Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

Mateusz Jedynak
Viktor Karl Gravdal
Trygg Meyer Johannessen

Compensation of pendulum motion in maritime crane operations

Bachelor's thesis in Electrical Engineering specializing in
Automation and Robotics
Supervisor: Ottar L. Osen
Co-supervisor: Aleksander L. Skrede
May 2022

NTNU
Kunnskap for en bedre verden

Mateusz Jedynak
Viktor Karl Gravdal
Trygg Meyer Johannessen

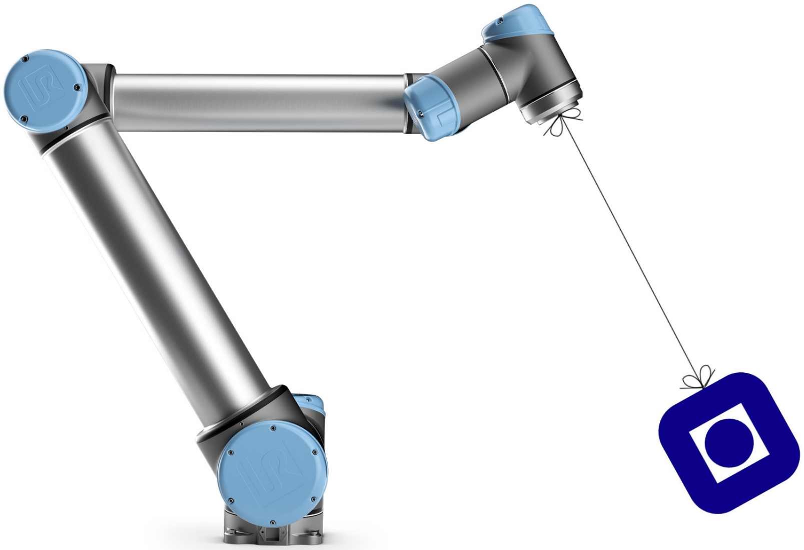
Compensation of pendulum motion in maritime crane operations

Bachelor's thesis in Electrical Engineering specializing in Automation and Robotics
Supervisor: Ottar L. Osen
Co-supervisor: Aleksander L. Skrede
May 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU
Norwegian University of
Science and Technology

Compensation of pendulum motion in maritime crane operations



Participants:
Mateusz Jedynak
Viktor Karl Gravdal
Trygg Meyer Johannessen

May 2022

PROJECT / BACHELOR THESIS
Department of ICT and Natural Sciences
Norwegian University of Science and Technology

Supervisor 1: Ottar L. Osen. Professor
Supervisor 2: Aleksander L. Skrede. PhD candidate
Supervisor 3: Agus Hasan. Professor

Preface

This project rapport finalizes a Bachelor's degree in Electrical Engineering, specializing in Automation and robotics at the Department of ICT and Natural Sciences at the Norwegian University of Science and Technology in Ålesund, Norway. As students over the past three years, we have been lucky to be exposed to a wide range of intriguing and complex subjects. Working on this project was a considerable challenge, resulting in a better grasp of modelling, simulation, robotics, systems control, and machine vision. Furthermore, it has been fascinating to see how the aforementioned components can be combined to make a functional application.

What intrigued us with this project was the chance that it may be of actual use in solving a real-world problem. We wanted to work with and learn more about modelling, robotics, control theory, and machine vision, and this project provided us with the opportunity to do so.

To fully understand the content in this Rapport, we recommend that the reader have a basic understanding of engineering, mathematics, software, robotics, and automation, as well as an interest in these fields.

Acknowledgement

First and foremost, we want to express our gratitude to the supervisor, Ottar L Osen, for his excellent advice and direction. He has demonstrated a strong interest in the project and has contributed excellent suggestions for control design and object detection. He was present at meetings, and wherever assistance was required, he responded swiftly to inquiries throughout the project.

Secondly, we want to thank Aleksander L. Skrede for his positive encouragement and support. He has provided excellent help and criticism on kinematics and robot control.

Thirdly, we want to thank Agus Hasan for his guidance when developing a simulation environment and describing forward kinematics.

Also, a heartfelt thank you to Seaonics AS for their interest in and collaboration on the project. They have supplied us with excellent feedback, equipment, documentation and even a working area throughout the process. Arne Johan Trandal, Erik Espenakk and Stig Espeseth were especially helpful during the project.

Summary

This report concerns the development of a prototype, with the purpose of investigating compensation of payload sway on a maritime crane. The report describes how several components are implemented to produce the prototype for this bachelor's thesis.

A 6-joint robotic arm is used as a prototype for a physical crane. The payload of the prototype is known. The payload's motion and dynamics are modelled using lagrangian mechanics. Kinematic equations defining the robot's motions are obtained analytically. Machine vision is used to detect the payload position. Several detection algorithms are explored, where color detection is employed in the prototype's final version. The mounting position of a camera is investigated. First on the floor, then on the prototype's end effector. Several control loops are simulated based on the payloads dynamical model. In addition, several controls are implemented on the prototype to compensate for the swaying of the payload.

In an ideal environment, the machine vision is adequate to detect the payload. The finalized prototype is capable of compensating for the payload swaying motion. However, more research and development is needed in order to reflect more realistic scenarios for a maritime crane.

Contents

Terminology

PID Proportional integral derivative controller.

RTDE Real Time Data Exchange.

LQR Linear Quadratic Regulator.

Logarithmic decrement is a technique used to derive the damping ratio for an underdamped system with oscillation [34].

Notation

K_p Proportional term of a PID controller

K_i Integral term of a PID controller

K_d Derivative term of a PID controller

θ Angle in X-direction

ϕ Angle in Y-direction

t_0 Trajectory start time

t_1 Trajectory end time

q_0 Trajectory start position

q_1 Trajectory end position

t_f Trajectory flex point

v_0 Trajectory start velocity

v_1 Trajectory end velocity

$SO(3)$ Special orthogonal group

$SE(3)$ Special Euclidean group

θ_1 Joint 1 angle, base

θ_2 Joint 2 angle, shoulder

θ_3 Joint 3 angle, elbow

L_1 Length of first link

L_1 Length of second link

L_2 Length of third link

e End effector

Abbreviations

DOF Degrees of Freedom, number of configurations for a object.

CAD Computer-aided design.

CAM Computer-aided manufacturing.

CAE Computer-aided engineering.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Maritime crane operation is a rigorous task requiring the crane operator's undivided attention. The operator must correct for load sway and identify the distance to the loading deck or the vessel deck during a typical crane operation. The operator is simultaneously putting the crane in the appropriate location throughout the process. The risk of damaging goods and people can be reduced by installing an anti-sway or anti-pendulum device.

Payload sway compensation for a marine crane is a field with limited research and resources. Because it is complex to simulate and emulate the maritime crane's environment and dynamics, among other factors. Additionally, undertaking research on an actual marine crane is expensive. Consequently, the purpose of this project is to create a prototype to test the effectiveness of various controllers in reducing payload sway. The prototype can provide valuable insight into how effective and practical implementing anti-sway to an actual crane is. Furthermore, contributing to development and research into future maritime crane technology.

1.2 Problem Formulation

The fundamental problem is anti-sway stabilization in a crane with a known payload. The goal is to create a prototype that replicates crane behaviour to test different controllers.

Problems to be addressed

- Derive a mathematical model of a pendulum system.
- Implementing a 6DOF robotic arm as a 3DOF knuckle boom crane emulator
- Using forward and inverse kinematics to define the robotic arm's movements.
- Implement a method to detect and recognize known payloads.
- Estimate known payloads position.
- Create a controller that compensates for oscillations while achieving the target position.
- Present, simulate and discuss results using Simulink/Matlab, Webots and Python
- Present future potentials for the controller and the system overall.

Inital concept of the prototype

Figure 1.1 depicts the initial prototype concept provided in the preliminary report. During the drafting of the preliminary report, a drawing of the prototype was made.

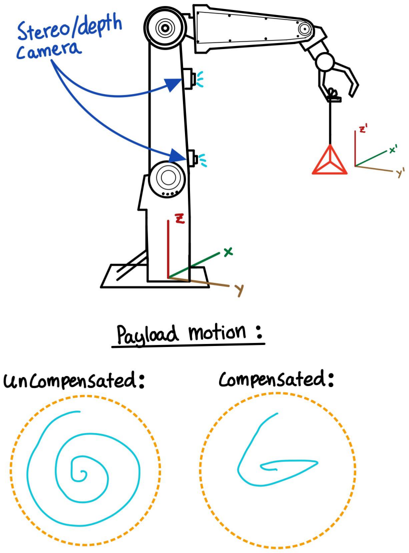


Figure 1.1: Preliminary sketch of the prototype

1.3 Limitations

- The prototype built for this project does not account for wave motions.
- The anti sway compensation only handles payload compensation in the XY-plane. It is assumed, that's cranes already has heave compensation.
- The payload detection is done within an ideal environment where the payloads appearance is known.

1.4 Structure of the Report

The report is structured as follows.

Chapter 1 - Introductions: Presents the background and objectives for this project.

Chapter 2 - Theoretical basis: Gives an introduction to the theoretical background of methods employed in the project.

Chapter 3 - Method: Contains a description of the methodology and materials that were considered throughout the project.

Chapter 4 - Result: Contains a description of the results.

Chapter 5 - Discussion: A summary and discussion of the results and findings in the project.

Chapter 6 - Conclusions: Present an overall conclusion to the project along with suggestions for further work.

Chapter 2

Theoretical basis

This chapter describes the theoretical foundation and context for supporting the further execution and content of the report. The chapter describes Lagrangian mechanics, which is utilized to create the system's model. The analytical form of kinematics is used to describe the Cartesian location of the end effector with specified joint angles, and vice versa. Theory describing trajectory planning for smooth movement of the end effector. Additionally, change of basis is mentioned, which contains extrinsic and intrinsic matrices for machine vision implementations. Furthermore, control theory used in the project is described such as, PID and LQR.

2.1 Previous Work

The authors of the paper *Modeling, Simulation and Control for Marine Crane Operations* [23]. Constructs two crane models, one in Simulink and the other in Simscape, both based on crane dynamics. To regulate the end-effector position in the horizontal direction, many control systems are developed. End-effectors are controlled in the desired direction using PID, PI, and PD controllers. The results were disappointing because the difference between the desired and measured end effector positions was too significant[23].

The thesis *Crane Payload Stabilization using Lagrangian Kinematics and Euler Angles* [27]. Examines a controller's ability to keep a payload stable while travelling along a predetermined path. It describes the motion of a spherical pendulum with a moving attachment point using Eulerangles and Lagrangian kinematics. The pendulum payload is dampened through feedback control. To ensure pendulum stability and attachment point movement, the damping is extended into a position loop and a velocity loop. Monitoring attachment point acceleration is avoided by using a second velocity loop. A 3 DOF knuckle boom crane end-effector serves as the attachment point. Crane kinematics are defined using rotation and transformation matrices, the Denavit-Hartenberg convention, and the Jacobian. For many instances, the controller's results are strong, providing a good foundation for practical implementation [27].

The design of a three-degrees-of-freedom robotic arm is the subject of Paper *Design of a Three Degrees of Freedom Robotic Arm* [28]. The authors includes equations that explain kinematic motions and the workspace of a robot.

The paper *Small-scale Robot Arm Design with Pick and Place Mission Based on Inverse Kinematics* [18], is about the design of a robotic arm with three degrees of freedom. The author employs geometry-based alternative kinematic equations.

2.2 Lagrangian Mechanics

Lagrangian mechanics is a formulation of classical mechanics that is based on the principle of stationary action and in which energies are used to describe motion[38].
It is an alternative to Newton's second law for describing dynamics[38]. The Lagrangian equation 2.1 can be thought of in a way as follows: "kinetic and potential energy of an object is all you need to know to fully predict where it will move next" [33]. The Lagrangian can be thought of as a state of motion, at any particular point in time. The Lagrangian is described by the kinetic and potential energies [33]. The Lagrangian is defined as the difference between kinetic energy [T] and potential energy [U] [38], and shown in equation 2.1.

$$\mathscr{L} = T - U \tag{2.1}$$

The Euler-Lagrange equation 2.2, which is the condition for the action to be stationary, is used to solve the equations of motion[38] for describing the dynamics.

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathscr{L}}{\partial q_i} = 0 \tag{2.2}$$

2.3 Kinematics

Kinematics describe the motion and relation of points and bodies. Kinematic equations are utilized to define the relationship between the various joints of a robot manipulator and the tool's or end-effector's orientation. The primary goal of kinematics is to describe the motion of the robot manipulator without taking torque and forces into account. The challenge of determining kinematics may be separated into two parts: forward and inverse kinematics[44]. Figure 2.1 illustrates the two parts.

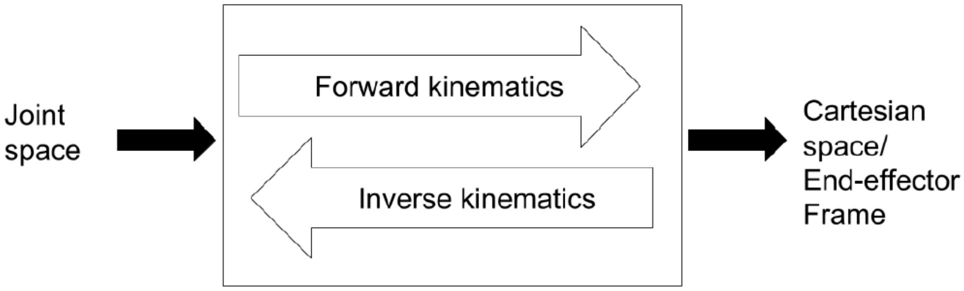


Figure 2.1: Illustration of forward and inverse kinematics[23]

Forward Kinematics is a means to determine the movement and motion of an end-effector from given positions and angles of the robot. Meaning from any given joint angle it is possible to calculate the resulting position of the end-effector[23].

Inverse Kinematics is the opposite of forward kinematics. It uses kinematic equations to calculate and describe the movement and motion of the joints to the end-effector. Using the position of the end-effector it is possible to calculate all the required joint angles[23].

2.4 Denavit Hartenberg Convention

Standard D-H Convention

The Denavit-Hartenberg, or D-H, convention is a widely used standard for choosing frames of reference in robotic applications[26]. This approach greatly simplifies the analysis and offers a systematic process for developing robotic manipulator kinematics[44]. The D-H convention describes Each homogeneous transformation A_i and represents them as a product of four fundamental transformations[26]:

$$A_i = R_{z,\theta_i} \cdot Trans_{z,di} \cdot Trans_{x,ai} \cdot R_{x,\alpha i} \tag{2.3}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta i} & 0 & 0 \\ s_{\theta i} & c_{\theta i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha i} & -s_{\alpha i} & 0 \\ 0 & s_{\alpha i} & c_{\alpha i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_{\theta i} & -s_{\theta i} c_{\alpha i} & s_{\theta i} s_{\alpha i} & a_i c_{\theta i} \\ s_{\theta i} & c_{\theta i} c_{\alpha i} & -c_{\theta i} s_{\alpha i} & a_i s_{\theta i} \\ 0 & s_{\alpha i} & c_{\alpha i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

θ_i, α_i,d_i and a_i are correspondent with link i and joint i [26]. The symbols s and c are short for $sine$ and $cosine$ respectively.

Modified DH parameters

Another method of developing robotic manipulator kinematics is using modified D-H parameters. The difference is that the positions of the coordinates system connection to the links and the sequence of the executed transformations change between the traditional D-H parameters and the modified D-H parameters[13]. The D-H convention then describes each homogeneous transformation T_i^{i-1} and represents them as a product of four fundamental transformations[13]:

$$T_i^{i-1} = R_{x_{i-1}}(\alpha_{i-1}) \cdot Trans_{x_{i-1}}(a_{i-1}) \cdot R_{z_i}(\theta_i) \cdot Trans_{z_i}(d_i) \tag{2.4}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha i-1} & -s_{\alpha i-1} & 0 \\ 0 & s_{\alpha i-1} & c_{\alpha i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & a_{i-1} \\ s_{\theta_i} c_{\alpha_{i-1}} & c_{\theta_i} c_{\alpha_{i-1}} & -s_{\alpha_i} & -s_{\alpha_{i-1}} d_i \\ s_{\theta_i} s_{\alpha_{i-1}} & c_{\theta_i} s_{\alpha_{i-1}} & c_{\alpha_{i-1}} & c_{\alpha_{i-1}} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

2.5 Open Platform Communications

Open Platform Communications (OPC) is an interoperability standard for transferring data securely and reliably[16]. The OPC Classic requirements are based on Microsoft Windows technology, and data is exchanged between software components utilizing the Distributed Component Object Model (COM/DCOM) [2].

OPC Unified Architecture (UA) has the same capabilities of the OPC Classic specifications, but with more capability. OPC UA is platform independent and hierarchically represents data. When values change based on a client's criteria, data/information is monitored and reported via exceptions [3].

2.6 Trajectory planning

The main goal of a trajectory planning is to find relationship between two element that belong to different domains, time and space. A trajectory is often defined as a parametric function of time that provides the desired position at each instant. Obviously, after defining a function, various features of its implementation, such as time discretization and load vibrations, must be considered [22].

Trajectory with asymmetric constant acceleration

The Trajectory with constant asymmetric acceleration consists of the two polynomials depicted in Figure 2.2 from [22]. This trajectory is characterized by two segments, constant acceleration and deacceleration. The flex point between the start and end time determines the duration of acceleration and deacceleration for the trajectory [22].

$$
\begin{aligned}
q_a(t) &= a_0 + a_1\,(t - t_0) + a_2\,(t - t_0)^2, && t_0 \le t < t_f \\
q_b(t) &= a_3 + a_4(t - t_f) + a_5(t - t_f)^2, && t_f \le t < t_1
\end{aligned}
\qquad
\begin{cases}
a_0 = q_0 \\
a_1 = \mathrm{v}_0 \\
a_2 = \dfrac{2h - \mathrm{v}_0(T + T_a) - \mathrm{v}_1 T_d}{2TT_a} \\
a_3 = \dfrac{2q_1 T_a + T_d(2q_0 + T_a(\mathrm{v}_0 - \mathrm{v}_1))}{2T} \\
a_4 = \dfrac{2h - \mathrm{v}_0 T_a - \mathrm{v}_1 T_d}{T} \\
a_5 = -\dfrac{2h - \mathrm{v}_0 T_a - \mathrm{v}_1(T + T_d)}{2TT_d}.
\end{cases}
$$

Figure 2.2: Trajectory with asymmetric constant acceleration polynomials obtained from [22]

2.7 Scaling Function

A scaling function scales input to output for a desired boundary. The function uses minimal and maximum input and output weight. Equation 2.6 shows the mathematical formulation for a scaling function.

$$
Output = \frac{(Input - in_{min}) \cdot (out_{max} - out_{min})}{in_{max} - in_{min}} + out_{min} \tag{2.6}
$$

2.8 Change of basis in machine vision

Controlling the position and orientation of various components in a system is critical in robotics. Each component has its own coordinate system in Euclidean space, represented in cartesian form along with its orientation. The transformation between such coordinates involves rigid body motion [45].

2.8.1 Extrinsic

In machine vision rigid body motion is commonly referred to as the extrinsic matrix. The extrinsic matrix describes the camera's orientation and translation from global coordinates [45], illustrated in figure 2.3.

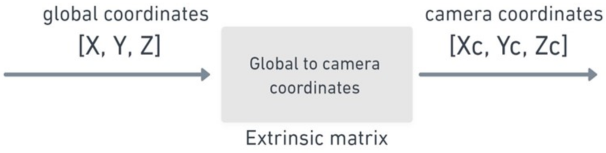


Figure 2.3: Illustration usage of extrinsic matrix

The rigid body motion provided by the extrinsic matrix belongs to the special euclidean *SE(3)* [45]. The SE(3) group is both an special orthognal group *SO(3)* rotation matrix and a translation vector. Equation 2.7 and 2.8 shows the mathematical properties of SO(3) and group SE(3)[45].

$$SO(3) \doteq \{R \in \mathbb{R}^{3\times3} \mid R^T R = I,\ \det(R) = +1\} \tag{2.7}$$

$$SE(3) \doteq \{g = (R,T) \mid R \in SO(3),\ T \in \mathbb{R}^3\} \tag{2.8}$$

To obtain the rigid body transformation $g = (R,T)$ in matrix representation for SE(3) homogenous coordinates needs to be implemented [45]. This implies the extrinsic matrix are in homogenous form.

2.8.2 Intrinsic

The intrinsic matrix in machine vision is a transformation matrix that converts points from the camera coordinate system to the pixel coordinate system [20]. Figure 2.4 shows illustration of usage of intrinsic matrix. The intrinsic matrix includes internal camera parameters. The most common are focal length, skew factor, offset from camera center and aspect ratio [45].

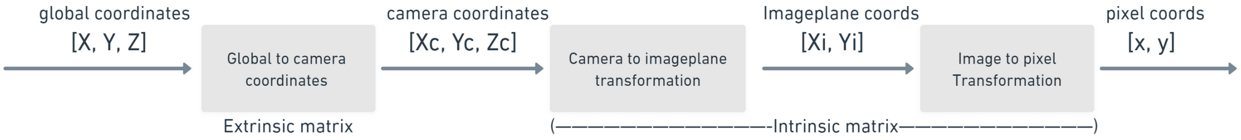

global coordinates
[X, Y, Z]
Global to camera coordinates
Extrinsic matrix
camera coordinates
[Xc, Yc, Zc]
Camera to imageplane transformation
Imageplane coords
[Xi, Yi]
Image to pixel Transformation
pixel coords
[x, y]
(———————————-Intrinsic matrix———————————)

Figure 2.4: Illustration of extrinsic and Intrinsic matrix

2.9 PID-controller

The purpose of a controller is to control the behavior of dynamical systems. The PID controller is one of the most common types of feedback controllers, because of its simplicity and effectiveness. PID stands for Proportional, Integral, and Derivative, and refers to the three terms that operate on the error signal to generate a control signal [31]. The general form for a PID-controller is shown in equation 2.9 [31].

$$u(t) = K_p \cdot e(t) + K_i \int e(t)dt + K_d \cdot \frac{d}{dt} \cdot e(t) \tag{2.9}$$

Adjusting the three parameters K_p, K_i, and K_d impacts the behavior of a closed loop system. Parameters can be tune to change the systems behaviour. In the controller, the proportional term is multiplied with the error term. The Proportional term is frequently enough to assure stability. The integral component is is added over time with the error. As a result, even little errors cause the integral component to slowly rise. Unless the error is 0, the integral response will rise over time, causing the Steady-State error to be zero [4]. The derivative response is proportional to the rate of change of the process variable [4].

2.10 Linear Quadratic Regulator

The Linear Quadratic Regulator (LQR) is a method for optimizing the feedback gains for a closed-loop systems based on the Q and R weighting [19][31]. The linear quadratic cost function in equation 2.10 [36] needs to be optimized. where both Q and R are positive defined [19]. x are the states and u are the inputs in state space form. The penalties for deviations of the state variables x and u are controlled by the weighting matrices Q and R [36]. The LQR generates a static feedback gain[19].

$$min J(t) = \frac{1}{2} \int_0^{t_f} x^T(t)Qx(t) + u^T(t)Ru(t)\,dt \tag{2.10}$$

Chapter 3

Materials and methods

This chapter presents the project organization, materials, hardware and software used in the project. Furthermore, describing the implementations of components, simulations and techniques developed during the project.

3.1 Project Organisation

The project group consists of three students. Mateusz Jedynak, Viktor Karl Gravdal and Trygg Meyer Johannessen. The Steering group consits of supervisors Ottar L. Osen, Aleksander L. Skrede and Agus Hasan.

Seaonics AS serves as the client with Arne Johan Trandal, Erik Espenakk, Arve Gudmundset and Stig Espeseth. Progressmeetings are held Biweekly with attending supervisors and clients.

3.2 Materials

3.2.1 UR10 Robot

A robotic arm with 6 degrees of freedom is used for the prototype. More specifically a UR Robotics model UR10, shown in image 3.1. The robot has 6 axes, payload handling up to 10kg and a reach of 1300mm, with a repeatability of 0.1mm[40].

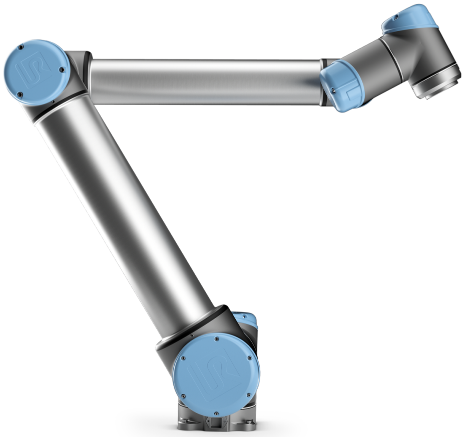


Figure 3.1: UR10 Robotic arm[40].

3.2.2 Camera Hardware

USB500W02 fish eye camera

At the initial development of machine vision a USB500W02 camera with a fish eye lens was used, shown in figure 3.2. The camera was only utilized briefly at the start of the project. Additionally, it was difficult to find decent camera documentation. After a few tests, it was discovered that the camera could not identify moving objects due to the slow transmission speed. Also, the camera resulted in noise and lack of precision due to the distortion from the fish eye lens.

Figure 3.2: USB Camera[11].

Intel RealSense Camera

To detect the payload an Intel RealSense d455 camera is used. The Intel RealSense camera has several detection capabilities. The Camera contains 3 different cameras. One RedGreenBlue (RGB) camera and a stereo-camera for depth measurement. The Intel RealSense camera is excellent for machine vision applications and 3d reconstruction. The camera supports a large amount of program libraries, allowing easier use of the cameras full functionality [6].

Figure 3.3: Intel realsense camera d455 [10].

The RGB camera has a resolution up to 1280x800, 60 Frames per second (fps) and 1 Mega Pixel (MP) sensor resolution [6]. The depth camera uses stereoscopic technology, or two cameras to achieve detection of the object in 3D. The depth camera has a frame resolution up to 1280x700. The depth cameras frame rate is 90 fps, which is 3 times faster than the RGB camera. The depth camera has a minimum working range from 52 cm and an accuracy of ±2% after 4 meters distance.

