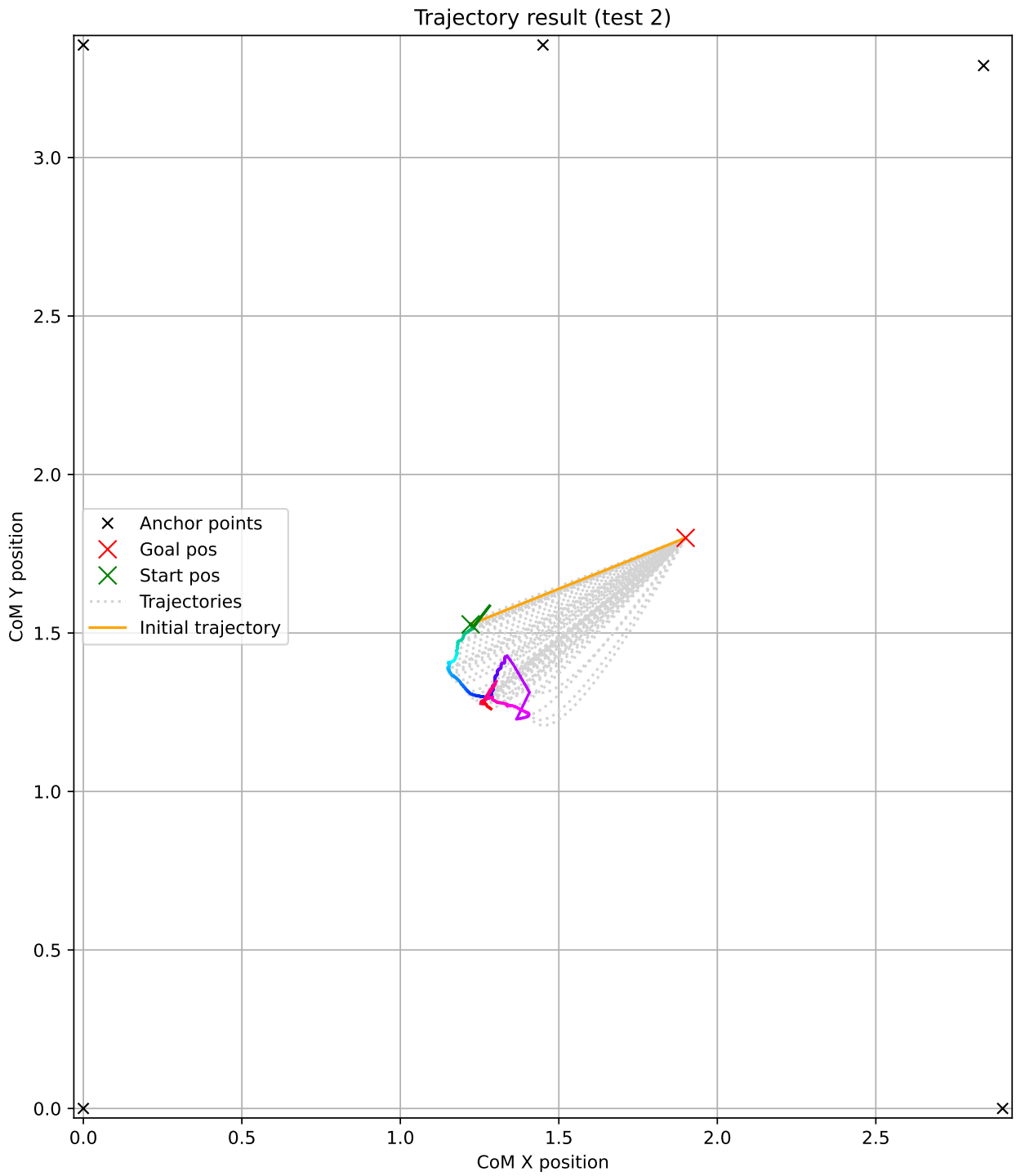


Figure C.15: Results from test 2

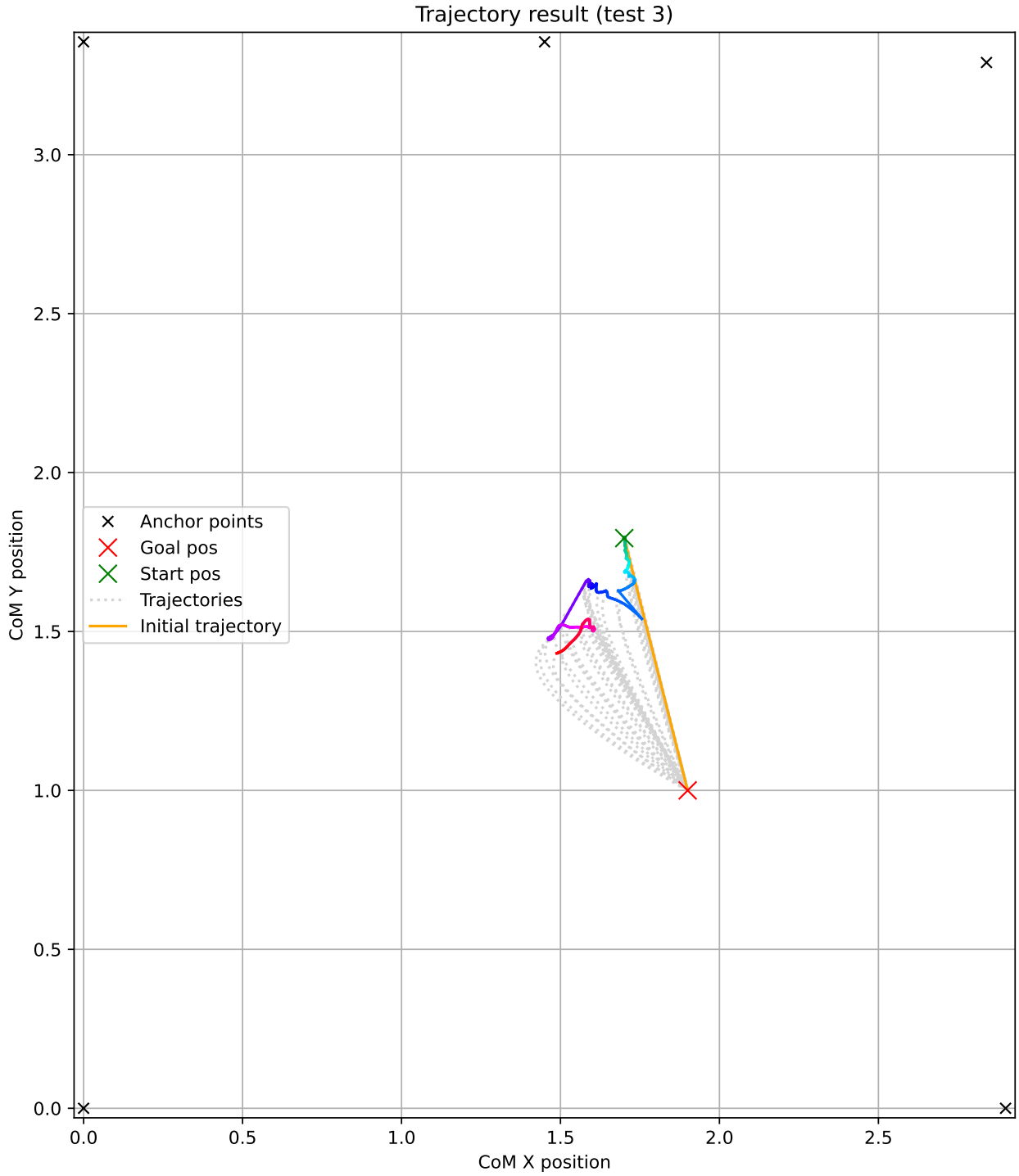


Figure C.16: Results from test 3

Figure C.17: Results from test 4

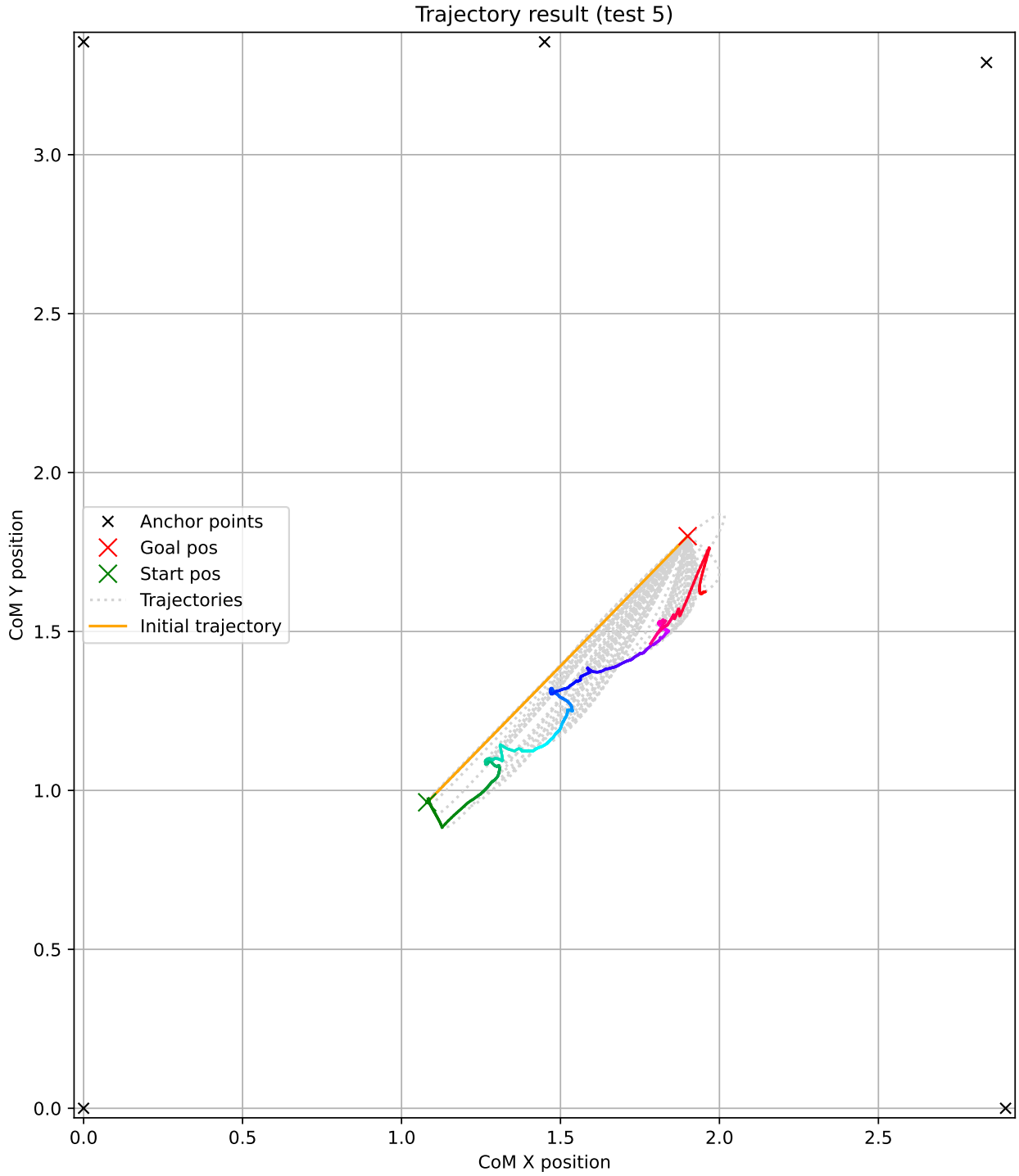


Figure C.18: Results from test 5