Frame rate, frame resolution, as well as sensor resolution are important factors which determine the accuracy of the object detection. For this project, involving object detection in real time, frame rate is essential for optimal results.

3.2.3 3D-designs

The figures below shows several 3D designs developed during the project. Figure 3.4 and 3.5 shows two designed end effectors which were mounted on the robot arm. A hole is made in the center of the end effector, to allow a wire or rope to connect to the payload. Figure 3.6 and 3.7 shows two designed payloads. All CAD designs were made in Fusion 360 and printed with 0.4 mm nozzle from a Prusa 3D-printer.

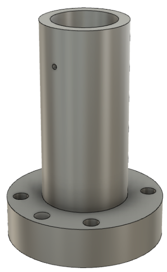


Figure 3.4: First designed end effector.

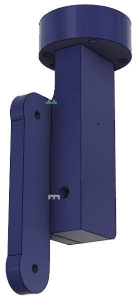


Figure 3.5: Final design end effector with camera mounting.

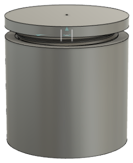


Figure 3.6: Alternative cylinder payload design.

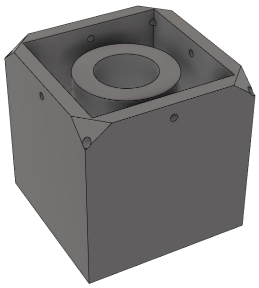


Figure 3.7: Main payload design.

3.2.4 Miscellaneous

- PC/laptop
- $2mm$ Rope.
- $2mm$ Steel wire.
- 40x40 Aluminium profiles
- Brackets
- Screws
- Port switch
- Ethernet cables
- Fasteners

3.3 Software

This section shows various software and libraries that were utilized in this project.

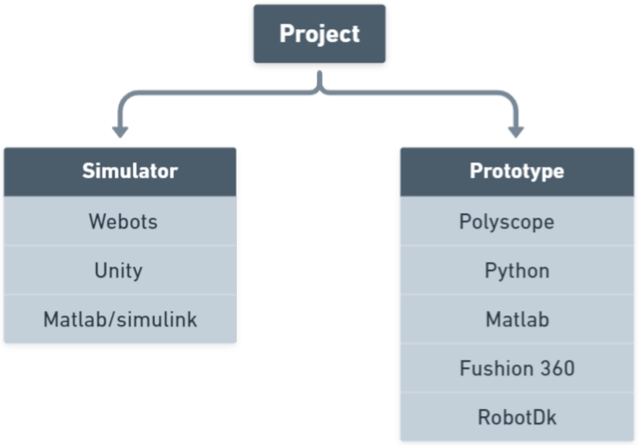


Figure 3.8: Sotware arcitecture for the bachelor project

3.3.1 GitHub Repository

It is important that everyone in a multi-member project has access to the source code. Another important factor is that work from one member does not negatively impact the output of others in the code. Therefore a GitHub repository was made. One of the group members has control over the main code and other members can download or make new branches. A link to the GitHub repository is presented below:
https://github.com/MateuszJed/Bachelor.git.

3.3.2 Programming languages

- Python
- Matlab / Simulink

3.3.3 Software

- **Fusion 360** is a CAD/CAM/CAE tool for product design and development[9].
- **RoboDK** is an industrial robot simulator and robot programming environment[39].
- **PolyScope** is a graphical user interface and software made by UR robotics. PolyScope controls the robot arm, as well as executing and writing robot programs [5].
- **Unity** is a cross-platform for game development, simulations and mobile games[42].The Unity engine can be used to make 3D and 2D interactive simulations.
- **Webots** is a multi-platform, open-source desktop application designed to simulate robots. It provides a complete development environment for simulating, programming, and modeling robots [25].

3.3.4 Libraries

Below is a list of libraries that was used during the project in Python and Matlab.

Python:	**Matlab:**
asyncio	Control system toolbox
cv2	Industrial communication tool
pyrealsense2	Simulink
glob	Symbolic Math toolbox
keyboard	
math/cmath	
matplotlib	
numpy	
os	
pandas	
rtde	
time	
scipy	
statistics	

Table 3.1: Libraries used in the project

3.4 Modeling of dynamical system

In this section, a mathematical model for a pendulum system is derived.

3.4.1 Pendulum dynamics

To describe the crane's payload dynamics, a simple pendulum model with a moving attachment point based on [27] was utilized. Additionally, other publications were reviewed for comparison and insight such as [30] and [38]. Even though the system is three-dimensional, it has been decomposed into XZ and YZ planes. This decomposition simplified the modeling and allowed simulation and testing in a single plane. In the model, the attachment point is the crane's end effector, and it can move in parallel to the XY plane. The notation for the pendulum model is described in figure 3.9 and table 3.2.

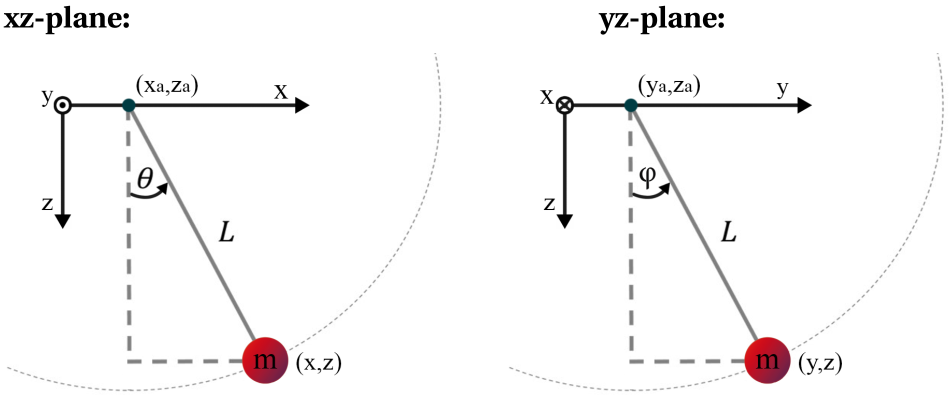


Figure 3.9: Simple pendulum in xz and yz plane with moving attachment point

Notation	
Definition	**Symbol**
Payload x-coordinate Payload y-coordinate Payload z-coordinate	x y z
Attachment point x-coordinate Attachment point y-coordinate Attachment point z-coordinate	x_a y_a z_a
Length Payload weight Angle x-direction Angle y-direction	L m θ ϕ

Table 3.2: Notation for simple pendulum model

Holonomic constraints:

$x = x_a + Lsin(\theta)$

$y = y_a + Lsin(\phi)$

$z = Lcos(\theta)$ *(xz-plane)*

Velocity-dependent constraints:

$\dot{x} = \dot{x_a} + \dot{\theta}Lcos(\theta)$

$\dot{y} = \dot{y}_a + \dot{\phi}Lcos(\phi)$

$\dot{z} = -\dot{\theta}Lsin(\theta)$ *(xz-plane)*

3.4.2 Modeling in xz-plane

The Lagrangian method was used to derive the equations of motions. Lagrangian Mechanics is described in theoretical section 2.2. The attachment point can only move in the x-direction, and is defined as $(x_a, 0, 0)$.
First derive the Lagrangian by utilizing the kinetic *(T)* and potential *(U)* energy.

$$\begin{aligned}T &= \frac{1}{2}m(\dot{x}^2+\dot{z}^2)\\ &= \frac{1}{2}m\left((\dot{x_a}+\dot{\theta}Lcos\theta)^2+(-\dot{\theta}Lsin\theta)^2\right)\\ &= \frac{1}{2}m\dot{x_a}^2+m\dot{x_a}\dot{\theta}Lcos\theta+\frac{1}{2}m\dot{\theta}^2L^2\end{aligned} \tag{3.1}$$

$$U = mgL(1-cos\theta) \tag{3.2}$$

The Lagrangian for this system can be derived from equation 3.1 and 3.2:

$$\mathscr{L} = \frac{1}{2}m\dot{\theta}^2L^2+m\dot{x_a}\dot{\theta}Lcos\theta+\frac{1}{2}m\dot{x_a}^2+mgL(cos\theta-1) \tag{3.3}$$

Solve the Euler-Lagrange formula shown in equation 2.2 to derive the motions of equations:

$$\frac{\partial\mathscr{L}}{\partial\dot{\theta}} = mL^2\dot{\theta}+m\dot{x}_aLcos\theta \tag{3.4}$$

$$\frac{d}{dt}\left(\frac{\partial\mathscr{L}}{\partial\dot{\theta}}\right) = mL\ddot{\theta}+mL\ddot{x}_acos\theta-mL\theta\dot{x}_asin\theta \tag{3.5}$$

$$\frac{\partial\mathscr{L}}{\partial\theta} = -m\dot{x}_a\dot{\theta}Lsin\theta-mgLsin\theta \tag{3.6}$$

$$\begin{aligned}\frac{d}{dt}\left(\frac{\partial\mathscr{L}}{\partial\dot{\theta}}\right)-\frac{\partial\mathscr{L}}{\partial\theta} &= mL^2\ddot{\theta}+mL\ddot{x}_acos\theta-mL\dot{\theta}\dot{x}_asin\theta+mL\dot{\theta}\dot{x}_asin\theta+mgLsin\theta=0\\ &= mL^2\ddot{\theta}+mL\ddot{x}_acos\theta+mgLsin\theta=0\end{aligned} \tag{3.7}$$

The model deviates with the sign for $mL\ddot{x}_acos\theta$ compared to [27]. However, it is only necessary to change the attachment point/end effector's sign in the control loop if needed. Equation 3.7 are the nonlinear undamped second order equation for the pendulum systems. For further development, the model needs to be linearized. The lineraization will be around small angles of "θ".

The undamped linearized model are shown in equation 3.8.

$$\begin{aligned} mL^2\ddot{\theta} + mL\dot{x}_a + mgL\theta &= 0 \\ \ddot{\theta} + \frac{g}{L}\theta + \frac{1}{L}\ddot{x}_a &= 0 \end{aligned} \tag{3.8}$$

Undamped statespace representation:

First all the state and input variables are defined for the undamped system .

State variables:	**Second-order ODE to First-order:**
$x_1 = \theta$	$\dot{x}_1 = x_2$
$x_2 = \dot{\theta}$	$\dot{x}_2 = -\frac{g}{L}x_1 - \frac{1}{L}u$
$u = \ddot{x}_a$	

The undamped state space representation:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-g}{L} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{L} \end{bmatrix} u \tag{3.9}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Parameter "L" estimation

The simple pendulum model's length "L" is not only the wire or rope's length but rather the distance between the attachment point and the payload's center of mass. The mass centroid of the payload must be estimated in order to determine the distance "L". By using the Fusion360's built-in centroid calculation. The distance between the payload's mass centroid and the top surface is 0.069m shown in the figure: 3.10.

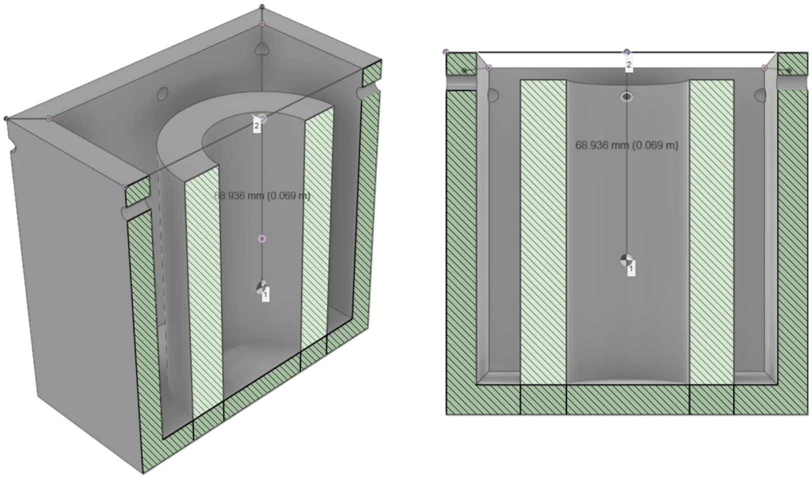


Figure 3.10: Distance between the mass centroid of the payload and the top surface

Further for an accurate length "L" estimation. The distance between the top surface of the payload and the attachment point must be measured. This is accomplished by taking measurements from the payload surface to the attachment point. The length "L" is calculated by adding the distances from the centroid to the payloads top surface and from the payload top surface to end effector. The parameter "L" is estimated to length 1.268m.

3.4.3 Damped statespace representation:

To obtain a more accurate model of the real system the model must be damped.

Damping-ratio estimation

The model obtained by the Lagrangian method is undamped. In order to derive the damped model, the damping ratio must be estimated. By measuring the behavior of the real system and analysing it trough an algorithm to estimate the damping ratio. The algorithm utilizes logarithmic decrement repetitive over the measured data. A numerical approach of *logarithmic decrement* ensures noise robustness. For each peak the algorithm calculates using 10 succeeding peaks. Iterating the same process for each peak. This is illustrated in figure 3.11. Further, the mean value of all the peak calculations is used to estimate the damping ratio.

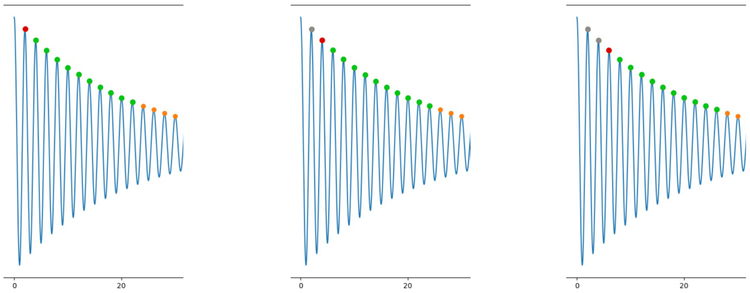
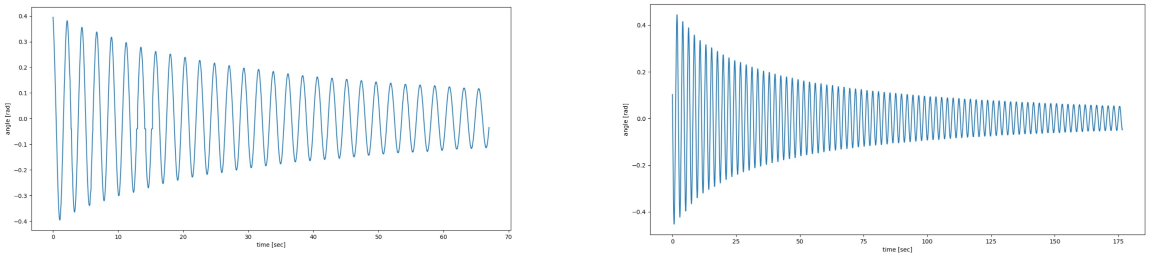


Figure 3.11: Illustration of multiple reruns of logarithmic decrement.

The damping ratio was estimated using two real-world data sets measured from the prototype. First measurement was 68 seconds long, whereas the second was 176 seconds long. The measurements taken with the prototype are sampled with 60Hz and had the camera mounted on the robots end effector. With the arm statically positioned. With $L = 1.268m$ and starting angle $\theta \approx 23°$. The two measurements are shown in figure 3.12a and 3.12b



(a) First measurement

(b) Second measurement

Figure 3.12: Two datasets used for estimating the damping ratio

Results of the estimated damping ratio "ζ",are presented in section 4.1.1.

Implementation of damping ratio

The damping ratio is implemented using an ordinary damped harmonic oscillator, differential equation 3.10, which includes the natural frequency and the damping ratio.

$$\ddot{\theta} + 2\zeta\omega_0\dot{\theta} + \omega_0^2\theta = 0 \tag{3.10}$$

The damping ratio "ζ" is estimated in the algorithm described in section 3.4.3. When comparing the damped harmonic oscillator equation 3.10 to the undamped dynamics in equation 3.8, the natural frequency "ω_0" equals $\sqrt{\frac{g}{L}}$.
The dynamics of the damped pendulum system with moving attachment point in X direction is as follows:

$$\ddot{\theta} + \zeta\sqrt{\frac{g}{L}}\dot{\theta} + \frac{g}{L}\theta + \frac{1}{L}\ddot{x}_a = 0 \tag{3.11}$$

Damped statespace representation:

First all the state and input variables are defined for the damped system .

State variables:

$$x_1 = \theta$$

$$x_2 = \dot{\theta}$$

$$u = \ddot{x}_a$$

Second-order ODE to First-order:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{g}{L}x_1 - \zeta\sqrt{\frac{g}{L}}x_2 - \frac{1}{L}u$$

The linear damped state space representation in xz-plane:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-g}{L} & -\zeta\sqrt{\frac{g}{L}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{L} \end{bmatrix} u \tag{3.12}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

3.4.4 Modeling in yz-plane:

The procedure used for deriving the model in xz-plane is the same for yz-plane. Therefore, substituting attachment point x_a with y_a and angle θ with ϕ in equation 3.11 results in the model in yz-plane shown in state space form in equation 3.13, where the states $x_1 = \phi$, $x_2 = \dot{\phi}$ and $u = \ddot{y}_a$.

State variables:	**Second-order ODE to First-order:**
$x_3 = \phi$	$\dot{x}_3 = x_4$
$x_4 = \dot{\phi}$	$\dot{x}_4 = -\frac{g}{L}x_3 - \zeta\sqrt{\frac{g}{L}}x_4 - \frac{1}{L}u$
$u = \ddot{y}_a$	

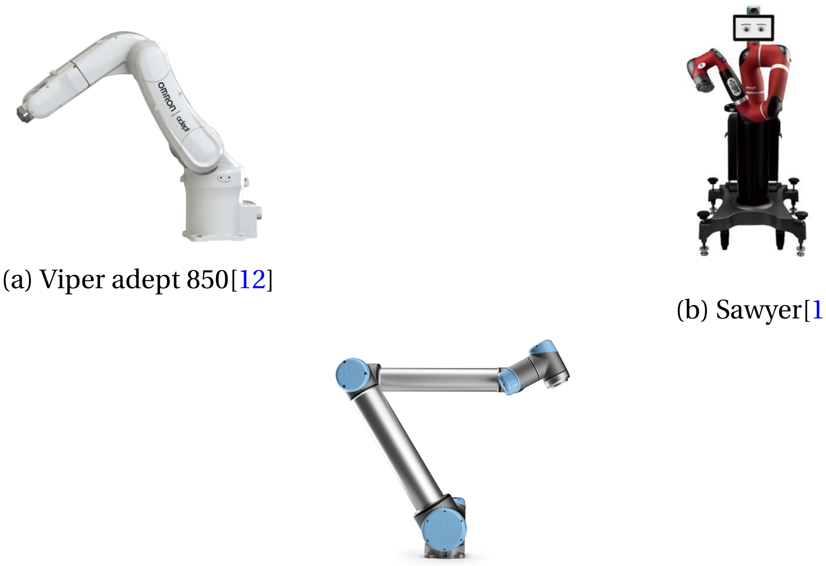
The linear damped state space representation in yz-plane:

$$\begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-g}{L} & -\zeta\sqrt{\frac{g}{L}} \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{L} \end{bmatrix} u \tag{3.13}$$

$$\begin{bmatrix} y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix}$$

3.5 Robot prototype

Working and testing on a real physical crane is unfeasible. Therefore, a smaller prototype placed in an ideal environment is used to emulate a crane with payload. At the outset of the project, the group conducted research on various suitable robotic arms. The group had access to a variety of robots, including an Omron Adept Viper 850, a Sawyer robot, and a UR10. Shown in figure 3.13



(a) Viper adept 850[12]

(b) Sawyer[17].

(c) UR10[40]

Figure 3.13: Robot arms researched in the project.

The Viper was physically placed on a moving assembly line and had a small working area. There were also other ongoing projects aiming to utilize the viper robot. Controlling the Viper is dependent on Omrons integrated software and solutions. Without experience with said software, we believed it could be more complicated because we lacked knowledge of its limitations.

The Sawyer robot is mounted on a mobile tray. However, there was little documentation and examples of the robot compared to the UR10, which is advantageous when lacking experience working with robots.
The UR10 was the most accessible of the robots. It is put on a mobile table, so it is not confined to a specific spot. In addition, there were several examples and documentation accessible. The UR10's reach is 1300mm, which is greater than those of its competitors. Ultimately, it was decided to use a UR10.

3.6 Open platform communications

The server was used to communicate between Matlab and Webots when performing simulation. An OPC server was created using the library *opcua-asyncio* to implement both an asynchronous OPC UA client and server in Python. The asynchronous server does not start a new process or thread for each request, allowing threads to run in parallel. In other words, if the server receives several requests from different clients simultaneously, all values will be updated on the server.

3.6.1 Server

A server was made by refactoring and modifying an author example code[7]. Functions *Server()* creates an asynchronous server with default values and *set endpoint()* set the IP address of the server. Furthermore, nodes and variables are established.

3.6.2 Client

For the project, two clients were created. One in Matlab and one in Webots.

The Matlab client is made utilizing Matlabs Industrial Communications Toolbox. The Function *Connect()*, including the server IP connects to the the server. The Functions *readValue()* and *WriteValue()* are used to transmit nodes to and from the server.

The Webots Client runs on Python. Using function *Client* along with the server IP to establish a connection. Functions *read_value()* and *Write_value()* are used to transmit nodes to and from the server.

3.7 Simulation

Simulating the operation of a real-world process is necessary to test solutions, troubleshooting and verifying result. For this project, three simulation platforms were explored including Unity, Webots and Matlab/Simulink.

3.7.1 Selection of simulation environment

Unity provides a better graphical engine than Webots, which is not crucial for the project. Webots however, includes API support for C, C++, Python, Java and Matlab. Whereas Unity requires programming in *C#* language. Unity would requires more experience to develop a suitable simulator.

Early in the project Unity was used for simulation. However, after a few meetings with Seaonics AS, which recommended Webots. The group adopted to use Webots. Later on, Seaonics also distributed a finished Knuckle boom crane model in Webots. After becoming familiar with Webots and having received a working crane model, the group decided to use Webots as the main simulation application for the project.

3.7.2 Webots simulator

To begin using Webots, a simple prototype of the UR10 robotic arm was created at the outset of the project. Figure 3.14 shows the first model.

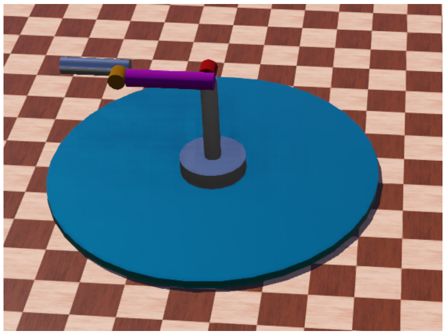


Figure 3.14: First model of UR10 in Webots

After some research, a working 3D-model of a UR10 was employed from a author on Github[35]. The model is based on a real model of the UR10, but the start angles or starting configuration for each joint were different than the real word robotic arm. Two UR10 robots are shown in figure 3.15. On the left is the original from the author from Github. On the right is after all joints have been adjusted to reflect the real world robotic arm.

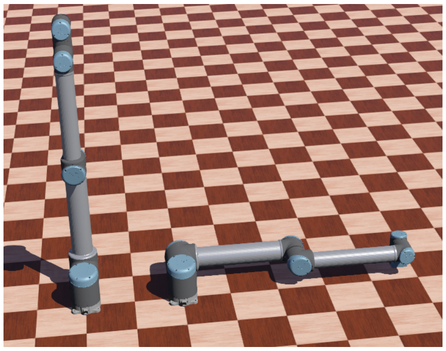


Figure 3.15: UR10 3D-models in webots. Original on the left. Adjusted start configuration on the right.

The Knuckle boom crane model provided by Seaonics was also explored and tested. The function *robot.getDevice()* is used to control a specific joint of either the crane or UR10 model. The OPC server and client mentioned in section 3.6 are used to send information to and from the simulation. The server contains one node with three variables. *angle position 1, 2, 3.* An OPC client in Webots continuously reads all angles from the server. The variables *angle position 1, 2, 3* corresponds to the actuation of joints *Base, shoulder, Elbow,* respectively.

The link below presents a simulation of the Knuckle boom crane model in Webots.
https://youtu.be/Cw0_8ZL_C_g

Figure 3.16 shows a finished simulation environment, including the UR10 prototype and a Knuckle boom crane. The right side of figure 3.16 is zoomed in to illustrate the size difference between the two models. Both models include real-size representation.

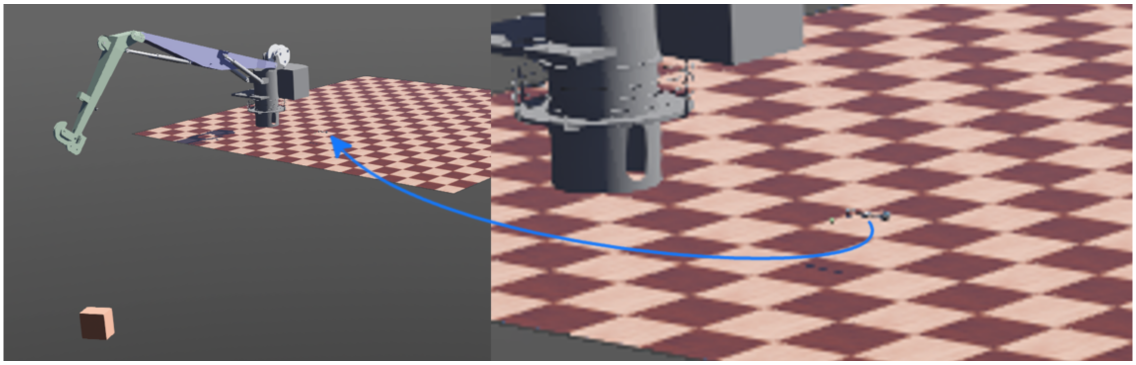


Figure 3.16: Simulation environment of Knuckle boom crane on the left and UR10 robot arm on the right.

3.8 Kinematics

Kinematic equations are needed in order to manipulate the end effector of the robotic arm. Kinematics are used to describe the position of the robot in Cartesian space using the angular position of joints and vice versa. The UR10 provides 6 axis of motion. However, only 3 are need to simulate crane motion. Therefore, the original 6DOF kinematic problem can be reduced to a 3DOF kinematic problem. The 3 remaining axis of the robot arm are set to a fixed position. Since the problem can be reduced to a less complex 3DOF problem, an analytic approach was chosen, rather than a numerical approach. An analytic approach is more precise and requires less computational cost. Illustration 3.17 shows a simplified 2D kinematic diagram of the now reduced 3DOF robotic arm. Table 3.3 shows notation of figure 3.17.

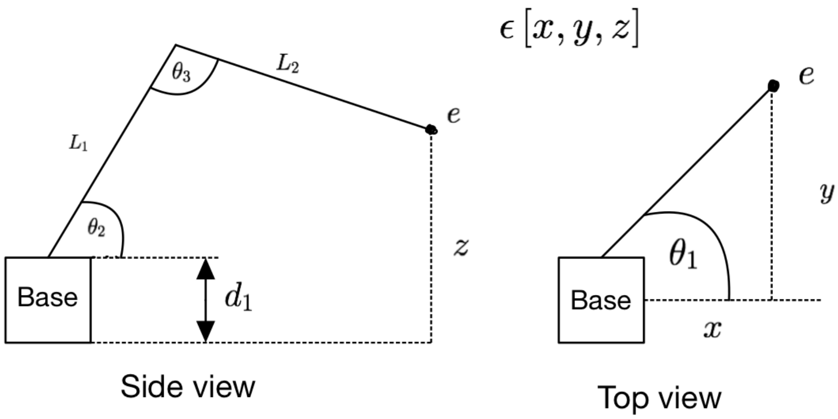


Figure 3.17: 2D Kinematic diagram.

Notation	
Definition	**Symbol**
Joint 1 angle, base	θ_1
Joint 2 angle, shoulder	θ_2
Joint 3 angle, elbow	θ_3
Length of first link	L_1
Length of second link	L_2
Length of third link	L_3
End effector	e

Table 3.3: Notation for 2D kinematic diagram

Deriving the kinematics for the 3DOF robotic arm proved difficult. Due to a lack of experience with kinematics and robotics in general. Throughout the project, the group derived two separate kinematic models. The first model proved inaccurate in some scenarios while the second model proved accurate in all scenarios. Both models are described in the sections below. The results for each model is presented in section 4.3.

3.8.1 First kinematic model

Forward Kinematics

To describe the forward kinematics five frames were assigned to the robotic arm following the modified D-H Convention described in section 2.4. Illustration 3.18 shows the assigned frames in a kinematic diagram.

The authors of paper[28] present a way of determining the kinematic equations for a general 3 DOF robotic arm. The paper was used deliberately when deriving the kinematics for the UR10. The assignment of frames are summarized in three steps[24]:

- Z_i axis is assigned pointing along the i-th joint axis.
- X_i axis is assigned pointing perpendicular from Z_i to Z_{i+1}
- y_i axis is assigned following the right hand rule.

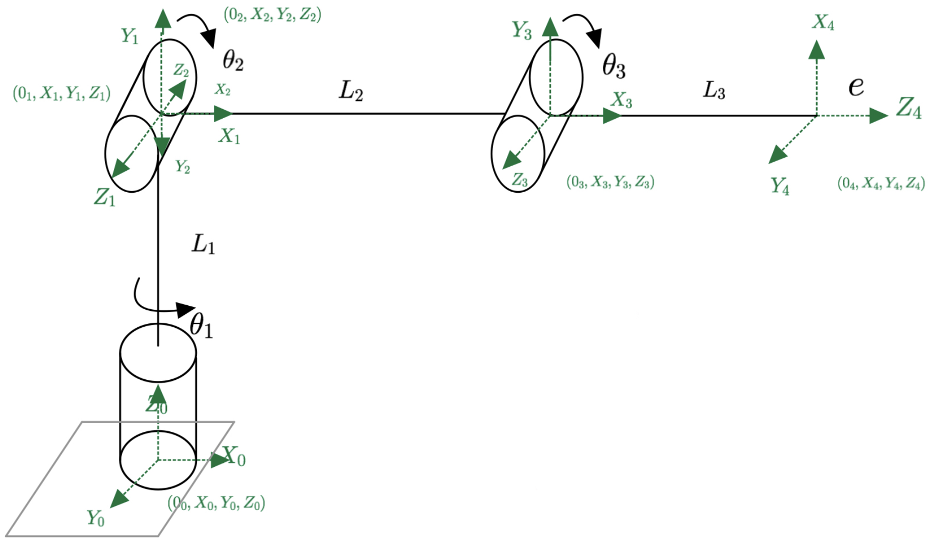


Figure 3.18: 3D Kinematic diagram of Frames 4 to 0.

Frame 1, which corresponded to Link (1), was connected at its top end. Because Z_0 and Z_1 are parallel, the origin of this frame may have been assigned to any place along Link (1). Frame 2 rotates with Link (2) and is placed at the junction of Links (1) and (2), with X_2 running along the shared perpendicular extending from Z_2 to Z_3. Frame 3 revolves with Link (3) and is joined at its contact with Link (2). Lastly, Frame 4 relates to the end-effector and is placed at its end point[28]. Based on these defined frames, Modified Danavit-Hartenberg (D-H) parameters [13] were determined as shown in table 3.4.

Modified D-H Parameters				
i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0°	0	L_1	θ_1
2	90°	0	0	θ_2
3	0°	L_2	0	θ_3
4	0°	L_3	0	0°

Table 3.4: Modified D-H Parameters

Definition of D-H parameters[28]:
α_i: Link twist or anlge about the X_i between (Z_i, Z_{i+1})
a_i: Link length or distance along X_i between (Z_i, Z_{i+1})
d_i: Link offset or distance along Z_i between (X_{i-1}, X_i)
θ_i: Joint angle or the angle about Z_i between (X_{i-1}, X_i)

The modified D-H parameters are used to compute a total homogeneous transformation matrix which describes the kinematic relation from the end effector to all joints corresponding with every frame. To compute the transformation for every coordinate frame. ($T_1^0, T_2^1, T_3^2, T_4^3$,) substitute the modified D-H parameters for the general formula for T_i^{i-1}[28]:

$$T_i^{i-1} = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & a_{i-1} \\ s_{\theta_i} c_{\alpha_{i-1}} & c_{\theta_i} c_{\alpha_{i-1}} & -s_{\alpha_i} & -s_{\alpha_{i-1}} d_i \\ s_{\theta_i} s_{\alpha_{i-1}} & c_{\theta_i} s_{\alpha_{i-1}} & c_{\alpha_{i-1}} & c_{\alpha_{i-1}} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

$$T_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_2^1 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

$$T_3^2 = \begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_4^3 = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

c_i is the cosine of the angle θ_i and s_i is the sine of the angle θ_i.

T_4^0 describing the end-effector frame with the reference/base frame may be derived by multiplying the homogeneous transformation matrices in the correct sequence[28]:

$$T_4^0 = T_1^0 T_2^1 T_1^2 T_4^3 \tag{3.17}$$

$$T_4^0 = \begin{bmatrix} c_1 c_{23} & -c_1 s_{23} & s_1 & c_1 c_{23} L_3 + c_1 c_2 L_2 \\ s_1 c_{23} & -s_1 s_{23} & -c_1 & s_1 c_{23} L_3 + s_1 c_2 L_2 \\ s_{23} & c_{23} & 0 & s_{23} L_3 + s_2 L_2 + L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

The first three elements in the final column of T_4^0 are the Cartesian coordinates of the origin of the end effector (frame 4) with respect to the reference/base (frame 0).

$$x = c_1 c_{23} L_3 + c_1 c_2 L_2 \tag{3.19}$$

$$y = s_1 c_{23} L_3 + s_1 c_2 L_2 \tag{3.20}$$

$$z = s_{23} L_3 + s_2 L_2 + L_1 \tag{3.21}$$

It is worth noting that $c23$ is the same as $cos(\theta_2 + \theta_3)$, and $s23$ is $sin(\theta_2 + \theta_3)$. Eqs. 3.19 to 3.21, calculate the location of the end effector as a function of the joint angles θ_1, θ_2, and θ_3.

Inverse Kinematics

The position coordinates equations, Eqs. 3.19 to 3.21, were utilized in inverse kinematics to derive the analytical expressions of the rotational angles (i.e. θ_1, θ_2, and θ_3) for the base, shoulder, and elbow joints as a function of the required x, y, and z coordinates. The following is a summary of the inverse kinematic equations' derivation based on[28]:

To begin, dividing Eq. 3.20 by Eq. 3.19 yields:

$$\frac{y}{x} = \frac{s_1 c_{23} L_3 + s_1 c_2 L_2}{c_1 c_{23} L_3 + c_1 c_2 L_2} = \frac{s_1 (c_{23} L_3 + c_2 L_2)}{c_1 (c_{23} L_3 + c_2 L_2)} = \frac{s_1}{c_1} = tan\theta_1 \tag{3.22}$$

As a result, θ_1 is represented directly as

$$\theta_1 = atan2\left(\frac{y}{x}\right) \tag{3.23}$$

Based on the signs of both the denominator and the numerator, the (atan2) function calculates the arctangent of the argument and the correct quadrant. Secondly, an explicit equation for the cosine of θ_3 as a result of the robotic arm configuration and the coordinates of the target point, may be obtained by adding the squares of Eqs. 3.19 to 3.21 and rearranging them.

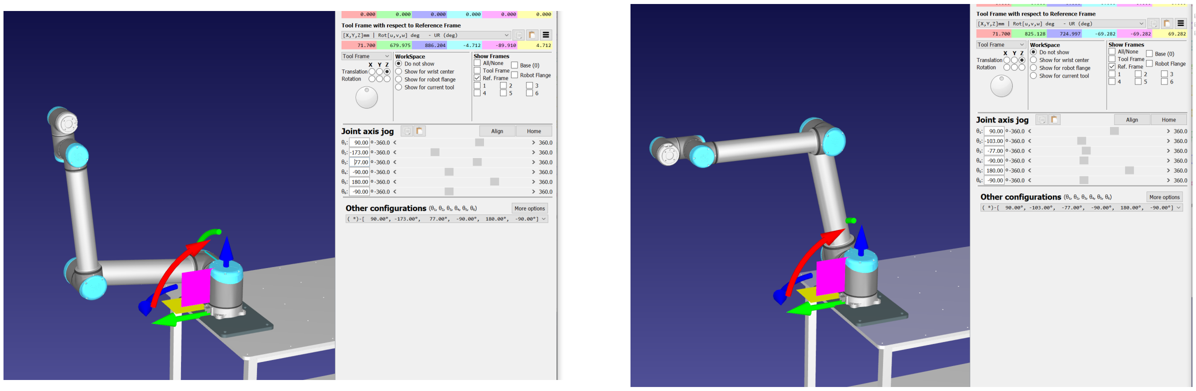
$$c_3 = \frac{x^2 + y^2 + z^2 - (L_1^2 + L_2^2 L_3^2) - 2L_1(z - L_1)}{2L_2 L_3} \tag{3.24}$$

Using the Pythagorean trigonometric identity

$$s_3 = \pm\sqrt{1 - c_3} \tag{3.25}$$

The ambiguity in the algebraic sign in the previous equation, Eq. 3.25, shows that there are several solutions. The two options relate to the two distinct configurations that are available. The arm may take two different angles to go to the same place (i.e. elbow up and elbow down)[28]. Figure 4.1 shows a simulation of two different solutions for the same coordinates in RoboDK. Notice that 3.19a is unfeasible to reflect a crane. However, 3.19b is. Another thing to note is that the two angles q_2 and q_3 can be defined within a specific range. When compared to the angles of the unfeasible solution, angle q_2 is always larger in the desired solution, than the unfeasible solution. q_3, will for all desired solutions be larger than 0. Continuing, similar to the previous eq. 3.23:

$$\theta_3 = atan2\left(\frac{s_3}{c_3}\right) \tag{3.26}$$



0.000	0.000	0.000	0.000	0.000	0.000

Tool Frame with respect to Reference Frame

[X,Y,Z]mm | Rot[u,v,w] deg - UR (deg)

71.700	679.975	886.204	-4.712	-89.910	4.712

Tool Frame

	X	Y	Z
Translation	○	○	●
Rotation	○	○	○

WorkSpace
● Do not show
○ Show for wrist center
○ Show for robot flange
○ Show for current tool

Show Frames
□ All/None □ Base (0)
□ Tool Frame □ Robot Flange
☑ Ref. Frame
□ 1 □ 2 □ 3
□ 4 □ 5 □ 6

Joint axis jog — Align — Home

θ₁:	90.00	° -360.0	360.0
θ₂:	-173.00	° -360.0	360.0
θ₃:	77.00	° -360.0	360.0
θ₄:	-90.00	° -360.0	360.0
θ₅:	180.00	° -360.0	360.0
θ₆:	-90.00	° -360.0	360.0

Other configurations (θ₁, θ₂, θ₃, θ₄, θ₅, θ₆) — More options

(*)-[90.00°, -173.00°, 77.00°, -90.00°, 180.00°, -90.00°]

(a) Undesirable solution.

Tool Frame with respect to Reference Frame

[X,Y,Z]mm | Rot[u,v,w] deg - UR (deg)

71.700	825.128	724.997	-69.282	-69.282	69.282

Tool Frame

	X	Y	Z
Translation	○	○	●
Rotation	○	○	○

WorkSpace
● Do not show
○ Show for wrist center
○ Show for robot flange
○ Show for current tool

Show Frames
□ All/None □ Base (0)
□ Tool Frame □ Robot Flange
☑ Ref. Frame
□ 1 □ 2 □ 3
□ 4 □ 5 □ 6

Joint axis jog — Align — Home

θ₁:	90.00	° -360.0	360.0
θ₂:	-103.00	° -360.0	360.0
θ₃:	-77.00	° -360.0	360.0
θ₄:	-90.00	° -360.0	360.0
θ₅:	180.00	° -360.0	360.0
θ₆:	-90.00	° -360.0	360.0

Other configurations (θ₁, θ₂, θ₃, θ₄, θ₅, θ₆) — More options

(*)-[90.00°, -103.00°, -77.00°, -90.00°, 180.00°, -90.00°]

(b) Desired solution.

Figure 3.19: Two possible solutions, for the same position simulated in RoboDK.

Finally, The second angle can be described as[28]:

$$\theta_2 = atan2\left(\frac{(z-L_1)(c_1-s_1)}{(x-y)}\right) - atan2\left(\frac{s_3 L_3}{(c_3 L_3 + L_2)}\right) \tag{3.27}$$

3.8.2 Second kinematic Model

Since the original kinematic model was incorrect, a second model was made.

Forward

The forward kinematics are defined similarly to the previous influenced by [28]. This time however, using the standard D-H convention described in section 2.4. Five frames are assigned. Additionally the world coordinates are set to match the actual base coordinates defined by the robotic arm. The assigned frames are illustrated in a kinematic diagram 3.20.

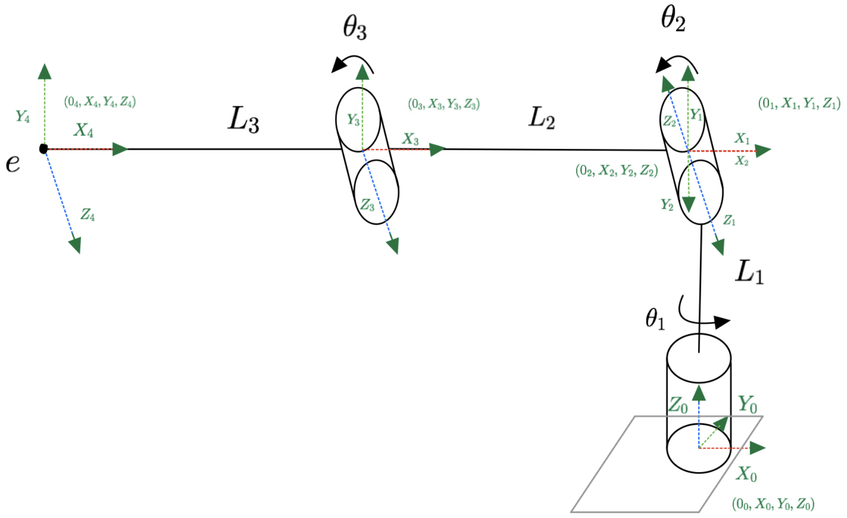


Figure 3.20: 3D Kinematic diagram of Frames 4 to 0.

The assignment of frames are again summarized in three steps[24]:

- Z_i axis is assigned pointing along the i-th joint axis.
- X_i axis is assigned perpendicular from Z_i to Z_{i+1} pointing along the robots predefined X-axis
- y_i axis is assigned following the right hand rule.

Frame 0, serving as the world coordinates is placed at the base of the arm. Frame 1 is connected at the top end of frame 0, creating link(1). Frame 2 rotates with Link (2) and is placed perpendicular to X_1 connecting link(1) and link(2). Lastly, Frame 3 rotates with link(3) and is connected with link(2). Frame 4 relates to the end-effector and is placed at the end of link 3 perpendicular to X_3[28]. Based on these defined frames, standard Danavit-Hartenberg (D-H) parameters [13] were determined as shown in table 3.5.

Standard D-H Parameters				
i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0°	0	L_1	θ_1
2	90°	0	0	θ_2
3	0°	$-L_2$	0	θ_3
4	0°	$-L_3$	0	0°

Table 3.5: Standard D-H Parameters

Definition of D-H parameters[28]:
α_i: Link twist or anlge about the X_i between (Z_i,Z_{i+1})
a_i: Link length or distance along X_i between (Z_i,Z_{i+1})
d_i: Link offset or distance along Z_i between (X_{i-1}, X_i)
θ_i: Joint angle or the angle about Z_i between (X_{i-1}, X_i)

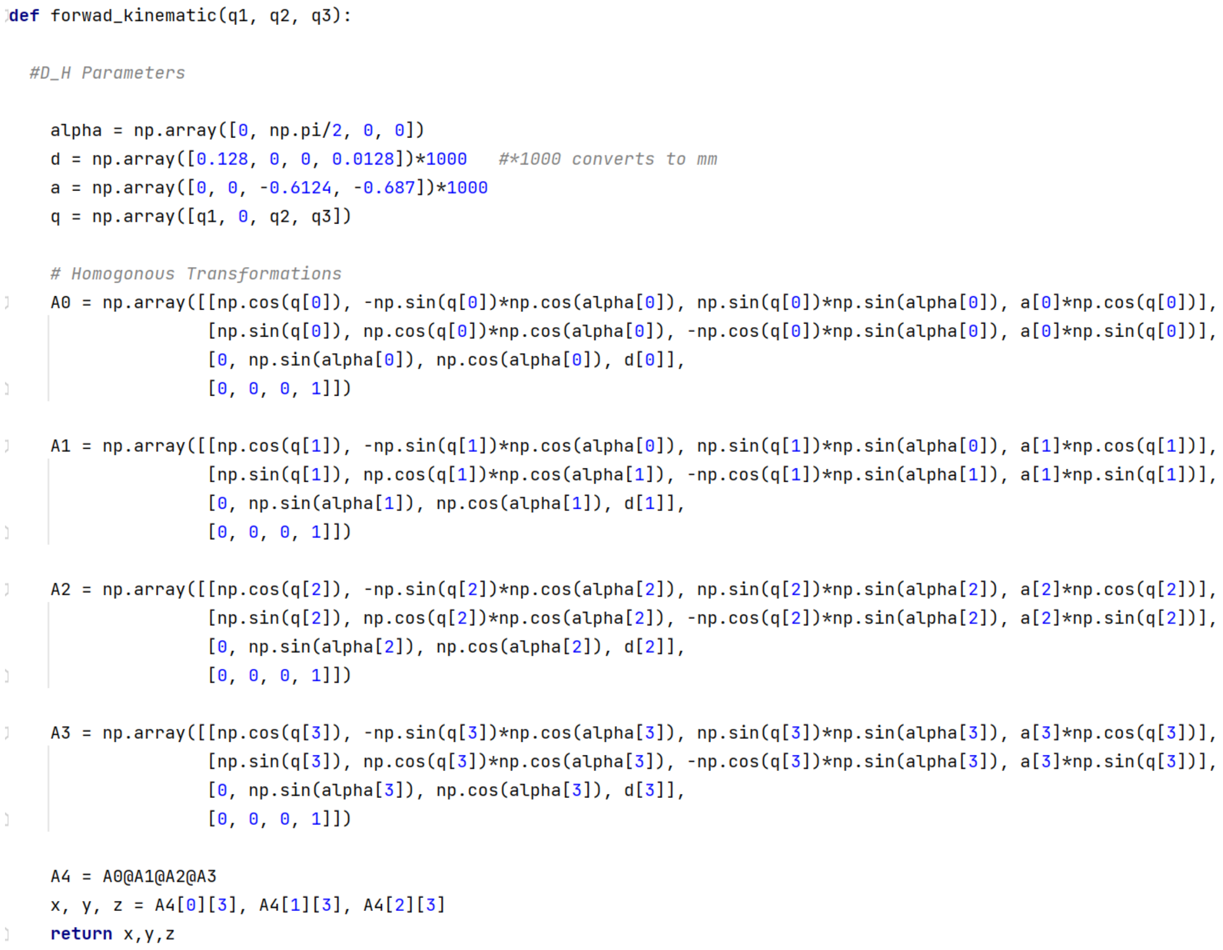
The standard D-H parameters are used to compute a total homogeneous transformation matrix which describes the kinematic relation from the end effector to all joints corresponding with every frame. To compute the transformation for every coordinate frame as described in chapter 2.4.($A_1^0, A_2^1, A_3^2, A_4^3$,) substitute the standard D-H parameters for the general formula for A_i^{i-1}[44]:

$$A_i^{i-1} = \begin{bmatrix} c_{\theta i} & -s_{\theta i}c_{\alpha i} & s_{\theta i}s_{\alpha i} & a_i c_{\theta i} \\ s_{\theta i} & c_{\theta i}c_{\alpha i} & -c_{\theta i}s_{\alpha i} & a_i s_{\theta i} \\ 0 & s_{\alpha i} & c_{\alpha i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.28}$$

Multiplying the homogeneous transformation matrices in the correct sequence describes the end-effector frame with the reference/base frame[28]:

$$A_4^0 = A_1^0 A_2^1 A_1^2 A_4^3 \tag{3.29}$$

The actual calculations for the forward kinematics is implemented as a function in Python. Figure 3.21 illustrates how the forward kinematics are calculated in Python. The first three elements in the final column of A_4^0 are the Cartesian coordinates of the origin of the end effector (frame 4) with respect to the reference/base (frame 0).

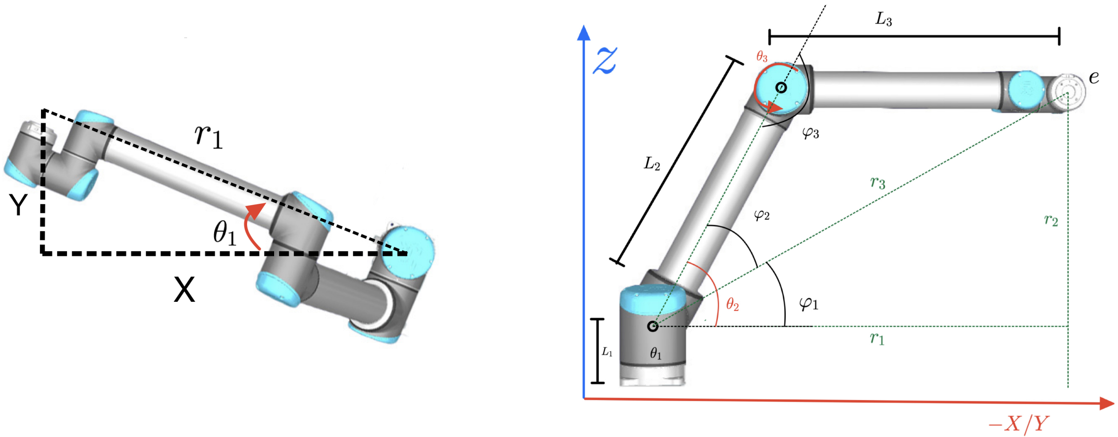


```
def forwad_kinematic(q1, q2, q3):

  #D_H Parameters

    alpha = np.array([0, np.pi/2, 0, 0])
    d = np.array([0.128, 0, 0, 0.0128])*1000   #*1000 converts to mm
    a = np.array([0, 0, -0.6124, -0.687])*1000
    q = np.array([q1, 0, q2, q3])

    # Homogonous Transformations
    A0 = np.array([[np.cos(q[0]), -np.sin(q[0])*np.cos(alpha[0]), np.sin(q[0])*np.sin(alpha[0]), a[0]*np.cos(q[0])],
                   [np.sin(q[0]), np.cos(q[0])*np.cos(alpha[0]), -np.cos(q[0])*np.sin(alpha[0]), a[0]*np.sin(q[0])],
                   [0, np.sin(alpha[0]), np.cos(alpha[0]), d[0]],
                   [0, 0, 0, 1]])

    A1 = np.array([[np.cos(q[1]), -np.sin(q[1])*np.cos(alpha[0]), np.sin(q[1])*np.sin(alpha[0]), a[1]*np.cos(q[1])],
                   [np.sin(q[1]), np.cos(q[1])*np.cos(alpha[1]), -np.cos(q[1])*np.sin(alpha[1]), a[1]*np.sin(q[1])],
                   [0, np.sin(alpha[1]), np.cos(alpha[1]), d[1]],
                   [0, 0, 0, 1]])

    A2 = np.array([[np.cos(q[2]), -np.sin(q[2])*np.cos(alpha[2]), np.sin(q[2])*np.sin(alpha[2]), a[2]*np.cos(q[2])],
                   [np.sin(q[2]), np.cos(q[2])*np.cos(alpha[2]), -np.cos(q[2])*np.sin(alpha[2]), a[2]*np.sin(q[2])],
                   [0, np.sin(alpha[2]), np.cos(alpha[2]), d[2]],
                   [0, 0, 0, 1]])

    A3 = np.array([[np.cos(q[3]), -np.sin(q[3])*np.cos(alpha[3]), np.sin(q[3])*np.sin(alpha[3]), a[3]*np.cos(q[3])],
                   [np.sin(q[3]), np.cos(q[3])*np.cos(alpha[3]), -np.cos(q[3])*np.sin(alpha[3]), a[3]*np.sin(q[3])],
                   [0, np.sin(alpha[3]), np.cos(alpha[3]), d[3]],
                   [0, 0, 0, 1]])

    A4 = A0@A1@A2@A3
    x, y, z = A4[0][3], A4[1][3], A4[2][3]
    return x,y,z
```

Figure 3.21: Forward Kinematics function, homogeneous transformations in python

Inverse Kinematics

As a function of the desired x, y, and z coordinates, inverse kinematics derives the analytical expressions of the rotational angles (i.e. θ_1, θ_2, and θ_3) for the base, shoulder, and elbow joints. Pythagoras' law and trigonometric principles can be used to solve the inverse kinematics. Inverse kinematics may be addressed by looking at the top and side views of the robot, respectively[18].