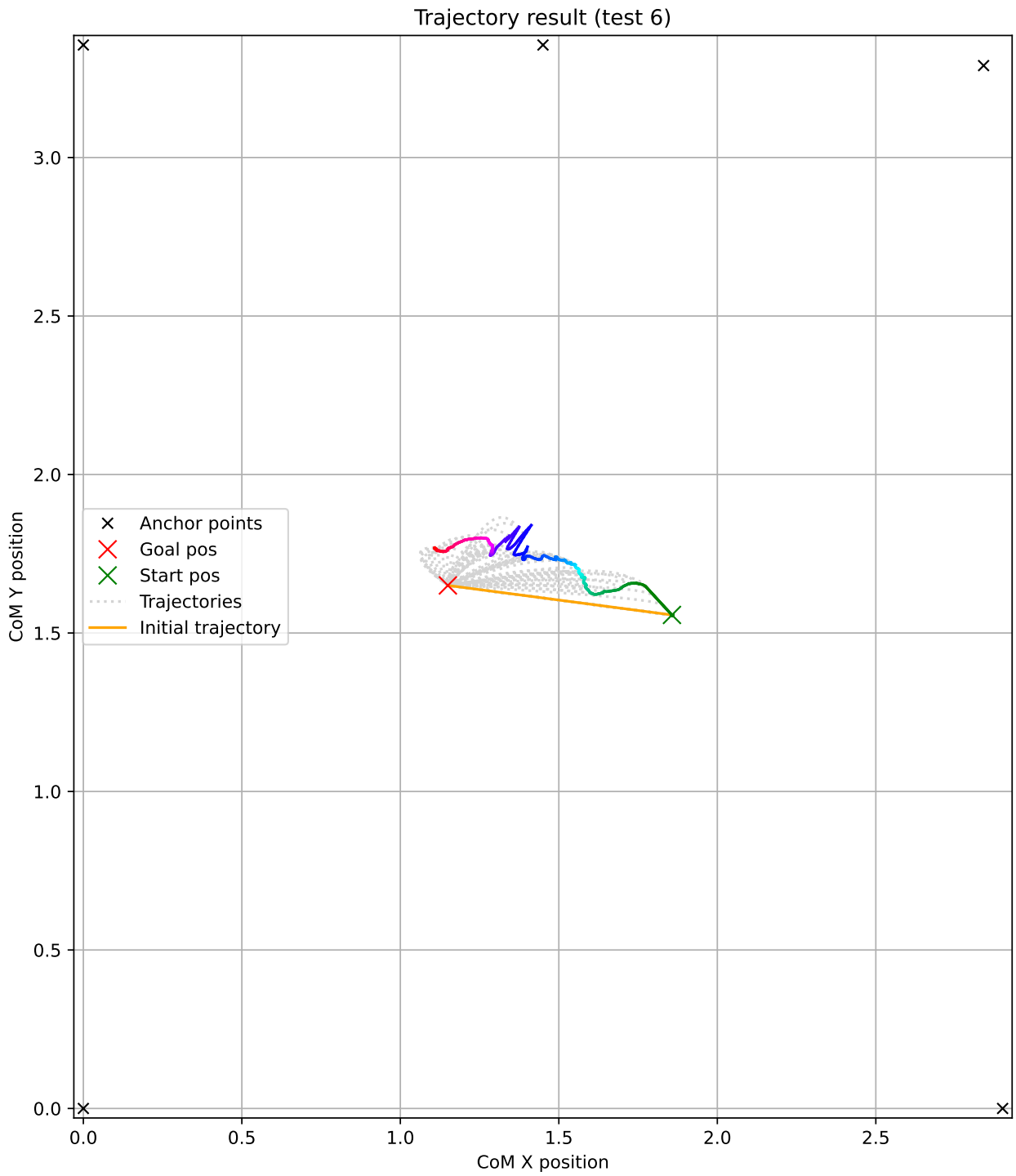


Figure C.19: Results from test 6

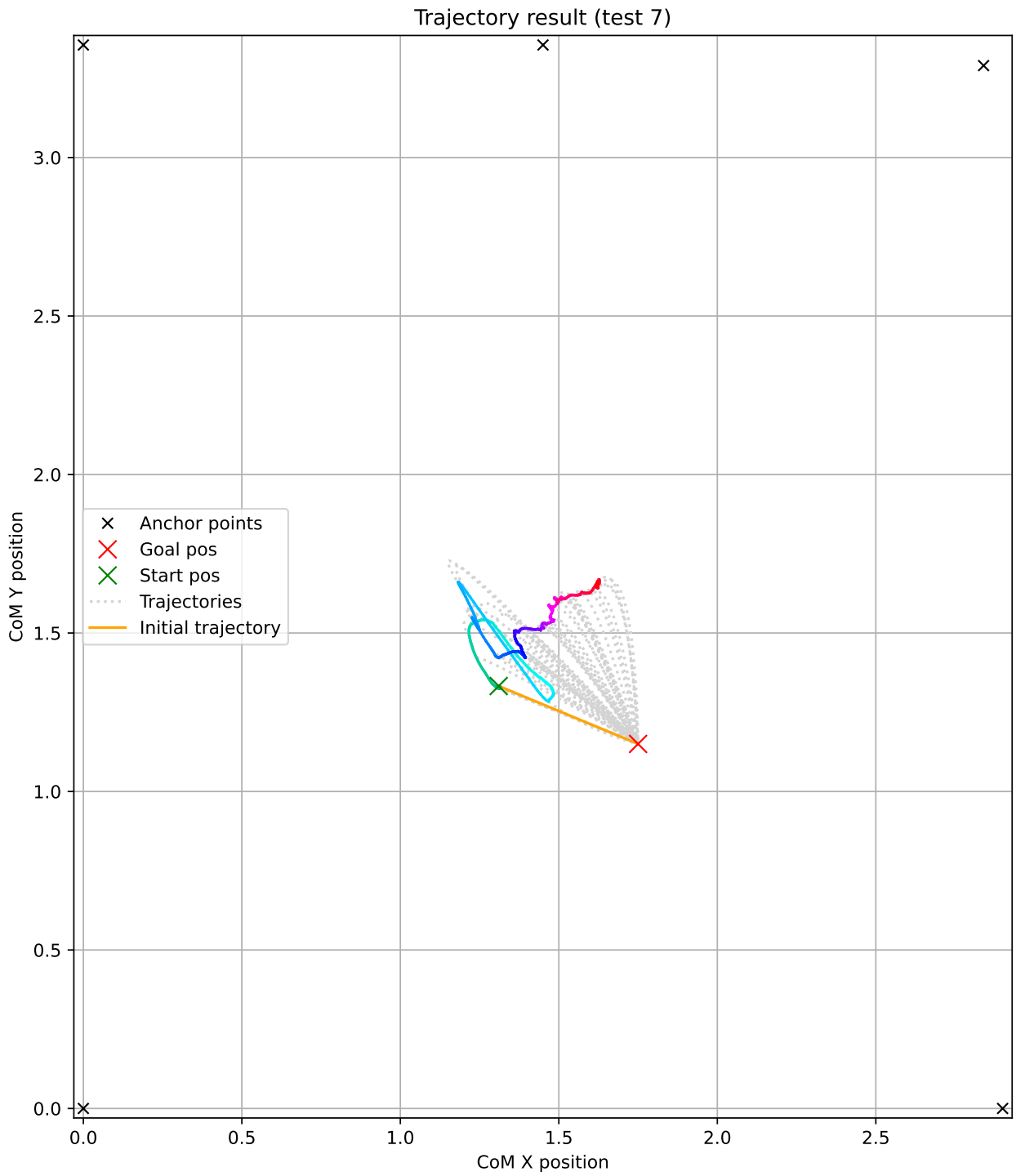


Figure C.20: Results from test 7

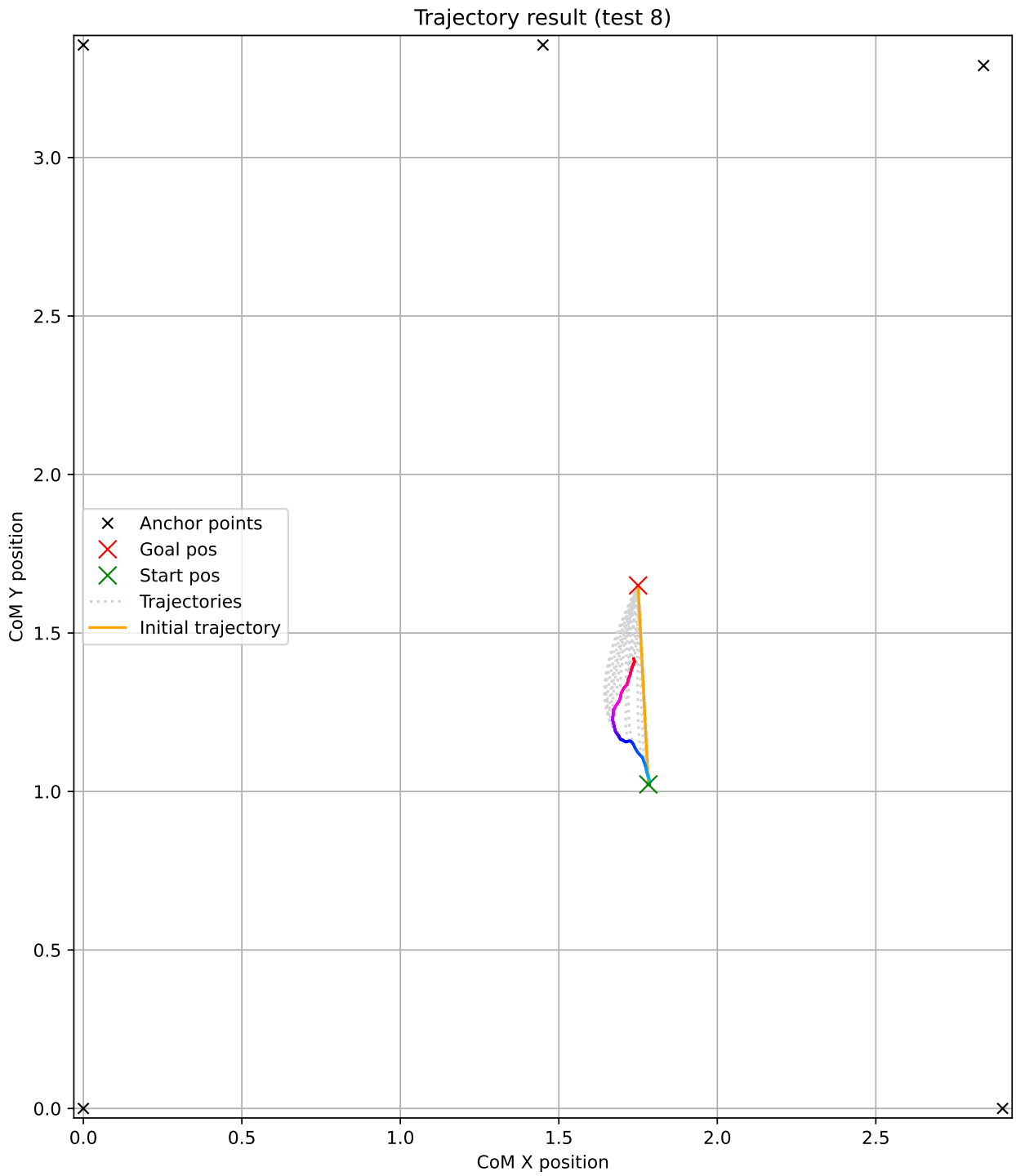


Figure C.21: Results from test 8

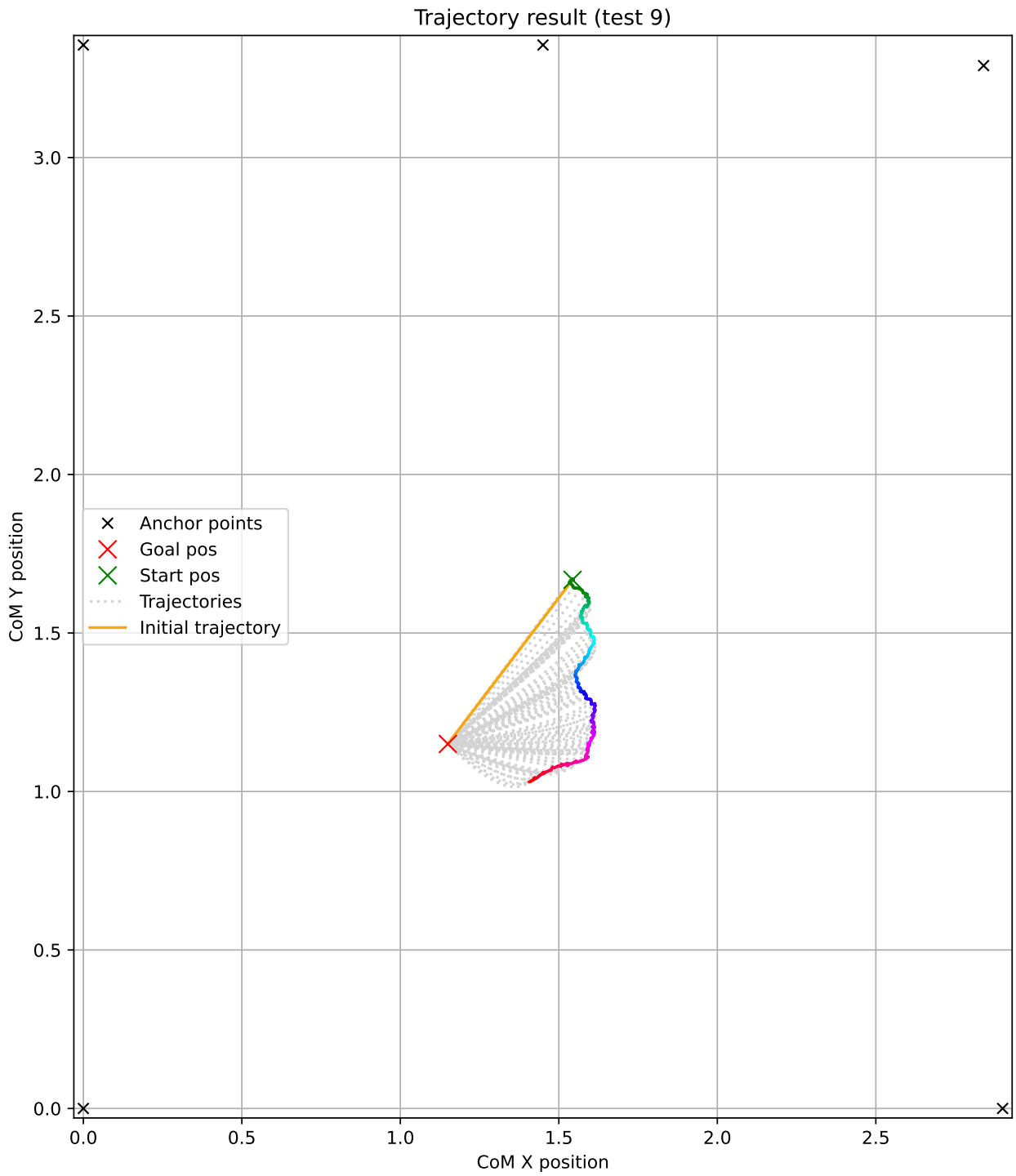


Figure C.22: Results from test 9