(a) Top view of robot

(b) Side view of robot.

Figure 3.22: Two different views of the robot.

The following summary of inverse kinematic equations are based on[18]:

From figure 3.22a equation 3.30 can be used to estimate the degree of the axis of the joint base

$$\theta_1 = atan2\left(\frac{Y}{X}\right) \tag{3.30}$$

where X and Y are the X-axis and Y-axis positions of the end-effector coordinates, and θ_1 is the degree of the joint base's angle[18].

A triangle guideline that depicts the lengths r_1, r_2, and r_3 as illustrated on the dotted green line in Fig. 3.22b is required to determine the degree of angle θ_2. The following equations are used to compute r_1, r_2, and r_3[18]:

$$r_1 = \sqrt{X^2 + Y^2} \tag{3.31}$$

$$r_2 = Z - L_1 \tag{3.32}$$

$$r_1 = \sqrt{r_1^2 + r_2^2} \tag{3.33}$$

Where Z is the Z- axis position of the end-effector coordinates and L_1 is the distance between the base and shoulder joint[18].
Furthermore, the angle degrees θ_2 and θ_3 may be derived from a side view of the 3-DOF robot arm. Several variables can be specified from the side, including $L_1, L_2, L_3, Z, \theta_2$, and θ_3. Where L_2 is the length of the connection between the shoulder and the elbow. L_3 is the length of the connection between the elbow and the end-effector. The horizontal line axis of the shoulder joint and the connection between the shoulder and elbow combine to produce θ_2. The horizontal line axis of the shoulder joint and elbow forms θ_3[18].

Looking at Fig 3.22b θ_2 can be calculated using the following equaiton[18]:

$$\theta_2 = \phi_1 + \phi_2 \tag{3.34}$$

Where

$$\phi_1 = atan2\left(\frac{r_2}{r_1}\right) \tag{3.35}$$

$$\phi_2 = \arccos\left(\frac{L_2^2 + r_3^2 - L_3^2}{2L_2L_3}\right) \tag{3.36}$$

Additionally, the following equation can be used to calculate θ_3:

$$\theta_3 = 180° - \phi_3 \tag{3.37}$$

Where

$$\phi_3 = \arccos\left(\frac{L_2^2 + L_3^2 - r_3^2}{2L_2L_3}\right) \tag{3.38}$$

The inverse kinematics are implemented as a function in Python.

3.8.3 Workspace

The workspace of the robot describes the various possible placements of the robot's end-effector in 3D space. The workspace is calculated analytically, based on the paper[28]. Using equation 3.24 from section 3.8.1, knowing cosine of the angles ranges between -1 and 1. The mathematical inequality shown below holds true [28].

$$-2L_2L_3 \le x^2 + y^2 + z^2 - 2L_1z + 2L_1^2 - (L_1^2 + L_2^2 + L_3^2) \le 2L_2L_3 \tag{3.39}$$

Simplification of equation 3.39 gives:

$$(L_2 - L_3)^2 \leq x^2 + y^2 + (z - L_1)^2 \leq (L_2 + L_3)^2 \tag{3.40}$$

Equation 3.40 presents a sphere with centre $(0, 0, L_1)$ with inner radius $(L_2 - L_3)$ and outer radius $(L_2 + L_3)$[28].

The robot arm is placed on a table. The working space must extend beyond the table's edge to accommodate the payload, as well as have restrictions to prevent the end effector from colliding with the ground below. As a result, the workspace is designated as a quarter sphere where $z \geq L_1$ and $y \leq (L2 - L3)$. To ensure the robot doesn't move outside the workspace, the solution for the base joint in equation 3.30 is constrained for results outside of range (0 - 180)°. Since the tan function is defined periodically with π, subtracting or adding π until the allowed solution is reached, constrains the base.

A 3D plot that portrays the arm's workspace was constructed in Matlab based on the accurate arm's link lengths and joint angle limitations, as shown in figure 3.23. In this plot, the workspace is defined by the green and red highlighted boundaries

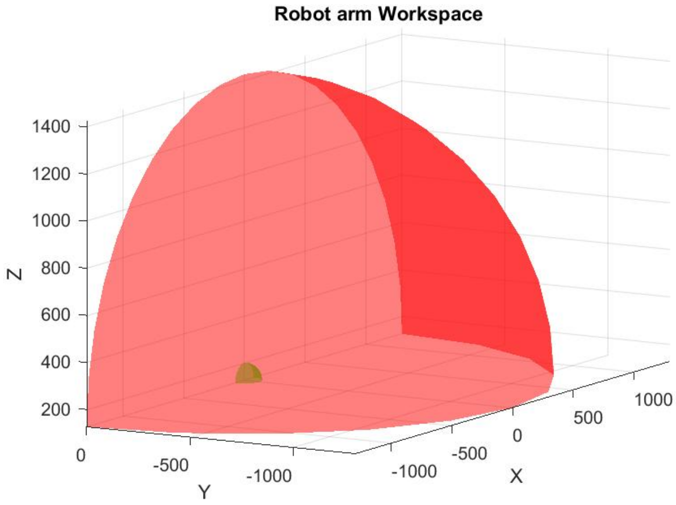
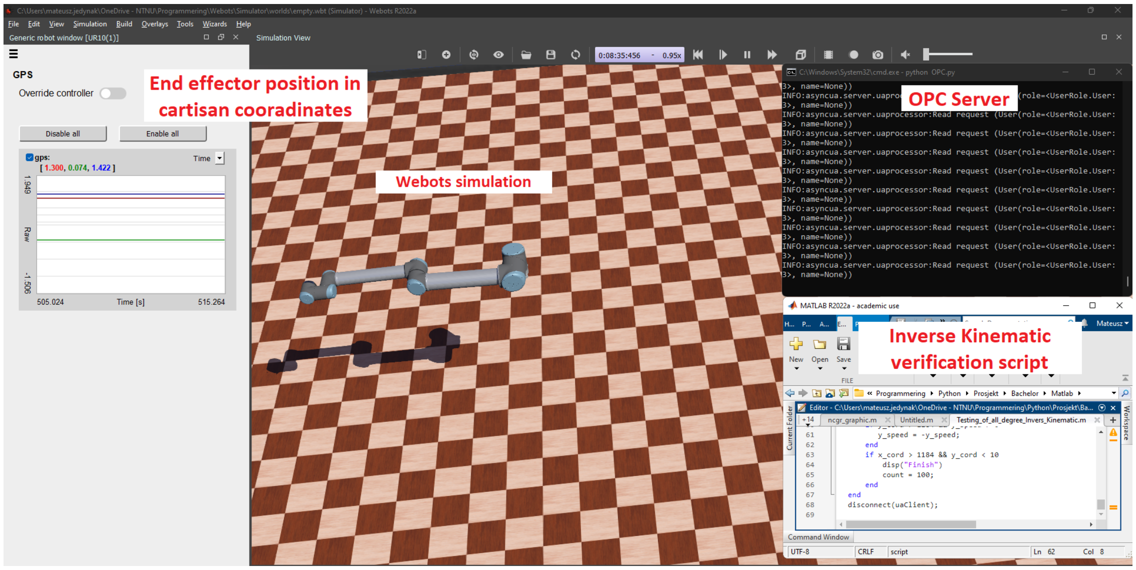


Figure 3.23: 3D plot of robot arm workspace with boundaries highlighted in red and green.

3.8.4 Verification through simulation

The mathematical models for both the inverse and forward kinematics are difficult to verify without a graphical simulation. The simulator in webots from section 3.7.2 is used to verify the models. The OPC UA server and clients mentioned in section 3.6 are used to transfer data from Matlab to Webots.

Matlab is used to script the kinematic models provided in sections 3.8.1 and 3.8.2. The calculated joint angles are then communicated to the simulator through the OPC server. A loop iterates the model's location from X: 0m, Y: -1.1184m, and Z: 0.0128m. The location is incremented until it reaches a goal of X: 1.184m, Y: 0.01m, and Z: 0.0128m. Figure 3.24 shows the Webots simulator, Matlab script and OPC Server combined to verify the solution.


End effector position in cartisan cooradinates

Webots simulation

OPC Server

Inverse Kinematic verification script

gps: [1.300, 0.074, 1.422]

```
3>, name=None))
INFO:asyncua.server.uapro                 (role=<UserRole.User:
3>, name=None))
INFO:asyncua.server.uaprocessor:Read request (User(role=<UserRole.User:
3>, name=None))
INFO:asyncua.server.uaprocessor:Read request (User(role=<UserRole.User:
3>, name=None))
INFO:asyncua.server.uaprocessor:Read request (User(role=<UserRole.User:
3>, name=None))
INFO:asyncua.server.uaprocessor:Read request (User(role=<UserRole.User:
3>, name=None))
INFO:asyncua.server.uaprocessor:Read request (User(role=<UserRole.User:
3>, name=None))
INFO:asyncua.server.uaprocessor:Read request (User(role=<UserRole.User:
3>, name=None))
INFO:asyncua.server.uaprocessor:Read request (User(role=<UserRole.User:
3>, name=None))
INFO:asyncua.server.uaprocessor:Read request (User(role=<UserRole.User:
3>, name=None))
INFO:asyncua.server.uaprocessor:Read request (User(role=<UserRole.User:
3>, name=None))
```

```
            y_speed = -y_speed;
        end
        if x_cord > 1184 && y_cord < 10
            disp("Finish")
            count = 100;
        end
    end
    disconnect(uaClient);
```

C:\Users\mateusz.jedynak\OneDrive - NTNU\Programmering\Webots\Simulator\worlds\empty.wbt (Simulator) - Webots R2022a
File Edit View Simulation Build Overlays Tools Wizards Help
Generic robot window [UR10(1)]
Simulation View
0:08:35:456 - 0.95x
GPS
Override controller
Disable all
Enable all
Time
1.949
Raw
-1.506
505.024
Time [s]
515.264
C:\Windows\System32\cmd.exe - python OPC.py
MATLAB R2022a - academic use
H... P... A... E...
Mateusz
New
Open
Save
FILE
« Programmering ▸ Python ▸ Prosjekt ▸ Bachelor ▸ Matlab ▸
Editor - C:\Users\mateusz.jedynak\OneDrive - NTNU\Programmering\Python\Prosjekt\Ba...
+14
ncgr_graphic.m
Untitled.m
Testing_of_all_degree_Invers_Kinematic.m
Current Folder
Workspace
Command Window
UTF-8
CRLF
script
Ln 62
Col 8

Figure 3.24: Illustration of inverse kinematic testing

A video showing how the aforementioned verification was carried out for the second kinematic model from section 3.8.2 is linked below.

https://youtu.be/bwDENS7TTP8

3.9 Controlling the robot

Several steps are needed in order to move and control the physical robotic arm. This section describes the methods used and developed to control and communicate with the robotic arm.

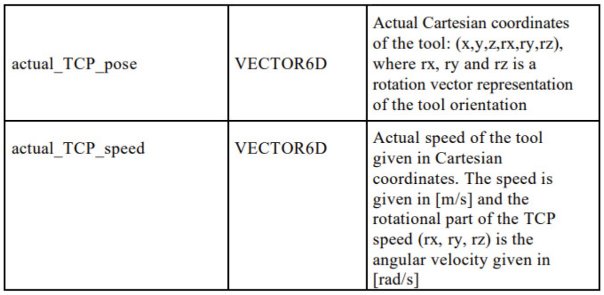
3.9.1 Safety

Working around a robotic arm can be dangerous, Especially when developing functionalities, and bypassing the robot controllers inbuilt functions. Therefore, when operating the robot, the emergency stop mounted on the robots control panel is at the ready. A detailed risk assessment is provided in appendix A.9.

3.9.2 Communication

The Real-Time Data Exchange *(RTDE)* interface enables synchronization between a client and the UR controller through a standard TCP/IP connection, without compromising the UR controller's real-time capabilities [43]. This functionality is critical for the project since it enables Fieldbus drivers to manipulate robot I/O and real-time operation for control and visualization of robot status [43].

The transmission frequency used in transmission was set to 60 Hz corresponding to the camera FPS. As a result, the control loop is capable of rapidly measuring and controlling the robot's end effector in real time. The registers for synchronization of variables on the client interface are defined in an xml-file. Certain variable names are predefined by universal robots. The rtde documentation offers information on the variable names that are predefined. Figure 3.25 shows the pre-configured variables that were primarily used in the project, which were obtained from the RTDE UR data-sheet.



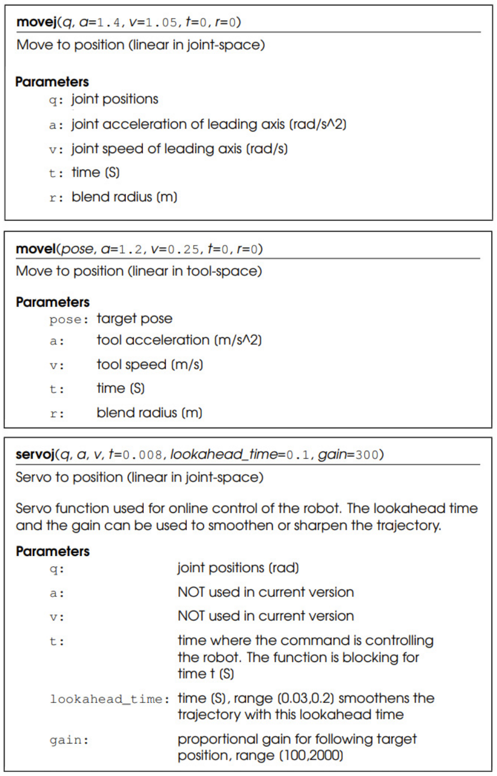
actual_TCP_pose	VECTOR6D	Actual Cartesian coordinates of the tool: (x,y,z,rx,ry,rz), where rx, ry and rz is a rotation vector representation of the tool orientation
actual_TCP_speed	VECTOR6D	Actual speed of the tool given in Cartesian coordinates. The speed is given in [m/s] and the rotational part of the TCP speed (rx, ry, rz) is the angular velocity given in [rad/s]

Figure 3.25: Predefined variables from Ur-controller output [43].

The variables *actual_TCP_pose* and *actual_TCP_speed* are used for receiving the robot-arms end effector position and speed in 6 DOF. The variables are given as 6D vectors, where each element is of datatype double *(IEEE 754 floating point, 64bit)* [43]. Furthermore, seven variables of datatype double send information to the robot controller. Six of those transmit the angle values for each joint. The final variable determines the state of the robot controller.

3.9.3 Actuation of joints

The robot controller supports different *actuation-commands* for actuating the UR10 joints. Figure 3.26 shows the data-sheet of the most common commands. The command *movej()* moves the robot linearly in joint space. This command has built in trajectory planning. The drawback is that the UR must finish the path before executing a new command. Figure 3.27 on the left side illustrates this problem. The *movel()* command has the same problem as movej, the difference being moving linearly in Cartesian space. Both commands utilizes the controllers built in kinematic calculations for all 6 joints. This limits the ability to emulate crane movements. To circumvent this issue, Real-time control of each individual joint is needed. This is done using the *servoj()* command. The advantage of *servoj()* is that it supports real-time applications and can permanently alter the final destination during a path, as illustrated in figure 3.26 on the right side. There is no built-in trajectory planning in the *servoj()* command. Therefore, a trajectory needs to be implemented in order to move the end effector smoothly.



movej(*q*, *a*=`1.4`, *v*=`1.05`, *t*=`0`, *r*=`0`)

Move to position (linear in joint-space)

Parameters

`q:` joint positions
`a:` joint acceleration of leading axis (rad/s^2)
`v:` joint speed of leading axis (rad/s)
`t:` time (S)
`r:` blend radius (m)

movel(*pose*, *a*=`1.2`, *v*=`0.25`, *t*=`0`, *r*=`0`)

Move to position (linear in tool-space)

Parameters

`pose:` target pose
`a:` tool acceleration (m/s^2)
`v:` tool speed (m/s)
`t:` time (S)
`r:` blend radius (m)

servoj(*q*, *a*, *v*, *t*=`0.008`, *lookahead_time*=`0.1`, *gain*=`300`)

Servo to position (linear in joint-space)

Servo function used for online control of the robot. The lookahead time and the gain can be used to smoothen or sharpen the trajectory.

Parameters

`q:` joint positions (rad)
`a:` NOT used in current version
`v:` NOT used in current version
`t:` time where the command is controlling the robot. The function is blocking for time t (S)
`lookahead_time:` time (S), range (0.03,0.2) smoothens the trajectory with this lookahead time
`gain:` proportional gain for following target position, range (100,2000)

Figure 3.26: Different actuation commands [21].

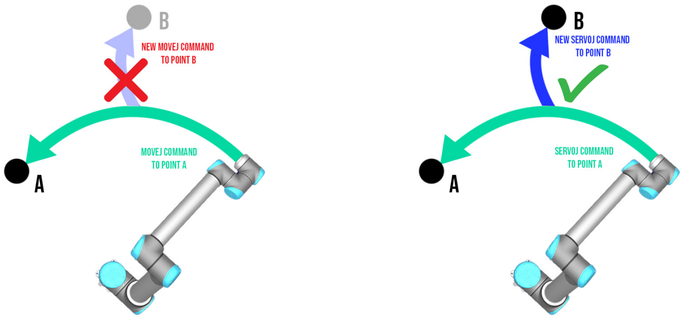


Figure 3.27: Illustration of command *movej()* (left), *servoj()* (right) with new endpoint during a path

3.9.4 Program structure of UR10-controller

The UR10 robot controller works as a server. The controller reads data from the client, executing command on the robotic arm, and updates the registers. The Ur10 robot controller is programmed in UR's software Polyscope. Figure 3.28 shows the program structure of the robot controller in flowchart form. First stages are a start up sequence before the program loops continuously until either connection is lost or emergency stop is detected. The final version of UR10 source code is provided in appendix A.2.

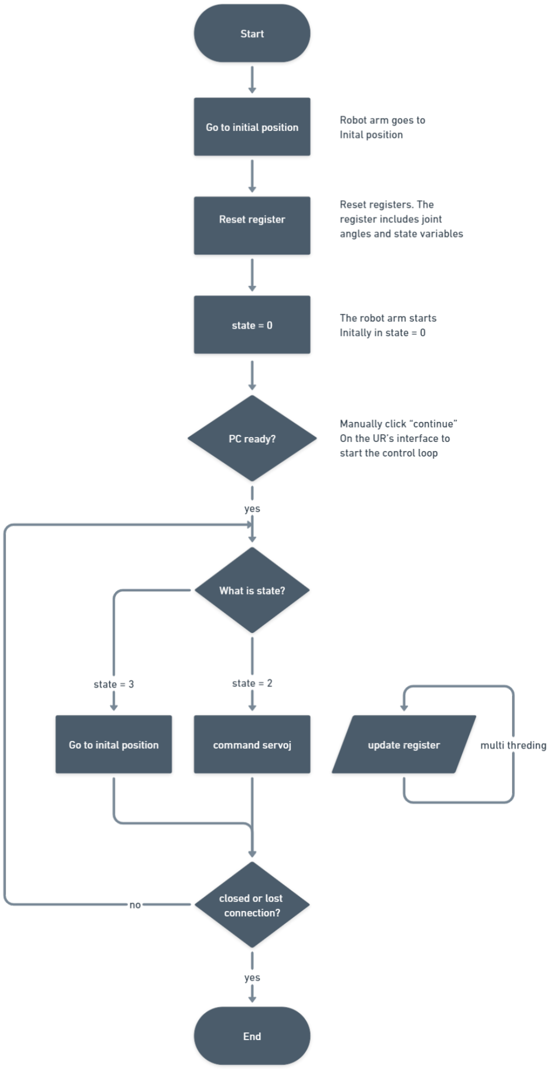


Figure 3.28: UR10 controller program structure in flowchart

3.9.5 Movement of end effector

With individual joint control allowed by the servoj command, a trajectory needs to be implemented to move the end effector more smoothly. The next position should be near the previous one, roughly along a continuous path to achieve a smooth transition.

In order to generate a trajectory, the path needs to be described using the control systems output. The original model described in section 3.4.3 yields the acceleration as the output parameter for the regulator. Analyzing the end effector's acceleration to a particular time proved to be more difficult than anticipated. We attempted two different approaches described below.

Trajectory with asymmetric constant acceleration

A first approach was implementing trajectory with asymmetric constant acceleration. The implementation in python was obtained from the textbook *Trajectory planning for automatic machines and robots* [22]. The advantage of using trajectory with asymmetric constant acceleration is that the flex point doesn't necessarily need to be at $t_f = (t_1 + t_0)/2$ [22].

The trajectory planning with asymmetric constant acceleration was used in several control loops. In some cases, trajectory planning was employed in conjunction with PID to achieve the desired location more smoothly, but frequently get updated to a new final destination.

Additionally, it was utilized to slowly move the end effector from point A to point B without swinging the payload.

Integrating acceleration

A second approach was to numerically integrate the acceleration twice to approximate the end effectors cartesian position. The numerical integration method uses the trapezoidal rule.

3.10 Detection of payload

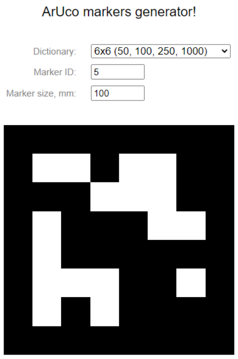
To compensate for the payload's pendulum motion, some form of object detection is needed. During the project, the group reviewed several possibilities for object detection. Initially, an IMU sensor was viewed as a method of estimating position. Due to the fact that the IMU readings could be affected by waves if the crane is placed on a ship. Based on the group's existing experience with machine vision, the group elected to focus on machine vision as the primary application. Further in this section, several machine vision implementations are described.

3.10.1 Machine vision algorithms

All parts of the object detection is implemented using Python. Python isn't the fastest programming language. Even so, it is robust and supports many libraries and features for machine vision. Some of these include color detection, ArUco marker detection and machine learning algorithms.

3.10.2 ArUco marker

One method for object detection is using an ArUco marker. ArUco uses Fiducial markers to detect the an object in a camera frame. ArUco is an open source library for python for detection of square markers. ArUco designates every fiducial marker with a unique ID. The markers were generated using the website: *https://chev.me/arucogen/*. Figure 3.29 illustrates an example ArUco marker with ID 5 and dictionary class 6x6. ArUco provides options over what resolution the markers inhabit. This is done by specifying the dictionary for a marker. Lower dictionary sizes and larger marker sizes, in general, increase detection distance from camera to the marker, and vice versa. The detection of larger markers, on the other hand, is more difficult due to the in-



ArUco markers generator!

Dictionary: 6x6 (50, 100, 250, 1000)

Marker ID: 5

Marker size, mm: 100

Figure 3.29: Illustration of ArUco marker [1].

creased number of bits that must be retrieved from the image [29]. For this project a 100x100mm size marker with dictionary *DICT_6X6_100* is used.
Additionally, ArUco has the possibility to detect the orientation of every marker. Figure 3.30 shows an example of detection of frame Axes to marker with use of Intel Realsense camera.

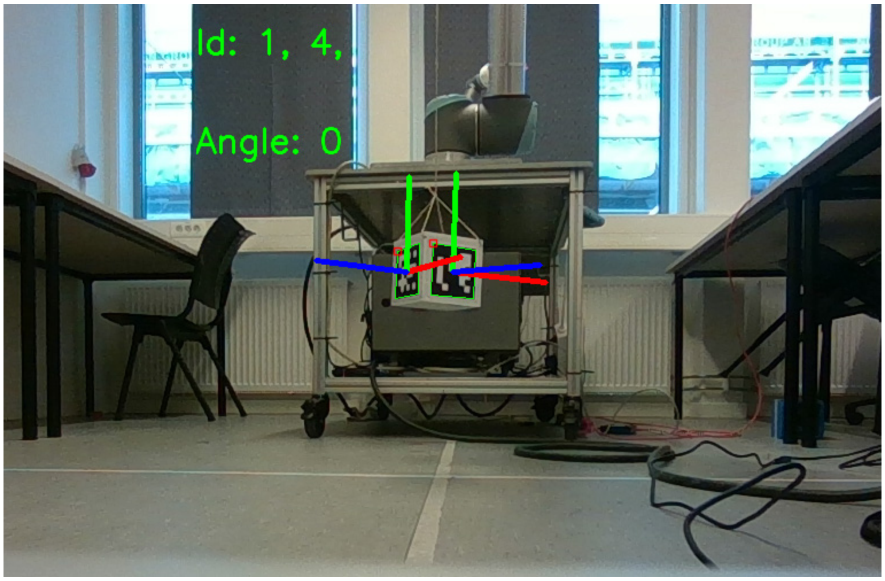


Figure 3.30: Example of use of ArUco marker algorithm

To incorporate an ArUco marker, the markers are placed on each side of the payload. Figure 3.31 show concepts where markers are placed on the payload prototype.

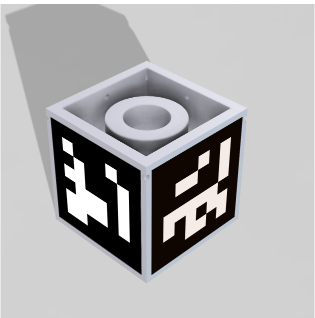


Figure 3.31: 3D-Model of payload prototype with ArUco Markers

To register the ArUco marker nothing can stay in front of the marker. Figure 3.32 show two ArUco markers. One on top of the payload and another on the left side of the payload.

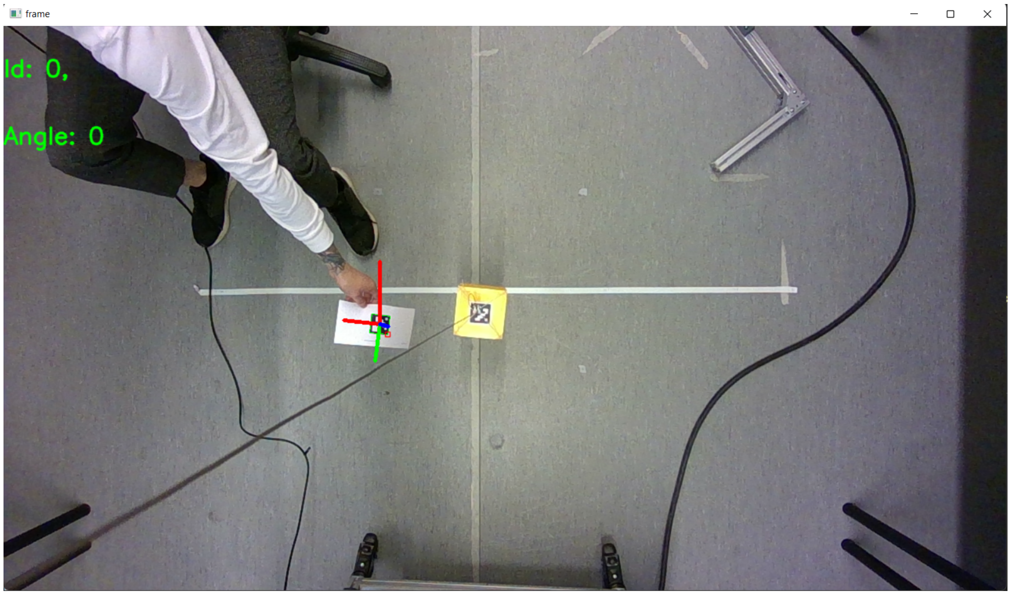


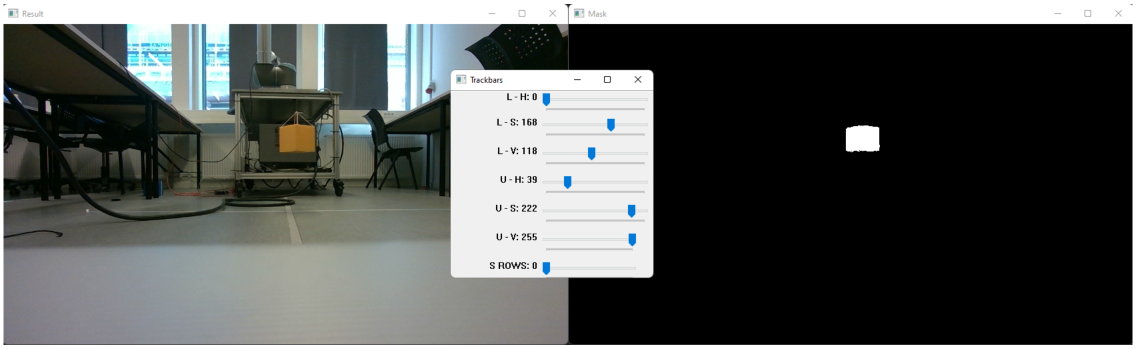
Figure 3.32: Detection of ArUco marker on the top of the payload and next to the payload

3.10.3 Color detection

Another method for object detection is color detection. Color detection uses Hue, Saturation and Brightness Value (HSV) to identify pixels in a image. In order to detect the correct colored object, color calibration is needed.

To preform color calibration, the OpenCV library in python is utilized. The first step is to read the camera frame and convert RGB/BGR picture to a HVS image. Secondly a track bar, to set lower and upper boundary to hue, saturation and value is made. The trackbar is made with a simple function *cv2.getTrackbarPos()* . All values are then saved as a two array list, one for lower boundary and second for upper boundary HSV parameters.

The Last step is creating a new frame mask. This mask checks for array elements between the camera picture frame array, and boundary HSV levels desired. Afterwards the result of the picture frame and HSV mask compared are displayed. Figure 3.33 shows an example of color calibration for a yellow payload.



L - H: 0
L - S: 168
L - V: 118
U - H: 39
U - S: 222
U - V: 255
S ROWS: 0

Figure 3.33: Color calibration example

The boundary values are then saved as a csv file for convenience. The csv file can later be imported and used to detect a colored object in a camera frame.

After acquiring the HSV parameters from calibration the next step is the actual detection. The raw camera input is transformed to a HSV image. Next, an integrated function from the OpenCV library *cv2.inRange()* is used. This function is used to "detect an object based on the range of pixel values in the HSV colorspace" [8]. The desired object parameters from the calibration are compared against the camera HSV image. The function *cv2.dilation()* is used to increase detection area. This also helps reducing noise from the image. The result is a detected object in the image frame.

Further, contours of the object are found using a OpenCV function *cv2.findContours()* [41]. Which retrieves contours from the binary image. This algorithm give out two variables, *Contours* and *Hierarchy*. Contours contains a list of every detected object with parameters such as size and position in pixel coordinates. Hierarchy include hierarchy of objects. In this case only one object is detected. Therefore hierarchy variables are never used.

The finished detection algorithm searches for the largest item. However, if the camera detects no item in the image frame, background noise from a little object with the same HVS scale could be problematic. Therefore, the program looks for objects with dimensions above 1500 pixels. This approach gives reduced measurement noise.

3.10.4 Machine Learning

Machine learning is another method of object detection. Various machine learning methods were considered during the project. The most promising was the *YOLOv5* framework.

YOLOv5 is an object identification algorithm that uses a grid approach to partition photos. Each grid cell is in charge of detecting items inside itself [14]. *YOLOv5* has extensive documentation on training, testing, and deployment on Github [15]. The group tested the completed database for object detection. However, employing the framework for the project would require creating a new data set. Not desiring to deviate from the project's objective and already possessing sufficient object detection methods. The team decided against pursuing machine learning further.

3.11 Camera Mounting and change of basis

For detecting the payload, two different camera mountings were considered. Firstly the camera was floor mounted with cameraview normal to the base zx-plane. Secondly the camera was mounted on the top of the robots end-effector with a veiw of the base xy-plane.

3.11.1 Camera coordinates

The camera coordinates are in the center of the Intel RealSense D455. The Z-axis of the camera is pointing straight outward. When staring straight at the camera, the Y-axis is pointing downwards and the X-axis is pointing left. Figure 3.34 shows the camera's coordinate system.

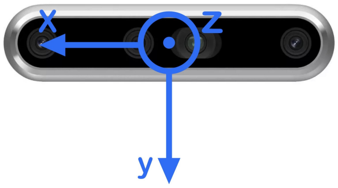
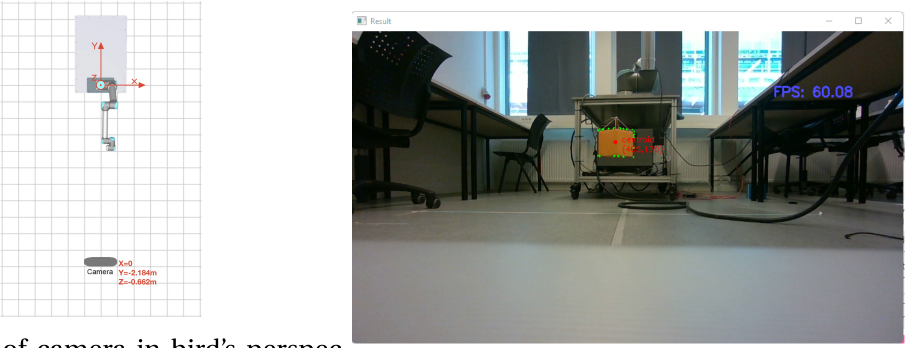


Figure 3.34: Camera coordinate frame [10]

3.11.2 Floor mounted camera

Mounting the camera stationary in front might not be practical for crane applications. In many cases, putting a camera in front of a crane will be difficult. However, because the camera is stationary towards the robot, this approach may be more intuitive to implement and was worth exploring. Figure 3.35 illustrates the camera's position on the floor as well as the view from the camera.



(a) Position of camera in bird's perspective.

(b) Direct vision from camera.

Figure 3.35: Position of camera and view from bird's perspective.

To detect the payload, Color detection algorithm described in section 3.10.3 was used. The position of x,y coordinates in pixel values are used to calculate the distance from the camera to the object. This is done by incorporating the stereo depth camera in the intel realsense. The *pyrealsense2* library in python, includes functions to detect distance in meters for a specific pixel in the camera frame. In some scenarios however, a particular pixel will return a value 0 because the camera cannot estimate the distance to the specific pixel.

Figure 3.36 illustrates the scenario where the camera cannot see a certain segment of the payload. The segment with missing pixels are marked with a blue circle.

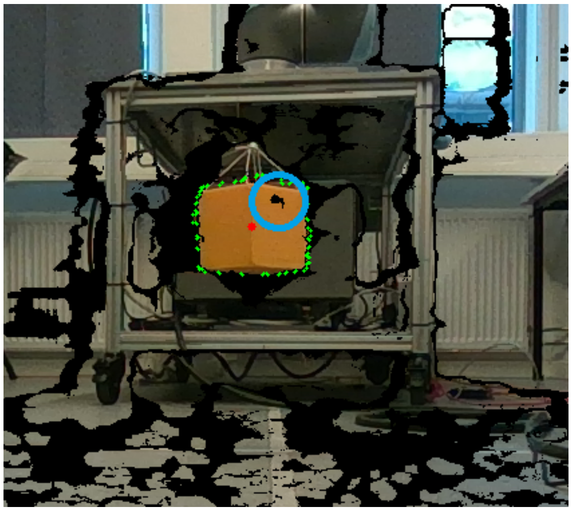


Figure 3.36: Example of missing segment from depth camera measurement

To circumvent this issue, five pixels around center were measured. The pixels with values of 0 are filtered out. After the filtering, the mean value of the remaining measurements were calculated in order to provide a accurate estimate of distance. Figure 3.37 shows an illustration of five points around the center of payload.

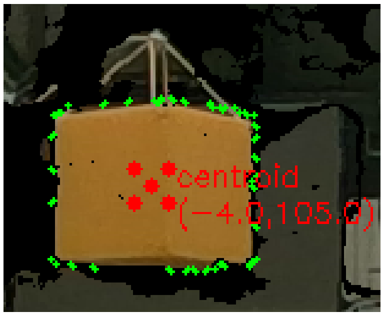


Figure 3.37: Example of five points used to estimate payload position.

Temporary detection

Following the detection of the payload, a temporary solution was required to determine the distance between the robot and the payload *(i.e Y-global coordinate)*. This was accomplished by subtracting the distance between the camera and the payload from the distance between the camera and the robot. In the temporary solution pixels were utilized to detect the position in the X-direction. Whereas a meter unit was used to measure the depth to payload in y direction.

After implementing the temporary solution, a more permanent robust solution was needed. The goal then became to transform the floor mounted cameras coordinates to global coordinates.

Pixel to meter conversion

The conversion from pixel to meter unit is required to eliminate scaling factors and for transforming to global coordinates. The intrinsic matrix described in section 2.8.2 is used to derive the correlation between 3D and 2D coordinates. The Realsense2 library includes a function that returns the intrinsic matrix. The intrinsic matrix contains the internal parameters specific to the Intel Realsense camera. The function *rs2_deproject_pixel_to_point()* from the Realsense2 library was used to calculate from 2d to 3d for the camera. The required inputs to the function are the intrinsic matrix, a pixel coordinate and the distance. These parameters allow the function to return the X,Y,Z camera coordinates of the given pixel in meters.

Floor mounted camera coordinates to global coordinates

The extrinsic matrix remains constant since the camera is fixed. The extrinsic matrix described in section 2.8.1 is used transform the camera coordinates to global coordinates. The transformation consists of a translation and a rotation. The translation from base/global center to camera were measured and give: $x = 0m, y = -2.184m$ and $z = -0.662m$. This is shown in figure 3.35a. The camera to global coordinates are aligned with a rotation matrix around the x-axis. The rotation is fixed as a -90°rotation. The transformation is summerized in equation 3.41 and 3.42. The resulting global coordinates X,Y,Z are marked in red, whilst the camera coordinates X,Y,Z are marked in blue.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-90°) & -\sin(-90°) \\ 0 & \sin(-90°) & \cos(-90°) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T \tag{3.41}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} 0 \\ -2.184 \\ -0.662 \end{bmatrix} \tag{3.42}$$

3.11.3 Top mounted camera

Mounting the camera on the end effector could be beneficial and useful. Because the rope on the robot is fixed and can be measured, the camera doesn't require stereo vision for distance measurement. Before mounting the camera on the end effector, a variety of techniques need addressing and implementation.

Sixth joint rotation compensation

The camera must be horizontal to always keep the payload in the camera's scope *(i.e normal to the xy plane)*. A knuckle boom crane is a 3-DOF system. To handle the cameras planer orientation, an extra degree of freedom is required. The camera is kept horizontally steady by using the UR10's 3.2.1 sixth joint (θ_6) to correct for the tilt caused by joints θ_2 and θ_3 Figure 3.38 illustrates the compensation for the cameras planer orientation.

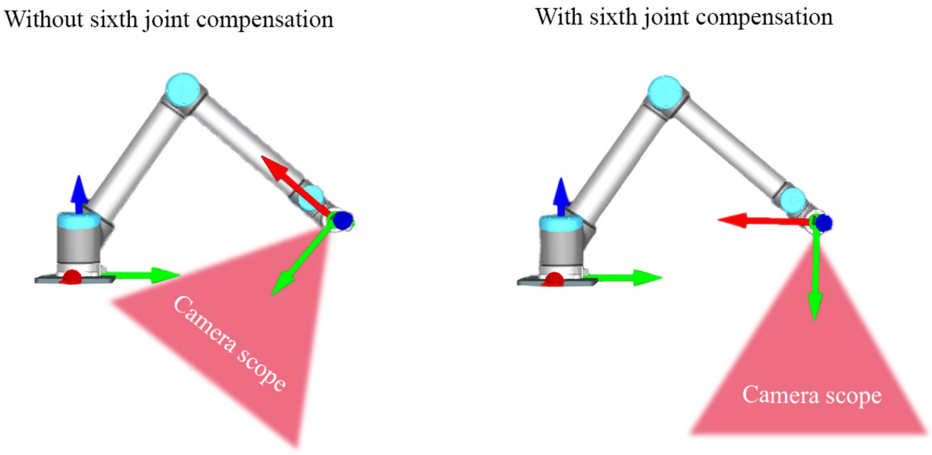

Without sixth joint compensation
With sixth joint compensation
Camera scope
Camera scope

Figure 3.38: Rotation compensation

θ_6's normal angle to the plane is found by adding θ_2 and θ_3. Additionally, adding 90°includes the final joints static offset from the initial configuration. Equation 3.43 yields:

$$\theta_6 = \theta_2 + \theta_3 + 90° \tag{3.43}$$

Transforming camera coordinates to base/global coordinates

To transform the camera coordinates to base coordinates the cameras extrinsic matrix needed to be derived. Since the camera doesn't have a stationary position about the arms base this proved more challenging than for the previous side mounted camera.

The extrinsic matrix, described in chapter 2.8.1 consists of a rotation and translation. The translation is described by the cameras translation relative to the base. Since the camera is mounted on top of the robot, the robots known end-effector position is used as translation. The translation is the same as the position described by the forward kinematics from chapter 3.8.2, including a slight offset.

The camera's rotation in relation to the base was difficult at first, but with a different approach, it became much easier. Initially, through rotation matrices, an attempt to derive the orientation transformations was made. Different rotation matrices around the Z and Y axes were multiplied. The transformation obtained some correct values in some positions, but not in others.

The second method involved fitting the coordinate system to a few known rotations of the base and then finding a general solution for all rotations. Figure 3.39 illustrates the base and cameras coordinate-system, with the base joint of the robot arm in 0°and 90°.

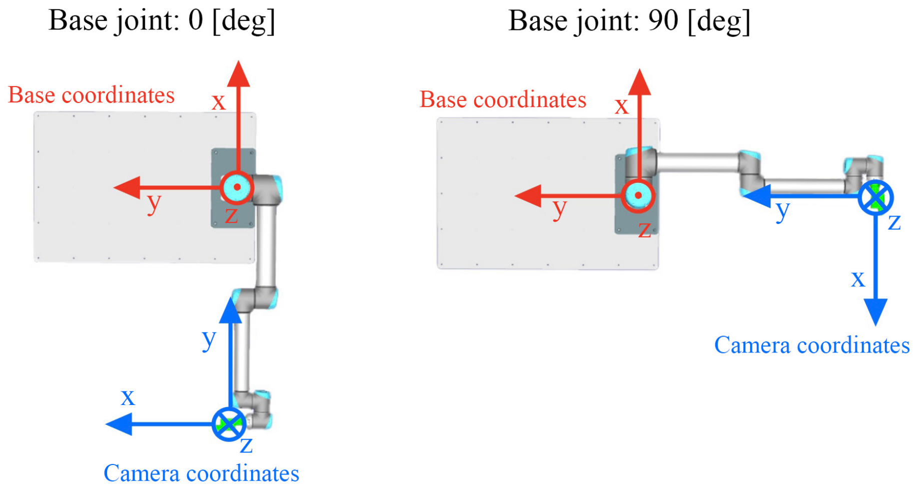


Figure 3.39: Camera and base coordinates system top view

Figure 3.39 shows a misalignment in camera and base coordinate orientation. With 0°at the base joint, the cameras X and Y axis should be switched, and the z axis should be flipped in the opposite direction, to align with the base coordinates. The transformation to base coordinates are shown in equation 3.44. For the equations X,Y,Z in blue are camera coordinates and X,Y,Z in red are base coordinates. T notes the translation vector.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T \tag{3.44}$$

With the base joint at 90°the cameras Y axis is a match and the Z and X axes should point in opposite direction, to align with the base coordinates. The transformation to base coordinates is shown in equation 3.45.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T \tag{3.45}$$

To make a general solution for all base joint angles. The trigonometrical functions $\cos(q_1)$ and $\sin(q_1)$ are used to fit for both equation 3.44 and 3.45 into a general solution. The general solution is shown in equation 3.46.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -\sin(q_1) & \cos(q_1) & 0 \\ \cos(q_1) & \sin(q_1) & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T \tag{3.46}$$

3.12 Control loop Simulation

Control-loop simulations of the model described in section 3.4.3 were carried out using various control-loop designs. Because the dynamics in the xz and yz planes are the same, the model is only simulated in the xz plane. In a Matlab/Simulink environment, the simulation will simulate the robot arm's end effector and the payloads motion.

3.12.1 Modular PID controller

The first implementation of simulation of the pendulum system was two feedback loops combined into a single system input, in form of acceleration of the end effector. One PID compensates for the payloads angular velocity, while the other controls the end effector's position. Figure 3.40 shows the control loop implemented in Simulink with use of the damped statespace pendulum system in xz plane with full order state.

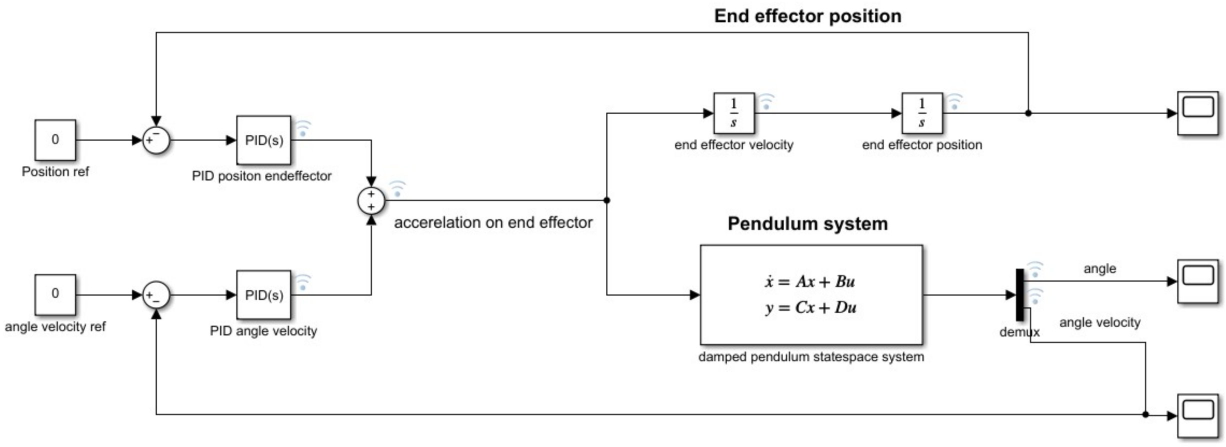


Figure 3.40: Diagram over modular PID control in Simulink.

The purpose of controlling the position of the end effector is to prevent it from drifting away. The physical arm reach is limited and may risk colliding with obstacles if the end-effector can drift freely.

The outputs of both PID controllers were practically polar opposites. As a result, optimising the relationship between each PID was just as crucial as tuning the parameters separately in order to successfully manage the angular velocity of the payload and the position of the end effector. Both controllers were fine-tuned by trial and error. This simulation's results are shown in section 4.2.1.

3.12.2 Modular PID controller with squared error

In most cases, the end effector should be able to control the payload more freely when the end effector is close to it's reference location. This can be done by squaring the error of the end effector's position while maintaining the sign. This technique provides a much lower input value to the PID when the error is minor and a significantly higher value when the error is larger because of the squaring. Figure 3.41 shown an implementation of squared position error in Simulink. This allows the angular velocity controller to be more dominant when the end effector is close to its reference.

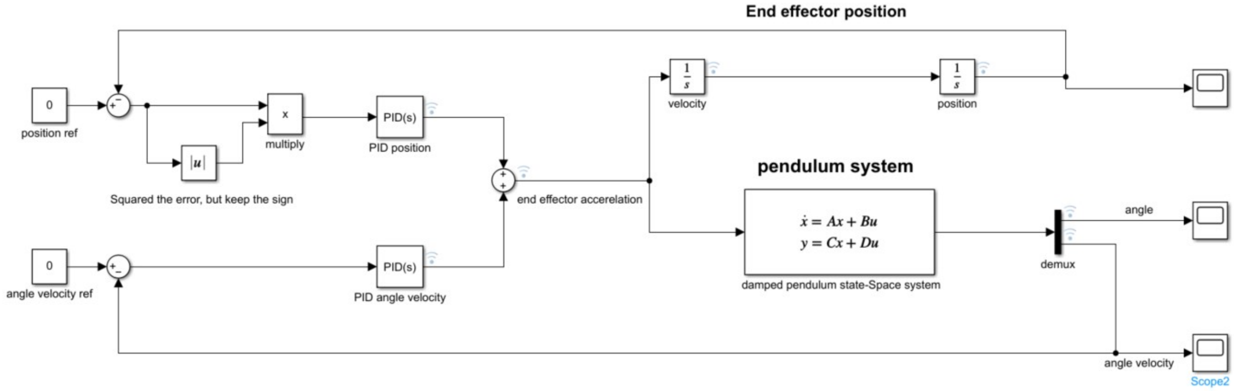


Figure 3.41: Diagram over modular PID control with squared error in Simulink.

The PID controllers were tuned by trail and error. The results of this simulation is shown in results section 4.2.2.

3.12.3 Modular LQR and PID

An LQR controller for angle and angular velocity control was implemented in Simulink for a full order feedback pendulum system. The feedback gain is calculated through the built-in lqr-function in Matlab. A PID controller controls the end effector's position.

With experience from the previous simulations, the angular controller performs significantly slower when combined with a position controller. Therefore, when designing the LQR, the Q and R are weighted to provide a fast system. The weights are shown in equation 3.47.

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad\qquad R = 0.5 \tag{3.47}$$

The feedback gain calculated by the LQR are shown in equation 3.48.

$$K_1 = 2344 \qquad\qquad K_2 = 3567 \tag{3.48}$$

Figure 3.42 illustrats the LQR feedback gain and PID position controller in Simulink.

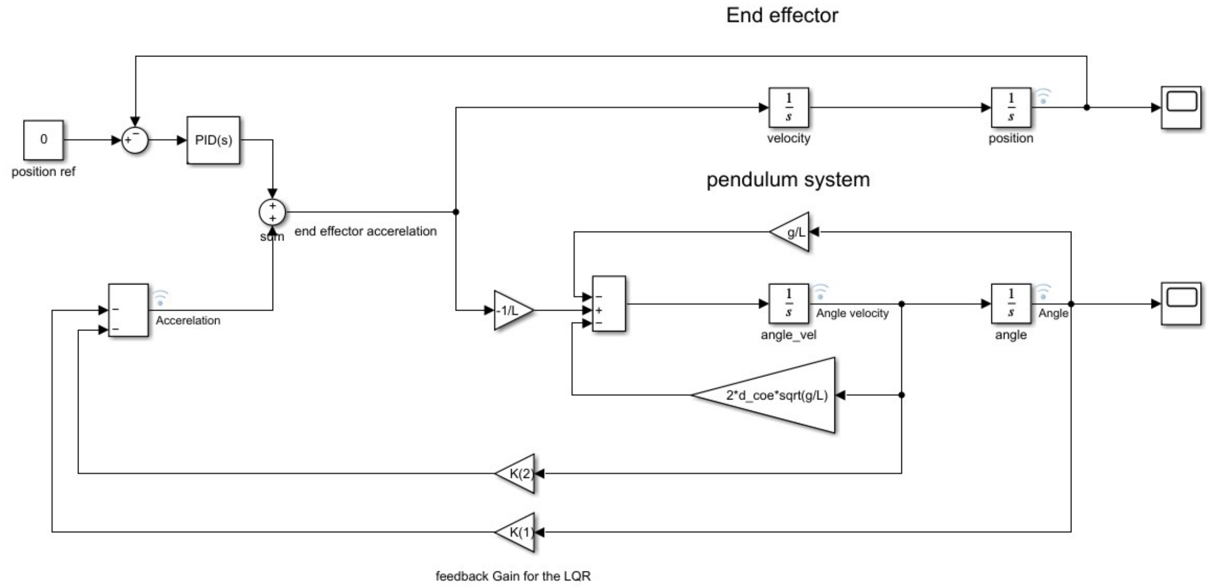


Figure 3.42: Diagram LQR and PID control-system simulation in simulink.