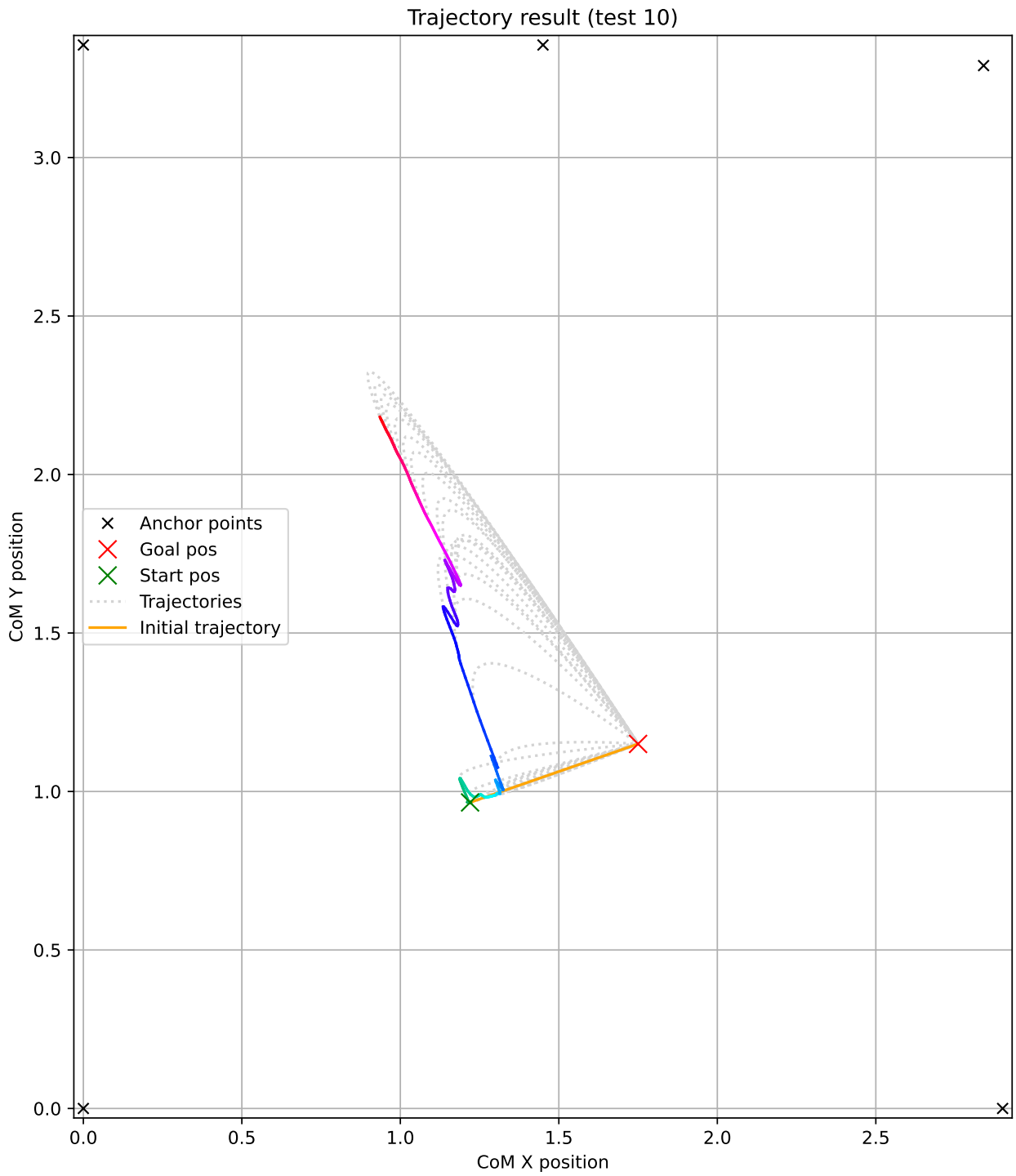


Figure C.23: Results from test 10

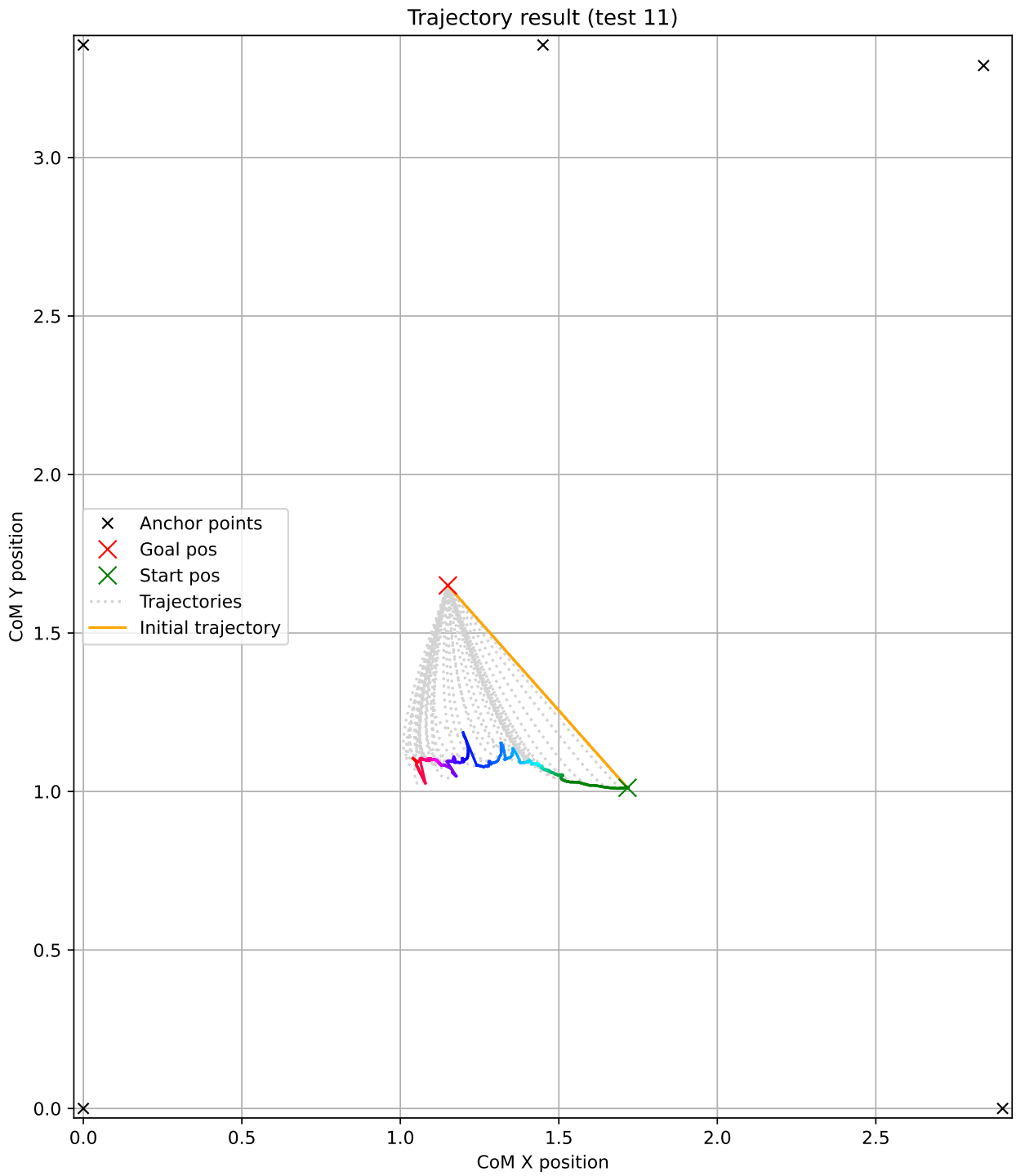


Figure C.24: Results from test 11

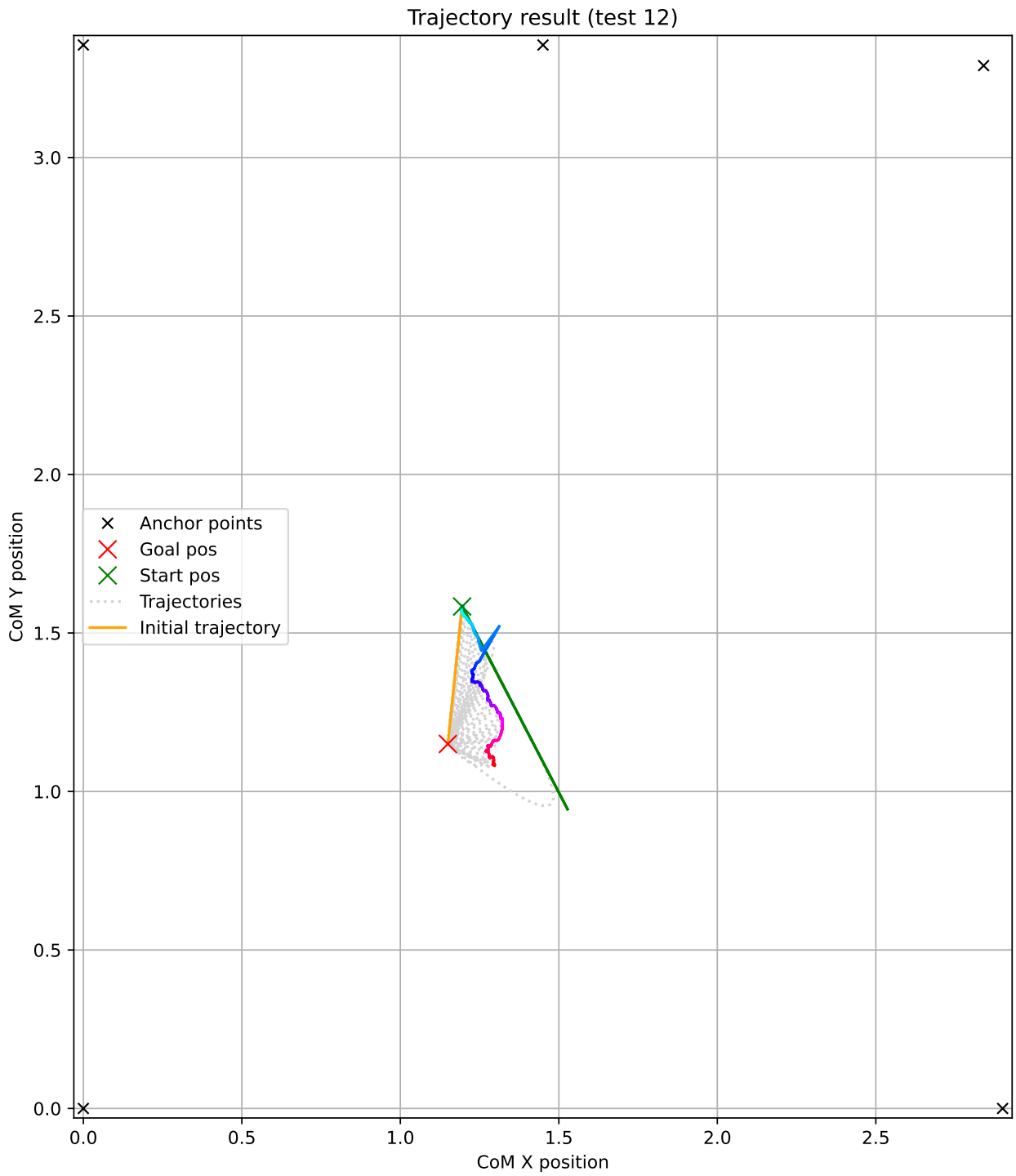


Figure C.25: Results from test 12

D Robotic frame information