3.13 Control System Implementation

After combining object detection, robot control and communication into a functioning system, we began implementing regulators on the prototype.

To compensate for the payload's swaying motion, we examined several control loops. The team has worked with and implemented PID controllers in the past. In addition, literature dealing with comparable systems has shown that a PID controller may produce good results [44]. Therefore, a PID controller was implemented on the prototype. The controllers mentioned below, are all implemented in Python. The PID output is the required position of the robot's end effector. The payload is detected using color detection.

3.13.1 PID Controller with position input

First implementation of PID-controller

The PID controller was implemented to control end effector of UR10 to regulate payload to the reference point. The controller compensates for the payload cartesian position in the XY plane. Therefore, The PID controller consists of two separate control loops. One for the X-direction, the other for the Y-direction.

During the initial stage, the camera was placed on the floor and the depth camera was utilized to determine the distance between the camera and the payload. This controller was created prior to transforming camera coordinates to global coordinates, mentioned in section 3.11.2. Consequently, the x-direction controller utilizes pixel units from the camera, whereas the y-direction controller employs meter units from the depth camera.

Both controllers were adjusted to accommodate the control loop. For the measurements in the X-direction, The camera measures values between -640 and 640 pixels. Therefore, the camera coordinates are scaled to fit between -0.8 and 0.8 meters using the scaling function described in section 2.7.

For the measurements in the Y-direction, the depth camera measures distances between 0.5 and 1.5 meters. Hence, the measurements were scaled between -0.3 and -1.4 to approximate the global coordinate system.
The parameters for both PID controllers were determined by trial and error.

PID-controller with floor mounted camera

Subsequently, the transformation of camera coordinates to global coordinates, as mentioned in section 3.11.2, makes measurement scaling unnecessary. Hence, the PID controllers were tuned to accommodate global coordinates from the floor-mounted camera. All PID parameters were determined using trial and error.

PID-controller with top mounted camera

After successfully mounting the camera on the robot's end effector and transforming the camera coordinates to global coordinates, the control-loop described previously in section was re-used. All PID parameters were modified using trial and error.

3.13.2 PID Controller with angle input

PID with restricted end effector movement

The preceding controllers use the payload location as an input to account for pendulum motion. An alternative option is to compensate for the angle between the end effector and the payload. In order to observe the difference between the two methods, a new PID controller was implemented. This time, the angle will serve as the input. The controller controls the angle of the payload in the ZX and ZY planes. Two parallel control loops provide output locations for the robot's end effector.

Equation 3.49 and 3.50 provide the angle between the robot's end effector and the payload for both x and y-direction.

$$\theta = \arcsin\left(\frac{\text{payload}_x - \text{endeffector}_x}{\text{wirelenght}}\right) \tag{3.49}$$

$$\phi = \arcsin\left(\frac{\text{payload}_y - \text{endeffector}_y}{\text{wirelenght}}\right) \tag{3.50}$$

In order to observe the difference between the two methods, a new PID controller was implemented. This time, the angle will serve as the input. The controller regulates the angle of the payload in the ZX and ZY planes. Two parallel control loops provide output for the system. The controller's error is calculated by subtracting the measured angle from the reference angle. The reference angle is 0, where the payload is stationary. The robot's reference location is added to the PID's output. This allows the robot to compensate for the sway around a given reference location.

PID with unrestricted end effector movement

After observing the performance of the previous controller, it is evident that restricting the end effectors' movement made the system slower. Therefore, the control loop structure was changed. Instead of adding the PIDs output to a reference location, the output is instead converted to end effector positions by rearranging equations 3.49 and 3.50.

$$\text{end effector}_x = \text{payload}_x - \left(\text{wirelenght} \cdot \sin\theta\right) \tag{3.51}$$

$$\text{end effector}_y = \text{payload}_y - \left(\text{wirelenght} \cdot \sin\theta\right) \tag{3.52}$$

Equations 3.51 and 3.52 yields a cartesian position for the robot arm. This permits the system to evaluate the angle of the payload without regard to the position of the end effector.

When the angle is below a threshold of ± 0.04 radians (2.29 degrees), And after a slight time delay. The arms end effector moves towards a reference position, using trajectory planning. The time and speed of the arms motion are dependent on the euclidean distance between the end effectors current position and its reference point. To calculate the trajectory's travel time, the euclidean distance is multiplied by an arbitrary scaling factor. Further, making the trajectory sufficiently smooth to prevent oscillations along its journey.

3.14 System architecture

The general structure of the system architecture is represented in 3.43.

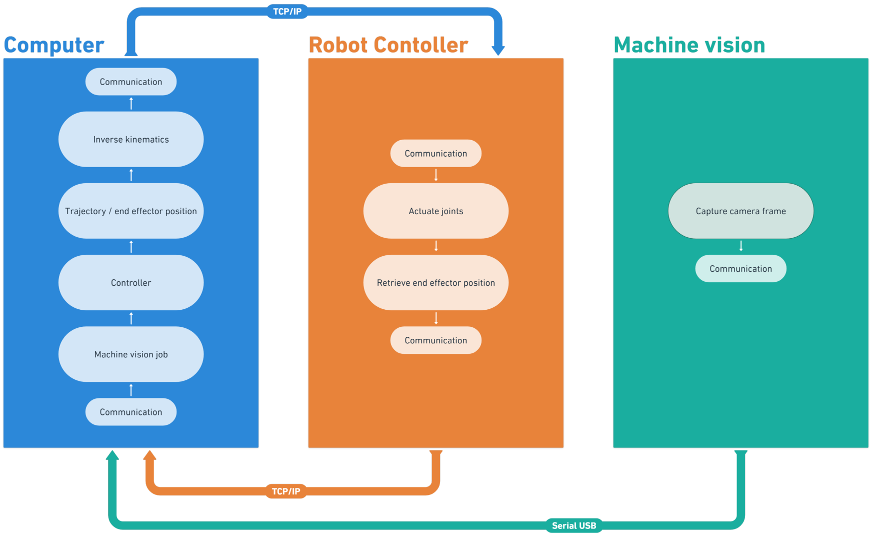
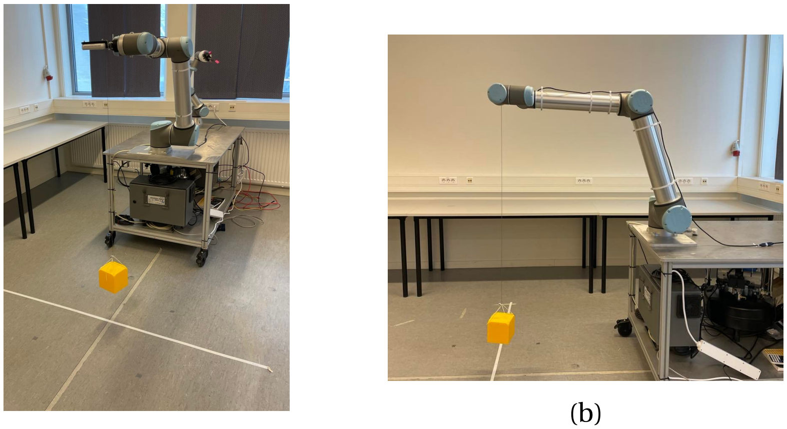


Figure 3.43: System Architecture

The prototype's finalized version is comprised of aspects and components created throughout the duration of the project. Figure 3.44 shows the finalized prototype. As previously explained in section 3.11.3, the camera is mounted to the end effector. The payload is detected using color detection as described in section 3.10.3. The kinematics derived in section 3.8.2 are used to compute the desired joint angles. The controllers mentioned in section 3.13 are used to regulate the pendulum motion. All of the aforementioned components are implemented in a local Python script. The desired joint angles are transmitted to the controller of the UR10 robot, which then executes the motion.



(a)

(b)

Figure 3.44: Finished prototype.

3.15 Testing setup

In order to test and document the performance of different control loops on our prototype and provide reliable results over several iterations, a testing rig/setup was made. The rig consists of 40x40 aluminium profiles, brackets and screws.

The rig is portable, allowing us to document the performance from the different control loops in the X and Y direction separately and X and Y combined. Image 3.45 shows the rig, while images 3.46, 3.47 and 3.48 show the rig in different testing scenarios.

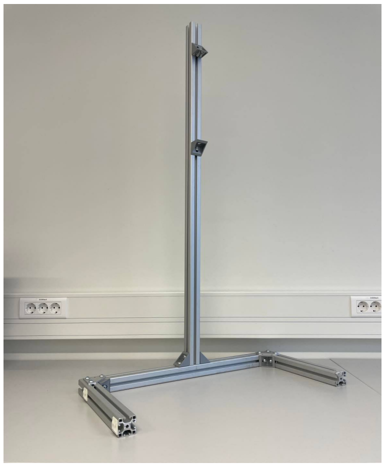


Figure 3.45: Testing rig

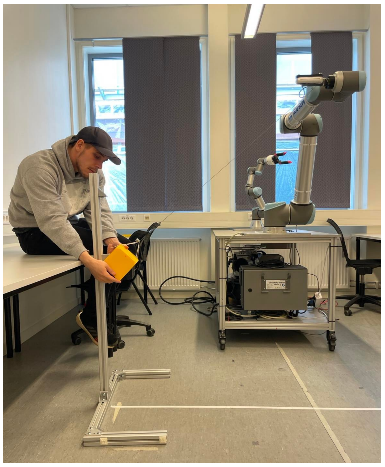


Figure 3.46: Testing in X - direction

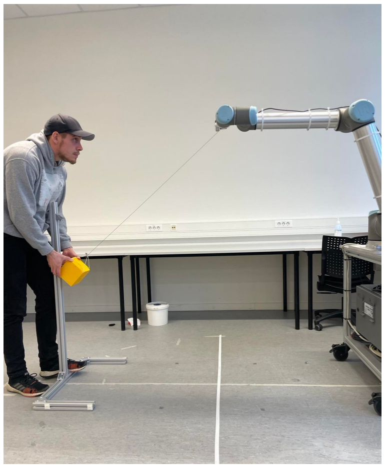


Figure 3.47: Testing in Y - direction

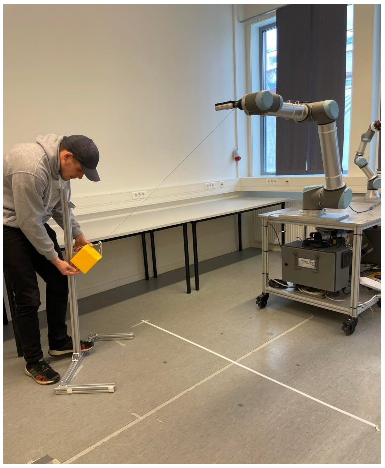


Figure 3.48: Testingin XY directions

A separate script was created for each controller in order to log data. The logging and regulation starts once the payload crosses the X-axis. After a specified time, the logging stops and the results are saved as a csv file.

Chapter 4

Result

The following chapter contain the results of the project's different components and implementations. The chapter's content include an estimation of the model's damping ratio. The model compared to the real system. In addition, simulation of the modelled system using a modular PID controller that yields a 12-second settling time. Including, simulation of Modular PID controller with squared error, with an improved settling time of about 8 seconds. Also, simulation of modular LQR and PID with an approximate settling time of 10 seconds. Furthermore, the outcomes for the first and second kinematics models, with the second model giving the correct solutions. additionally, a plot of trajectory planning is simulated with accurate results.

The results of mounting the camera on the floor and the end effector following, the machine vision application. The maximum inaccuracy on the floor is 34mm, while the maximum error on the top is -13mm. The highest inaccuracy for the camera mounted on top was 18.6mm when the payload was stationary and the camera was moved to various locations to estimate the payload's position.

Lastly, the results for the controllers implemented on the prototype, are presented.

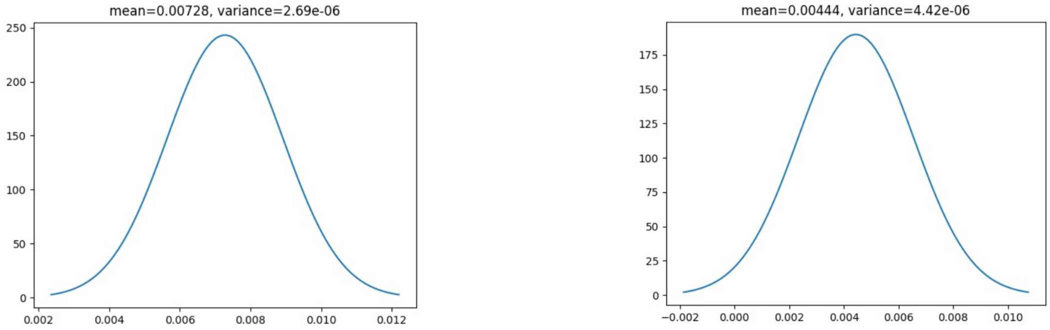
4.1 Modeling

Below shows the result for the system model and parameter estimation.

4.1.1 Parameter estimation

Damping ratio

The damping ratio were determined by evaluating the measurements described in Section 3.4.3. Figures 4.1a and 4.1b shows the normal distribution of the estimated damping ratio for the first and second measurements, respectively.



(a) First measurement.

(b) Second measurement.

Figure 4.1: Normal distribution of damping ratio "ζ" for both measurements.

4.1.2 Undamped model

Figure 4.2 shows the *undamped* model derived in section 3.4.2 compared to measurements from the real system. The initial angle for both is ≈ 0.4 rad or 23°.

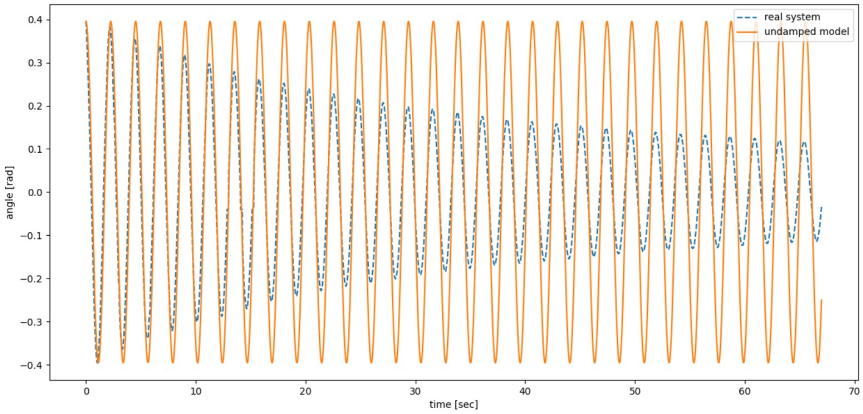


Figure 4.2: Undamped model compared to measurement from the real system.

4.1.3 Damped model

Figure 4.3 shows the damped model derived in section 3.4.3 compared to measurements from the real system. The damping ratio ζ in the damped model is 0.00728. The initial angle for the model and the real system is ≈ 0.4 rad or 23°.

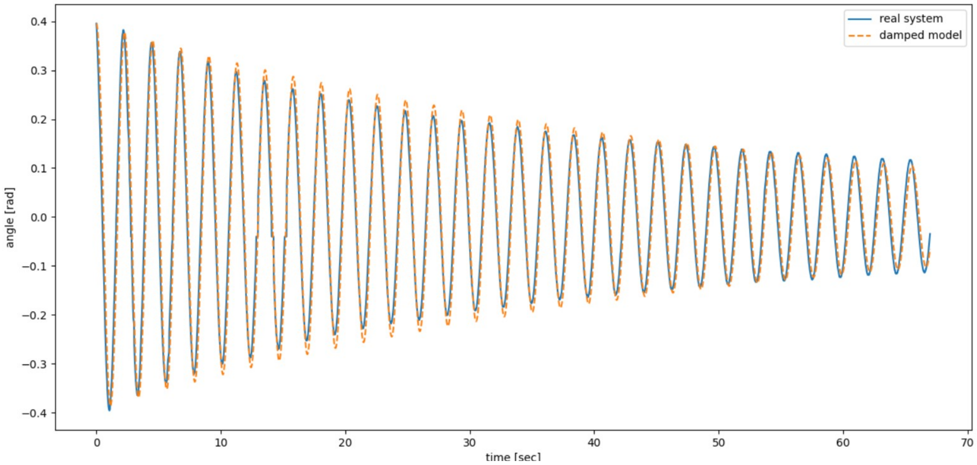


Figure 4.3: Damped model compared to measurement from the real system.

Figure 4.4 shows the damped model derived in section 3.4.3 compared to a second measurement from the real system. The damping ratio ζ in the damped model is 0.00444. The initial angle for the model and the real system is ≈ 0.43 rad or 24°.

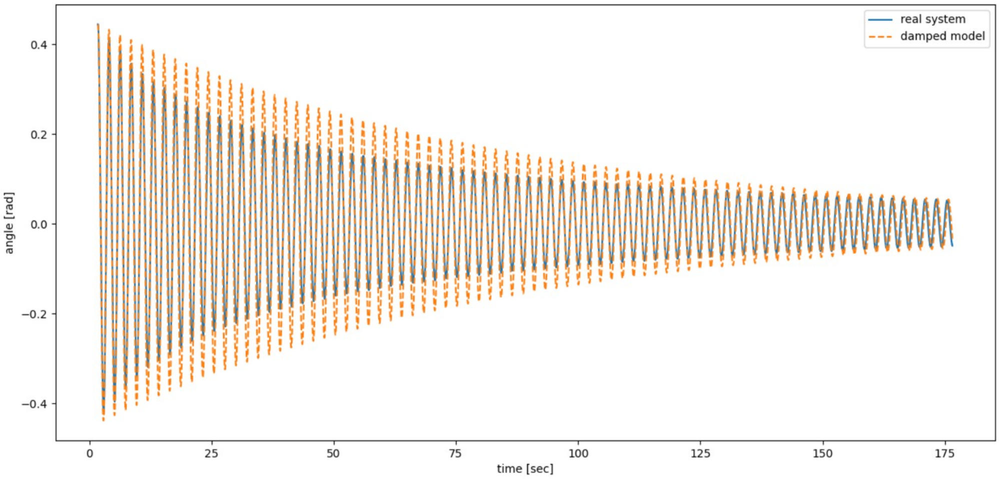


Figure 4.4: Damped model compared to a second measurement from the real system.

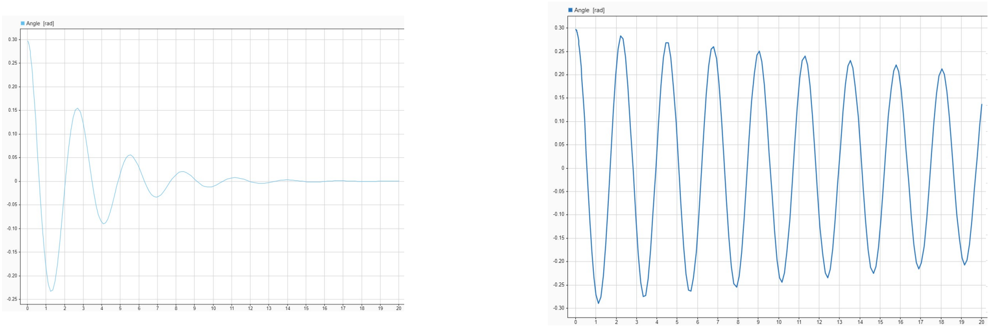
4.2 Control loop simulation

4.2.1 Modular PID controller

Figure 4.5 and 4.6 shows results for a Simulink simulation of the modular PID controller, described in section 3.12.1. Figure 4.5a shows a plot of the payload angle for the closed loop simulation. Figure 4.5b shows a plot of the payload angle for the open loop simulation. Figure 4.6a shows a plot of the end effector's position during the closed loop simulation. Figure 4.6b shows a plot with output from the angular velocity-PID, the position-PID and their combined output. Table 4.1 shows the PID parameters used in the simulation.

PID parameter	
PID controller:	K_p, K_i, K_d
Angle velocity PID	1.8, 0.001, 0.005
End-effector position PID	1.4, 0, 2

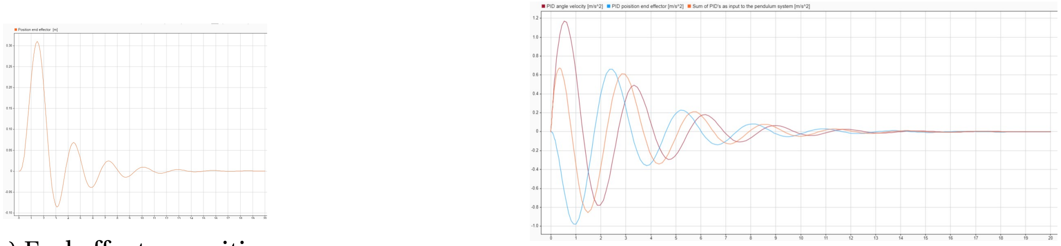
Table 4.1: PID parameters in modular PID controller



(a) Closed loop

(b) Open loop

Figure 4.5: Plot of payload angle for closed and open loop modular PID controller simulation.



(a) End effector position.

(b) Outputs of angular velocity-PID, position-PID and combined.

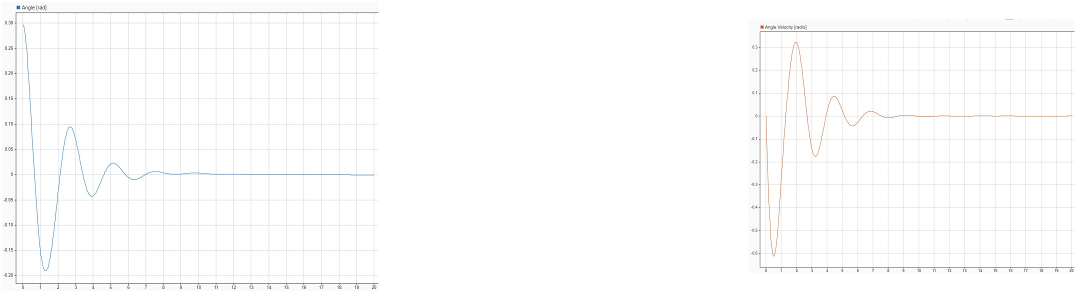
Figure 4.6: Plot of end effector position and PID outputs from the modular PID controller simulation.

4.2.2 Modular PID controller with squared error

Figure 4.7 and 4.8 shows results for a closed loop Simulink simulation of the modular PID controller with squared error, described in section 3.12.2. Figure 4.7a shows a plot of the payload angle for the simulation. Figure 4.7b shows a plot of the payload angular velocity for the simulation. Figure 4.8a shows a plot of the end effector's position for the simulation. Figure 4.8b shows a plot with output from the angle velocity-PID, the position-PID and their combined output. Table 4.2 shows the PID parameters used in the simulation.

PID parameters	
PID controller:	K_p, K_i, K_d
Angle velocity PID	1.8, 0.001, 0.005
End effector position PID	1.4, 0, 2

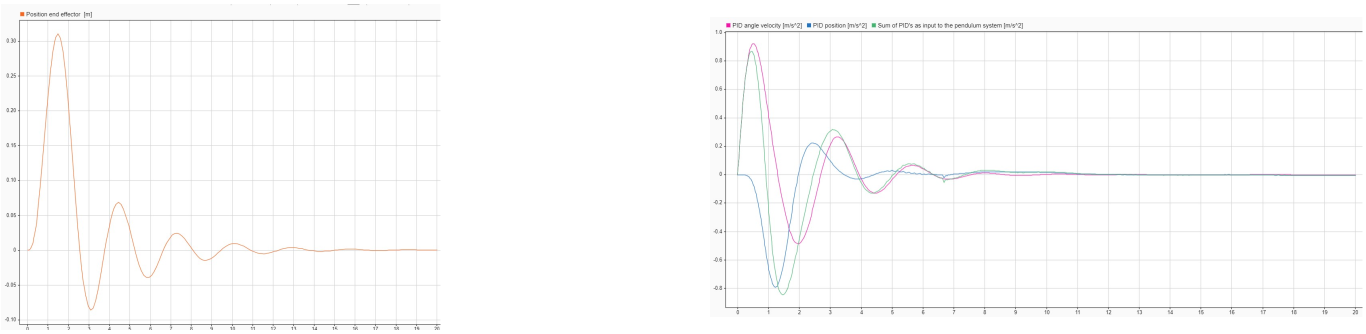
Table 4.2: PID parameters in modular PID controller with squared error



(a) Payload angle

(b) Payload angular velocity

Figure 4.7: Plot of payload's angle and angular velocity for modular PID controller with squared error in a closed loop simulation.



(a) End effector position.

(b) Outputs of angular velocity-PID, position-PID and combined.

Figure 4.8: Plot of end effector position and PID outputs from the modular PID controller with squared error simulation.

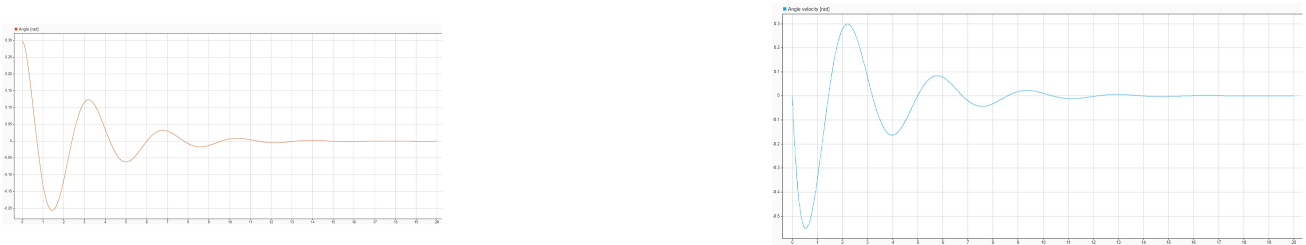
4.2.3 Modular LQR and PID

Figure 4.9 and 4.10 shows results for a closed loop Simulink simulation of the modular LQR and PID controller, described in section 3.12.3.

Figure 4.9a shows a plot of the payload angle for the simulation. Figure 4.9b shows a plot of the payload angular velocity for the simulation. Figure 4.10a shows a plot of the end effector's position for the simulation. Figure 4.10b shows a plot with output from the full order state feedback, the position-PID and their combined output. Table 4.3 shows the PID and feedback gain parameters used in the simulation.

Controllers parameters	
PID controller:	K_p, K_i, K_d
End effector position PID	1.4, 0, 2
Feedback gain:	$K_{\textbf{angle}}$, $K_{\textbf{angular velocity}}$
gain from LQR	0.05508, 3.3857

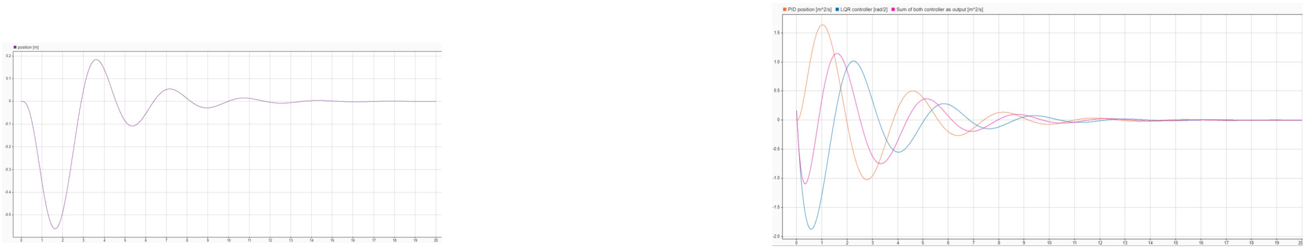
Table 4.3: PID parameters and feedback gain in modular LQR and PID



(a) angle

(b) angle velocity

Figure 4.9: Plot of payloads angle and angular velocity for modular LQR and PID simulation.



(a) Position of end effector

(b) Output from Position-PID, full order state feedback and combined.

Figure 4.10: Plot of end effector position along with position-PID output and full order state feedback output from the modular LQR and PID simulation.

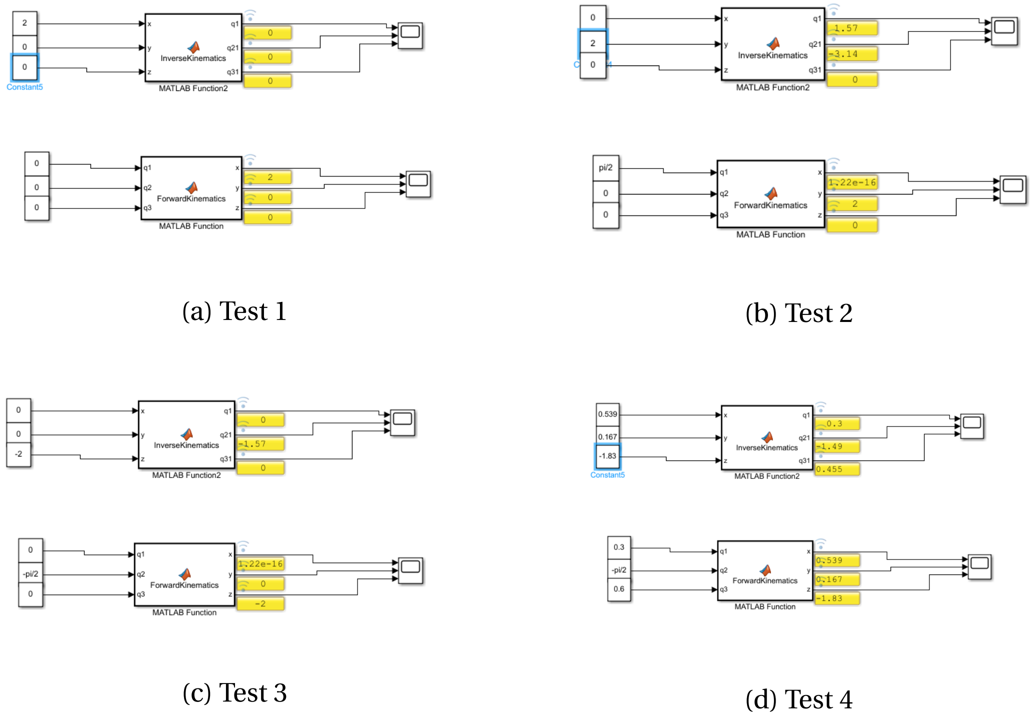
4.3 Kinematics

The Kinematic solutions for both models described in sections 3.8.1 and 3.8.2 are validated in Matlab/Simulink. The results show a simulation where distinct Cartesian coordinates are converted via inverse kinematics, which are then passed to forward kinematics for comparison.

A video showing the results of both kinematics models, simulated in Webots is linked below. In the video the first model is on the left and the second is on the right
`https://youtu.be/cx6vYecd0sM`.
Another video showing testing of the kinematics on the real system is shown in the link below.
`https://youtu.be/du6FzrQY2eo`
In the video the prototype is made to trace a spiral sphere.

4.3.1 First Model

Figure 4.11 shows kinematic solutions for distinct configurations. The link lengths for the model are set to $L_1 = 0m, L_2 = 1m, L_3 = 1m$.



(a) Test 1

(b) Test 2

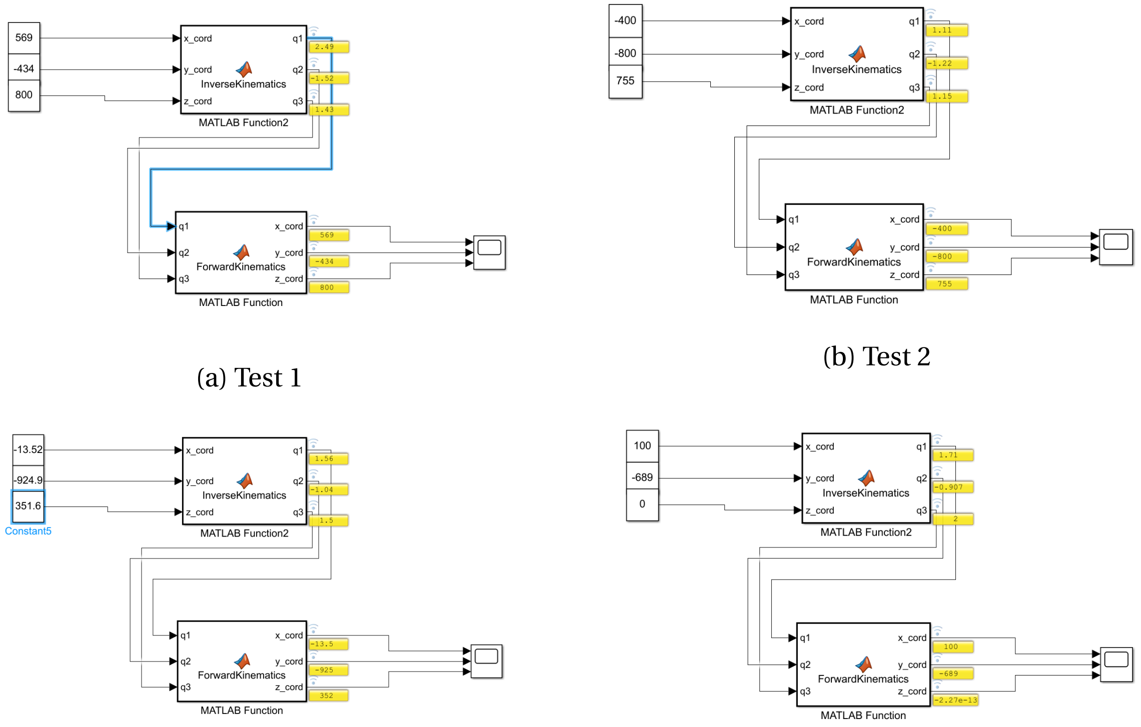
(c) Test 3

(d) Test 4

Figure 4.11: First model Kinematic solutions from forward and inverse in Simulink/Matlab.

4.3.2 Second Model

Figure 4.12 shows kinematic solutions for distinct configurations. The link lengths for the model are set to $L_1 = 0.128m, L_2 = 0.612m, L_3 = 0.688m$.



(a) Test 1

(b) Test 2

(c) Test 3

(d) Test 4

Figure 4.12: Kinematic solutions from forward and inverse in Simulink/Matlab.

In figure 4.13 a sine wave with amplitude 100mm is used as input to verify the inverse and forward kinematics.

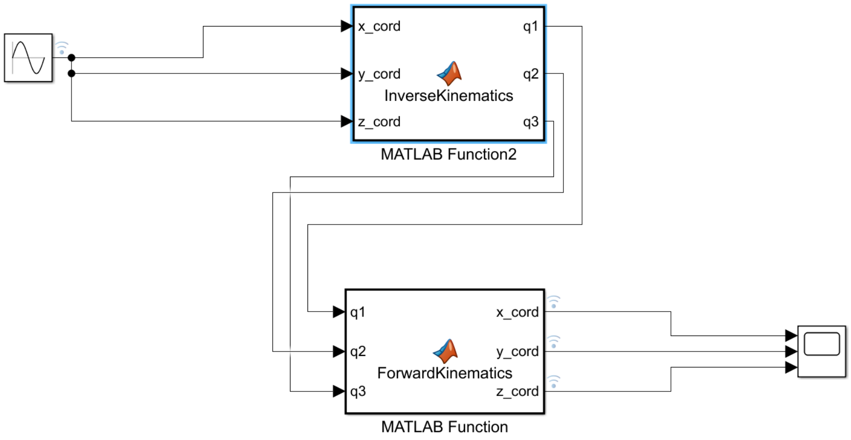
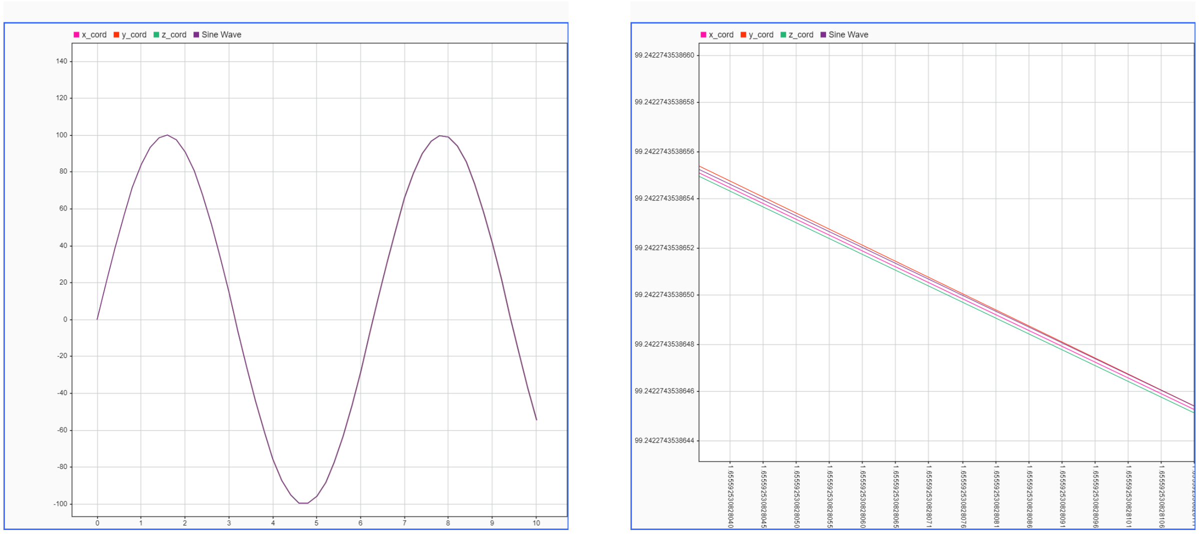


Figure 4.13: Inverse and Forward verification with Sine wave input.

Figure 4.14 shows the simulation results. The sine wave input to the inverse kinematic, along with the calculated cartesian positions from the forward kinematics.



(a) Kinematic solution with sine wave input

(b) zoomed in

Figure 4.14: Plot of inverse and forward kinematics output, from Simulink simulation.

4.4 Trajectory planning

Initial conditions for this trajectory is: start position: $q_0 = 0$, end position: $q_1 = 10$, start velocity: $v_0 = 1$, end velocity: $v_1 = 0$, start time: $t_0 = 0$, end time: $t_1 = 4$, flex point: $t_f = 1$.
Figure 4.15 shows a trajectory implemented in python environment.

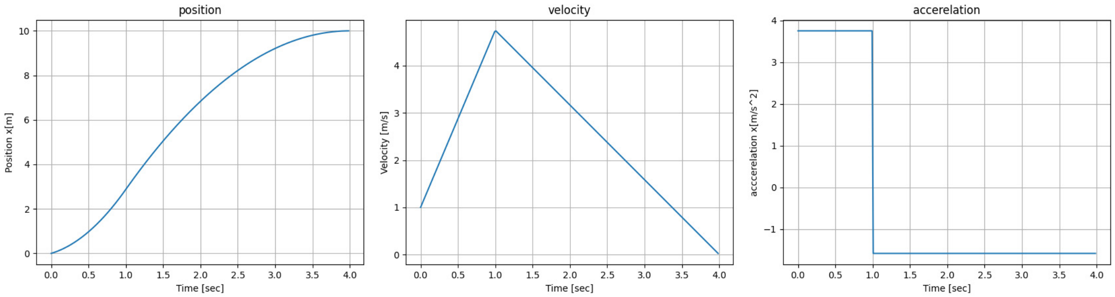


Figure 4.15: Trajectory with asymmetric constant acceleration

4.5 Machine vision and change of basis

The results from machine vision and transformed camera coordinates are shown below. The payload is detected using color detection described in section 3.10.3.

4.5.1 Floor mounted camera

Table 4.4 shows results with the floor mounted camera from section 3.11.2. The table shows the test of estimate the global coordinates from five fixed payload positions, with corresponding error. The cameras resolution is set to 1280x720.

Estimated global coordinates from floor mounted camera		
Payload coordinates (X,Y,Z) [mm]	**Estimated coordinates (X,Y,Z) [mm]**	**Error from payload coordinates (X,Y,Z) [mm]**
$(-320,-768,-755)$	$(-354,-744,-755)$	$(34,-24,0)$
$(-320,-1318,-755)$	$(-349,-1289,-755)$	$(29,-29,0)$
$(0,-1004,-755)$	$(6,-977,-755)$	$(-6,-27,0)$
$(320,-768,-755)$	$(323,-738,-755)$	$(-3,-30,0)$
$(320,-1318,-755)$	$(323,-1284,-755)$	$(-3,-34,0)$

Table 4.4: Estimated global coordinates from five fixed payload positions with corresponding error.

4.5.2 Top Mounted Camera

Cameras resolution at 1280x720

Table 4.5 shows results with the top mounted camera from section 3.11.3. The table shows the test of estimate the global coordinates from five fixed payload positions, with corresponding error.

Estimated global coordinates from top mounted camera		
Payload coordinates (X,Y,Z) [mm]	**Estimated coordinates: (X,Y,Z) [mm]**	**Error from payload coordinates (X,Y,Z):[mm]**
$(-320,-768,-755)$	$(-316,-755,-755)$	$(-4,-13,0)$
$(-320,-1318,-755)$	$(-324,-1307,-755)$	$(4,-11,0)$
$(0,-1004,-755)$	$(1,-1008,-755)$	$(-1,4,0)$
$(320,-768,-755)$	$(323,-772,-755)$	$(-3,4,0)$
$(320,-1318,-755)$	$(318,-1314,-755)$	$(2,-4,0)$

Table 4.5: Estimated global coordinates from five fixed payload positions with corresponding error. The camera resolution is 1280x720.

Cameras resolution at 848x480

Table 4.6 shows results with the top Mounted camera from section 3.11.3. The table shows the test of estimate the global coordinates from five fixed payload positions, with corresponding error.

Estimated global coordinates from top mounted camera		
Payload coordinates (X,Y,Z) [mm]	**Estimated coordinates: (X,Y,Z) [mm]**	**Error from payload coordinates (X,Y,Z):[mm]**
$(-320,-768,-755)$	$(-320,-754,-755)$	$(0,-14,0)$
$(-320,-1318,-755)$	$(-327,-1310,-755)$	$(7,-8,0)$
$(0,-1004,-755)$	$(0,-1007,-755)$	$(0,3,0)$
$(320,-768,-755)$	$(320,-772,-755)$	$(0,4,0)$
$(320,-1318,-755)$	$(320,-1316,-755)$	$(0,-2,0)$

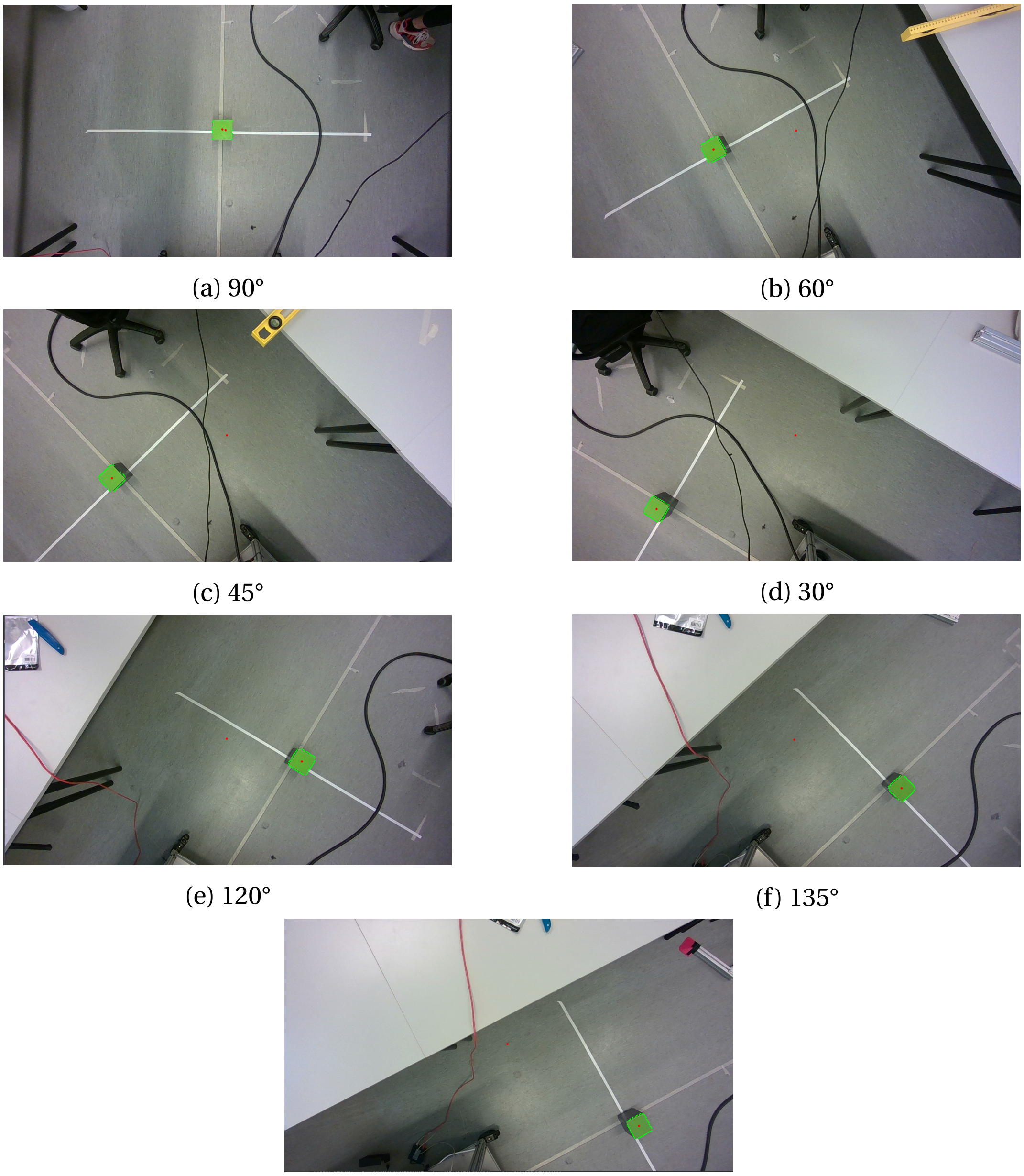
Table 4.6: Estimated global coordinates from five fixed payload positions with corresponding error. The camera resolution is 848x480.

Estimated global coordinates at various locations within the camera view

Table 4.7 shows results from the cameras estimated global coordinates for a stationary positioned payload at various locations within the camera view. The payload is under the top mounted camera while the base-joint and the camera frame rotates. The cameras resolution is set to 1280x720. Figures 4.16a - 4.16g displays the camera view for the different base joint configurations.

Estimated global coordinates from various camera views.			
joint-base angle:	**Estimated coordinates: (X,Y,Z) [mm]**	**Error from reference (X,Y,Z):[mm]**	**figure:**
90°(ref)	(−14.88, −985.99, −677.68)	(0, 0, 0)	4.16a
60°	(−22.07, −986.98, −677.68)	(7.19, 0.99, 0)	4.16b
45°	(−25.89, −986.30, −677.68)	(11.01, 0.31, 0)	4.16c
30°	(−33.48, −984.05, −677.68)	(18.6, −1.94, 0)	4.16d
120°	(−15.72, −978.80, −677.68)	(0.84, −7.19, 0)	4.16e
135°	(−18.22, −978.33, −677.68)	(3.34, −7.66, 0)	4.16f
150°	(−21.96, −975.88, −677.68)	(7.08, −10.11, 0)	4.16g

Table 4.7: Table over estimated global coordinates with different base-angles with corresponding error.



(a) 90°

(b) 60°

(c) 45°

(d) 30°

(e) 120°

(f) 135°

(g) 150°

Figure 4.16: Cameraview over stationary payload with different basejoint angles

4.6 Open loop system response

Figure 4.17 shows the systems response without compensation.

The X-direction had an initial angle of approximately 60°. The unregulated system settles between ±5° after 70 seconds.

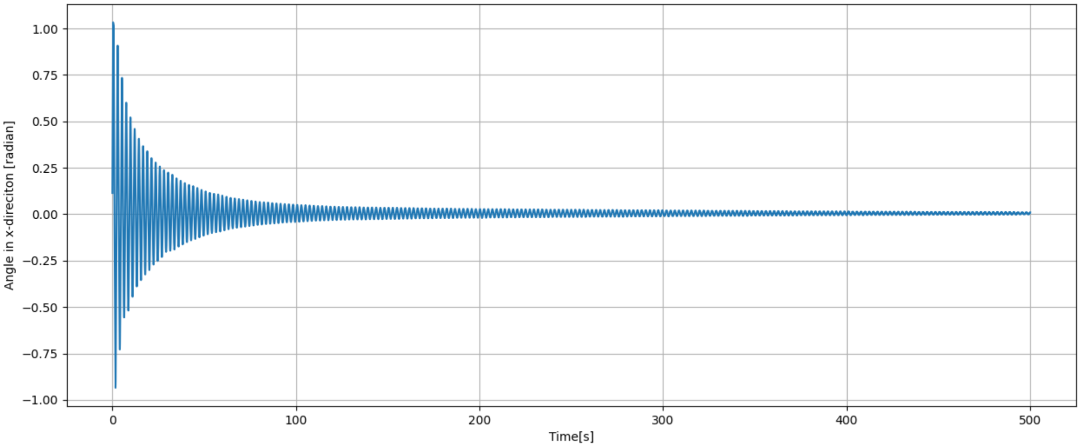


Figure 4.17: System response without compensation in x-direction

4.7 PID-Controller with position input.

This section presents results for the implemented PID Controllers with position input, as described in section 3.13.1.

4.7.1 First implementation of PID-controller

The result of the first implemented PID controller from section 3.13.1 is displayed in figure 4.18 and 4.19. Table 4.8 shows the PID's Parameters. The performance is measured through 11 trials. For all measurements the payload is launched from an angle of approximately −62.5° in both directions. The control loop starts when the payload crosses the x-axis. The controllers reference is -50 pixel in X-direction and -0.19m in Y-direction.