Mass = 59494.65 grams
Volume = 13383163.94 cubic millimeters
Surface area = 3745043.49 square millimeters
Center of mass: (millimeters)
X = 415.05
Y = 284.53
Z = 86.46

Principal axes of inertia and principal moments of inertia: (grams * square millimeters)
Taken at the center of mass.
Ix = (1.00, -0.06, 0.01) Px = 1773101968.80
Iy = (0.06, 1.00, 0.01) Py = 4596737142.95
Iz = (-0.01, -0.01, 1.00) Pz = 6167035217.45

Moments of inertia: (grams * square millimeters)
Taken at the center of mass and aligned with the output coordinate system. (Using positive tensor notation.)
Lxx = 1784504701.79 Lxy = -177425024.42 Lxz = 31248750.71
Lyx = -177425024.42 Lyy = 4585694957.38 Lyz = 13459571.48
Lzx = 31248750.71 Lzy = 13459571.48 Lzz = 6166674670.04

Moments of inertia: (grams * square millimeters)
Taken at the output coordinate system. (Using positive tensor notation.)
Ixx = 7045743260.06 Ixy = 6848581986.27 Ixz = 2166292134.25
Iyx = 6848581986.27 Iyy = 15279597808.13 Iyz = 1477077575.40
Izx = 2166292134.25 Izy = 1477077575.40 Izz = 21232295778.11

Relative position of motors from COM (millimeters)

Distances from COM to motors					
	Motor 1	Motor 2	Motor 3	Motor 4	Motor 5
X	-367.85	-0.65	419.9	339.6	-450.05
Y	250.47	250.47	168.27	-319.53	-238.73
Z	-9.96	-9.96	-9.96	-9.96	-9.96

Table D.1: Distances from COM to motor positions

Distance from plane containing anchor point to COM = 120 millimeters