PID parameters	
PID controller:	K_p, K_i, K_d
X-direction PID	0.5, 0, 0.1
Y-direction PID	0.5, 0, 0.1

Table 4.8: PID parameters for x- and y-direction controllers

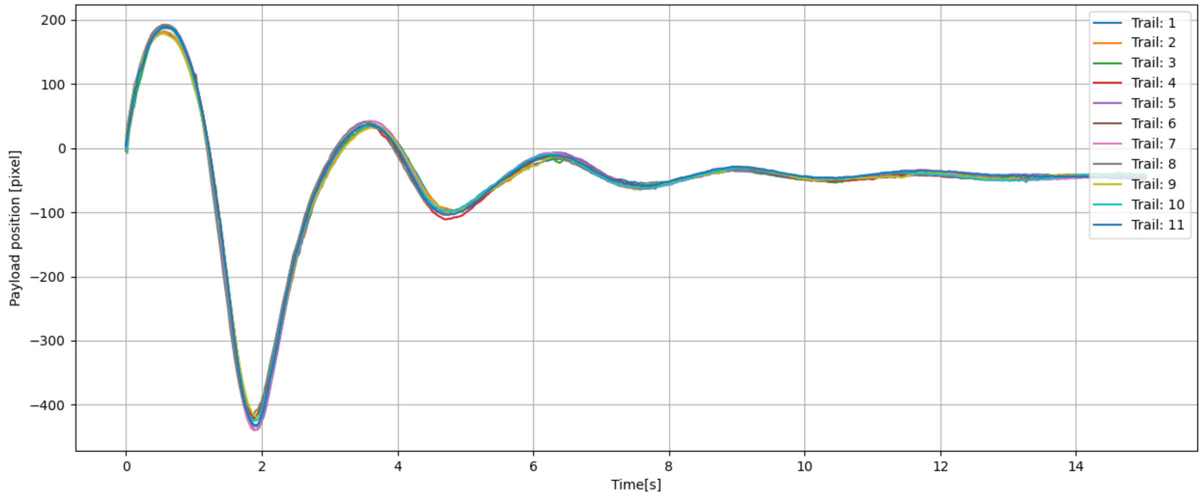


Figure 4.18: Plot of payload pixel coordinate in X-direction

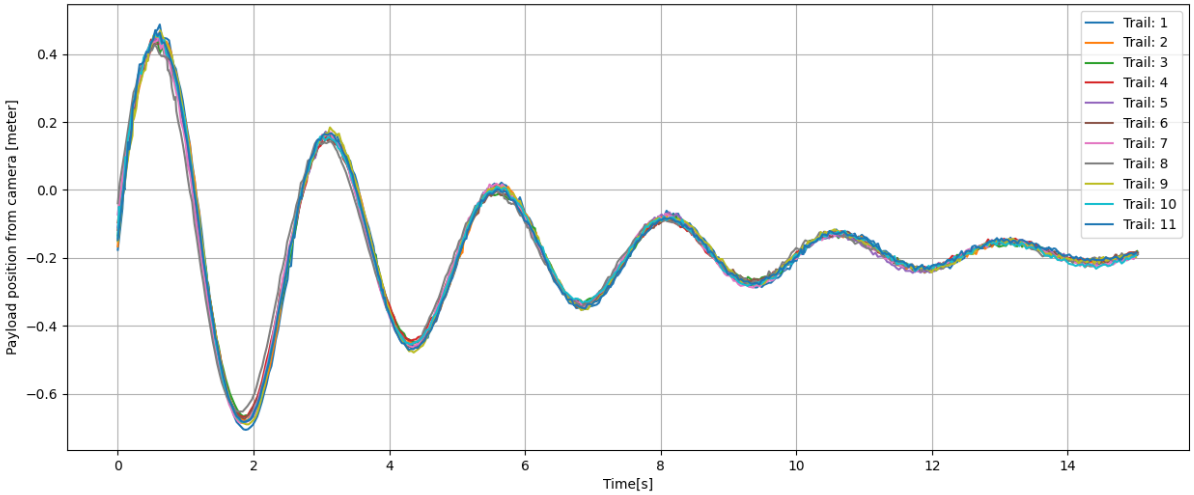


Figure 4.19: Plot of payload position from camera.

4.7.2 PID-controller with floor mounted camera

The results of the PID-controller with floor mounted camera from section 3.13.1 is displayed in figure 4.20 and 4.21. Table 4.9 shows the PID's Parameters. The performance is measured through 10 trials. For all measurements the payload is launched from an angle of approximately -68°in both directions. The control loop starts when the payload crosses 0.15m in the x-direction. The controllers reference is 0m in X-direction and -0.95m in Y-direction.

PID parameters	
PID controller:	K_p, K_i, K_d
X-direction PID	0.5, 0, 0.1
Y-direction PID	0.5, 0, 0.1

Table 4.9: PID parameters for x- and y-direction controllers

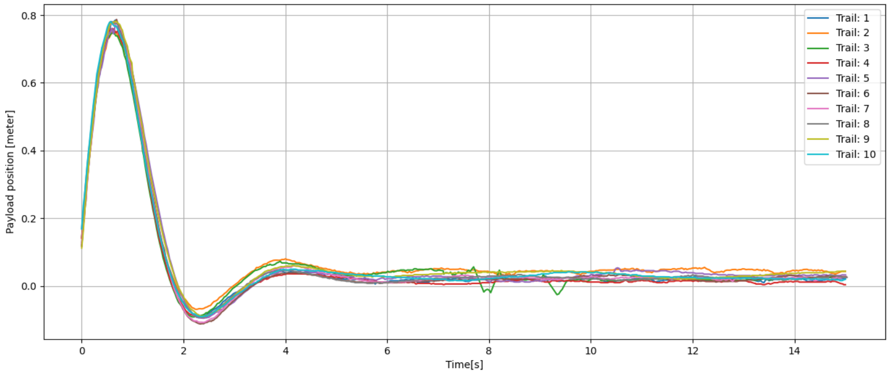


Figure 4.20: Plot of payload's global position in X-direction

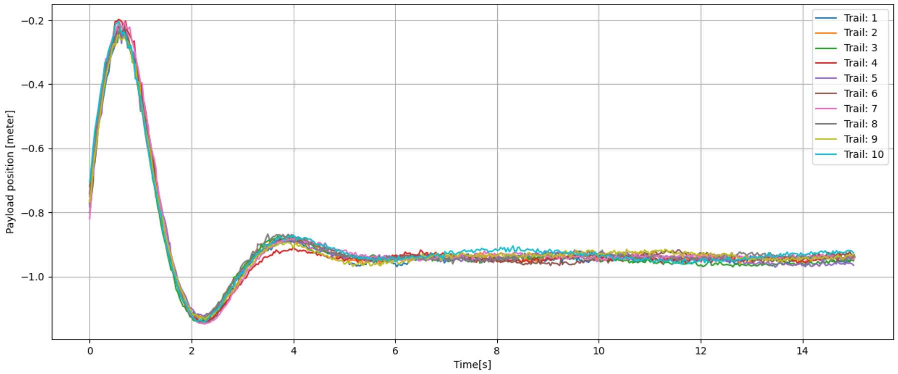


Figure 4.21: Plot of payload's global position in Y-direction

4.7.3 PID-controller with top mounted camera

The results of the PID-controller with top mounted camera from section 3.11.3 is displayed in figure 4.22 and 4.23. Table 4.10 shows the PID's Parameters. The performance is measured through 8 trials. For all measurements the payload is launched from an angle of approximately -64°in both directions. The control loop starts when the payload crosses 0.15m in the x-direction. The controllers reference is 0.2m in X-direction and -0.79m in Y-direction.

PID parameters	
PID controller:	K_p, K_i, K_d
X-direction PID	0.5, 0.006, 0.006
Y-direction PID	0.5, 0.006, 0.006

Table 4.10: PID parameters for x- and y-direction controllers

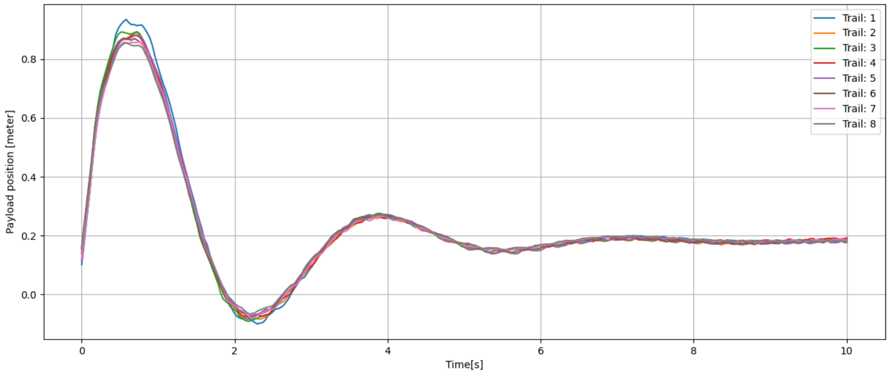


Figure 4.22: Plot of payload's global position in X-direction

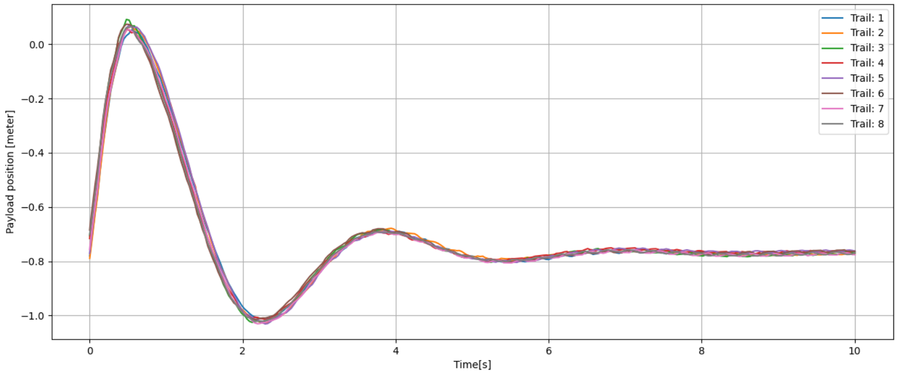


Figure 4.23: Plot of payload's global position in Y-direction

Figure 4.24 and 4.25 shows end effector and payload position in X and Y direction for one trail.

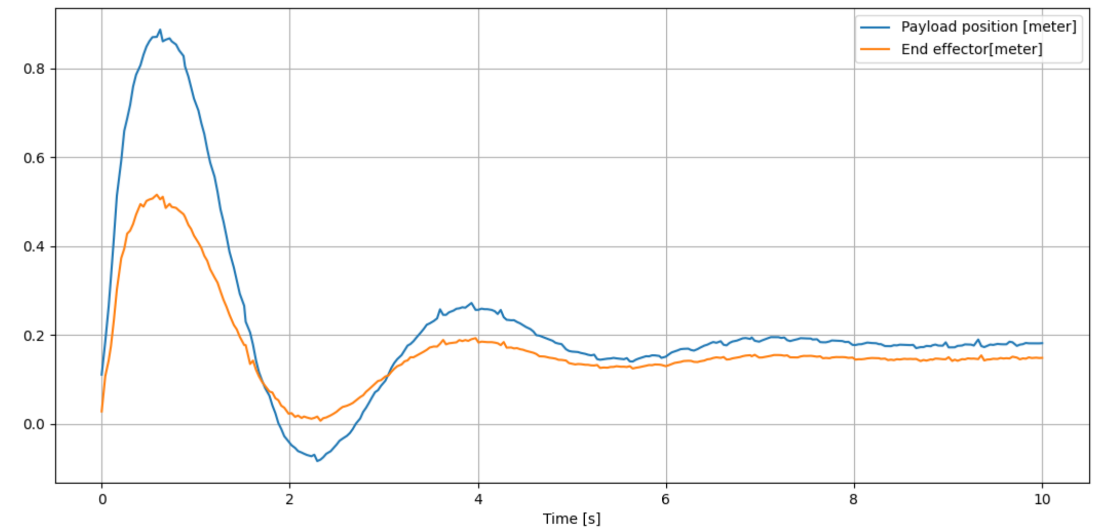


Figure 4.24: Payload and end effector position in x-direction

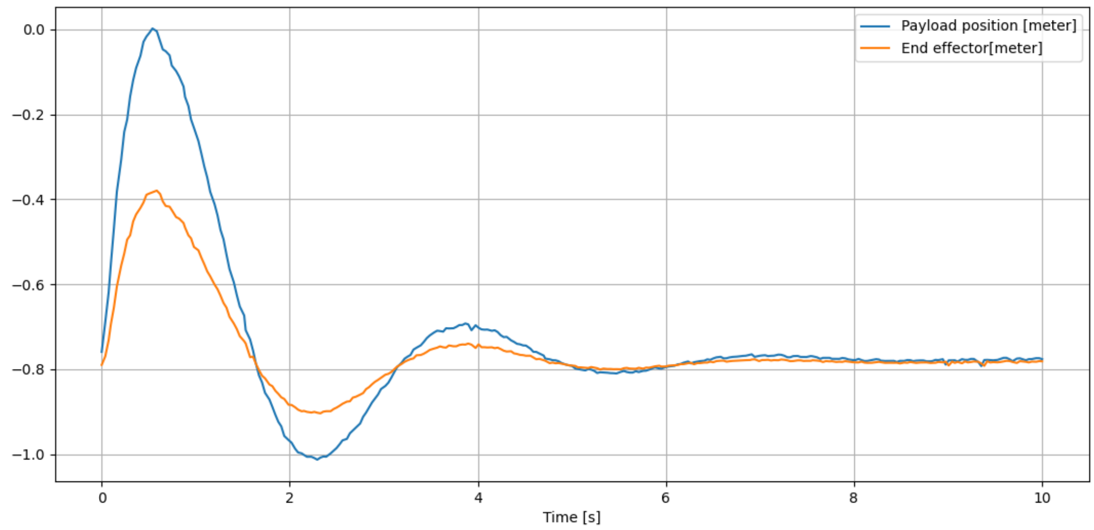


Figure 4.25: Payload and end effector position in y-direction

The video below demonstrates the entire system with a PID controller with position input and top mounted camera.
https://youtu.be/_qr3bth8z2k

Different PID-parameters

Figure 4.26 and 4.27 shows the systems response with different PID parameters. On the right corner of graphs are description over K_p, K_i and K_d values for each graph.The controllers reference is 0m in X-direction and -0.9m in Y-direction

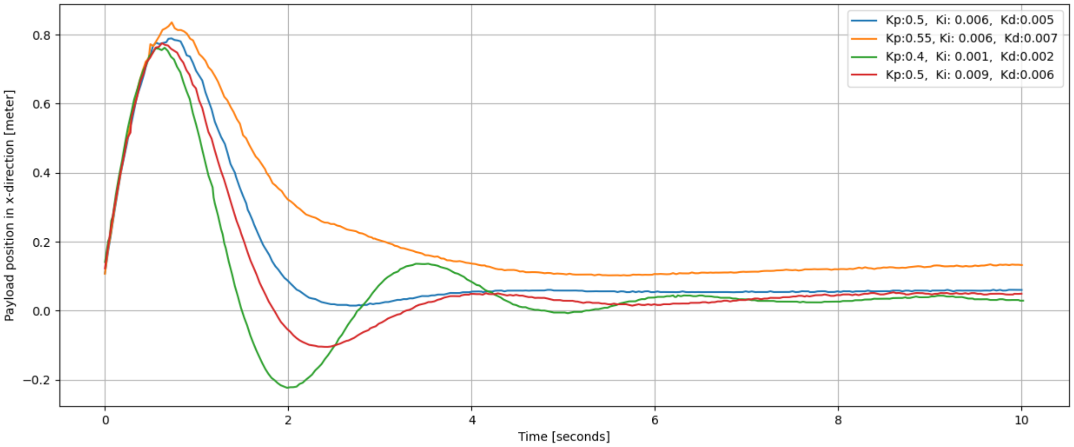


Figure 4.26: Plot of system response with different PID values in x-direction

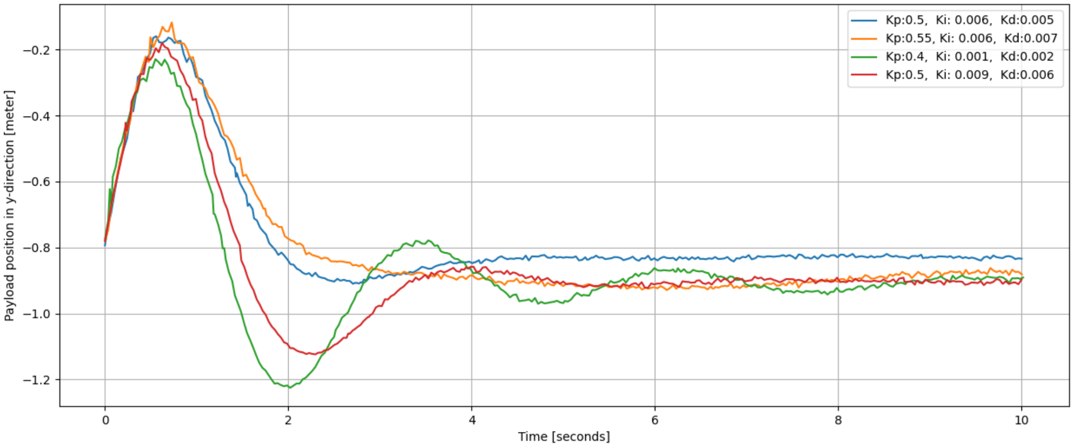


Figure 4.27: Plot of system response with different PID values in y-direction

4.8 PID-Controller with angle input

This section presents results for the implemented PID Controllers with angular input, as described in section 3.13.2. With the camera mounted on the end effector.

4.8.1 PID with restricted end effector movement

The result of the PID controller with restricted end effector movement from section 3.13.2 is displayed in figure 4.28 and 4.29. Table 4.11 shows the PID's Parameters. The performance is measured through 10 trials. For all measurements the payload is launched from an angle of approximately -68°in both directions. The control loop starts when the payload crosses the x-axis. The controllers reference is 0m in X-direction and -0.9m in Y-direction.

PID parameters	
PID controller:	K_p, K_i, K_d
X-angle PID	0.5, 0.006, 0.005
Y-angle PID	0.5, 0.006, 0.005

Table 4.11: PID parameters for x- and y-angle controllers

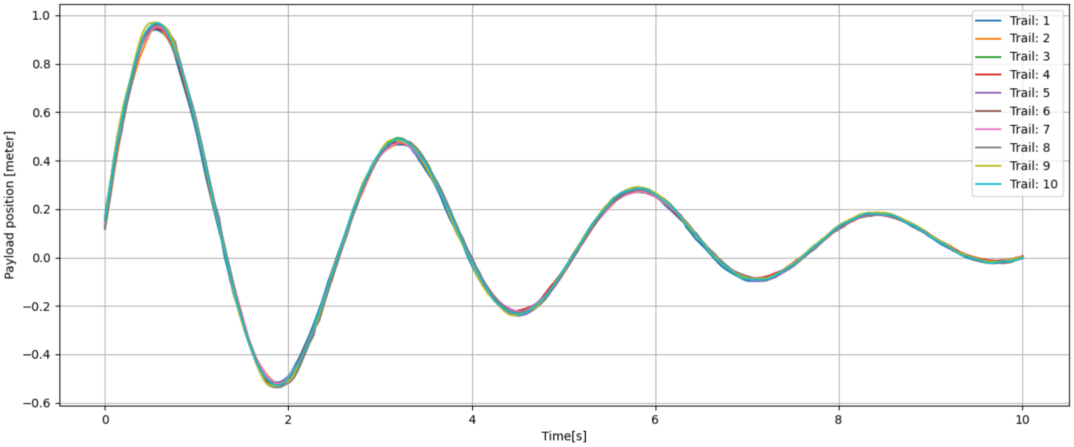


Figure 4.28: Plot of payload's position in X-direction

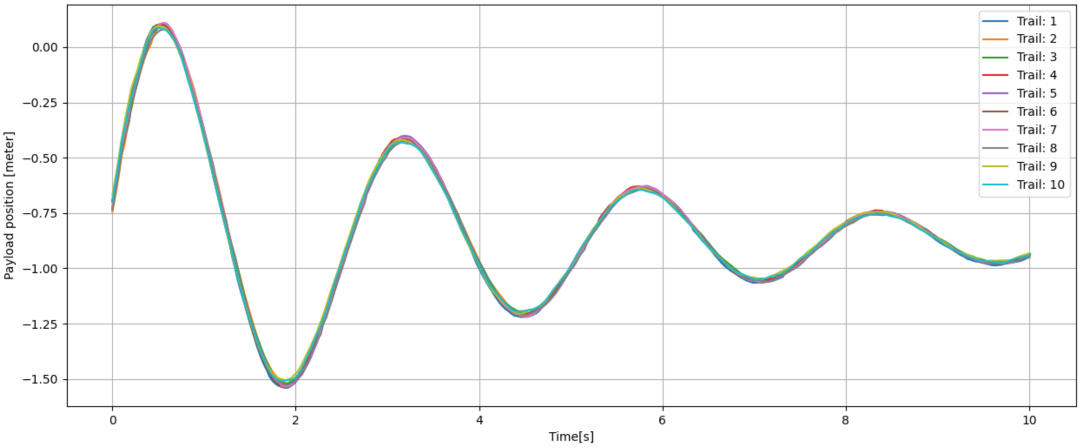


Figure 4.29: Plot of payload's position in Y-direction

Figure 4.30 and 4.31 shows end effector and payload position, along with the wire angle in X- and Y-direction for one trial.

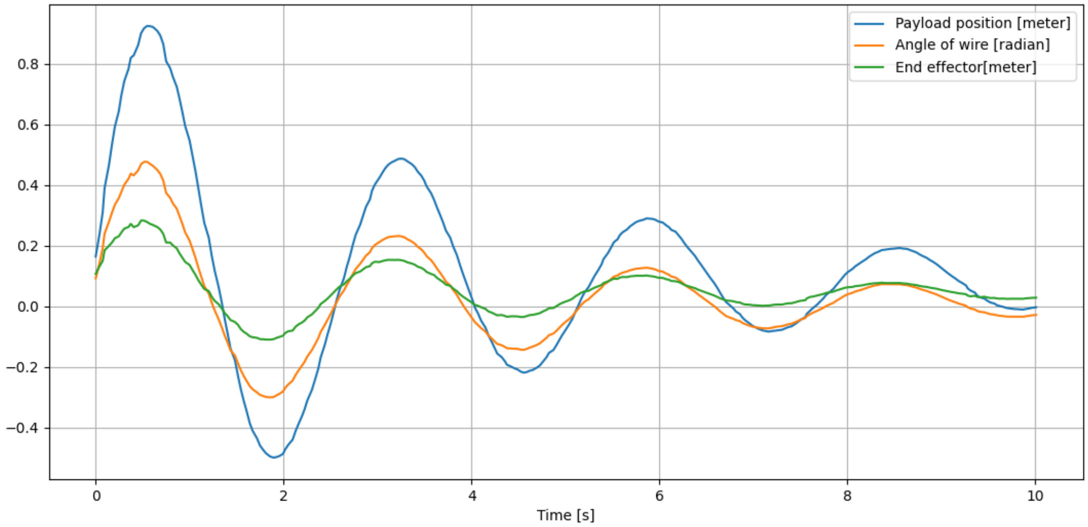


Figure 4.30: Plot of payload position, angle of wire and end effector position in x-direction.

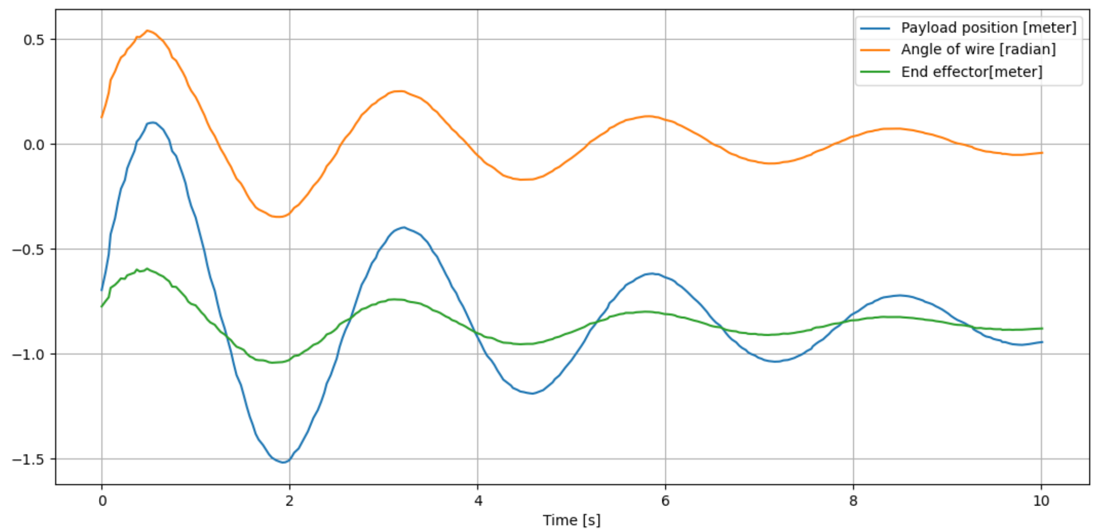


Figure 4.31: Plot of payload position, angle of wire and end effector position in y-direction.

The video below demonstrates the entire system with a PID controller with restricted end effector movement.
https://youtu.be/9GgXt6Ay2mw

4.8.2 PID with unrestricted end effector movement

The results of the PID-controller with unrestricted end effector movement and trajectory to reference position from section 3.13.2 is displayed in figure 4.32, 4.33 4.34 and 4.35. Table 4.12 shows the PID's Parameters. The performance is measured through 10 trials. For all measurements the payload is launched from an angle of approximately -70°in both directions. The control loop starts when the payload crosses 0 in the x-direction. The trajectory end position q_1 is 0m in X-direction and -0.85m in Y-direction. The scaling factor for the trajectory's travel time is 20.

PID parameters	
PID controller:	K_p, K_i, K_d
X-angle PID	0.5, 0.006, 0.005
Y-angle PID	0.5, 0.006, 0.005

Table 4.12: PID parameters for x- and y-angle controllers

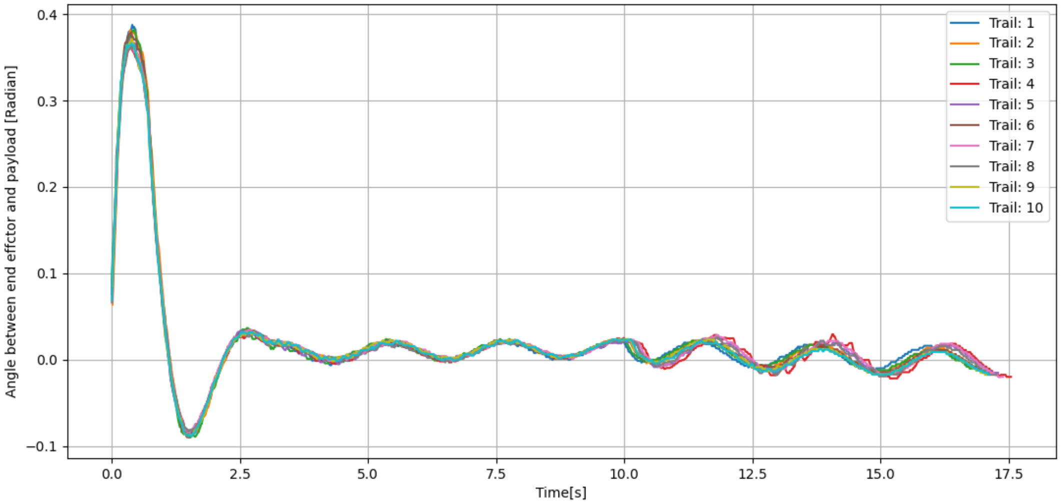


Figure 4.32: Plot of payloads angle in X-direction.

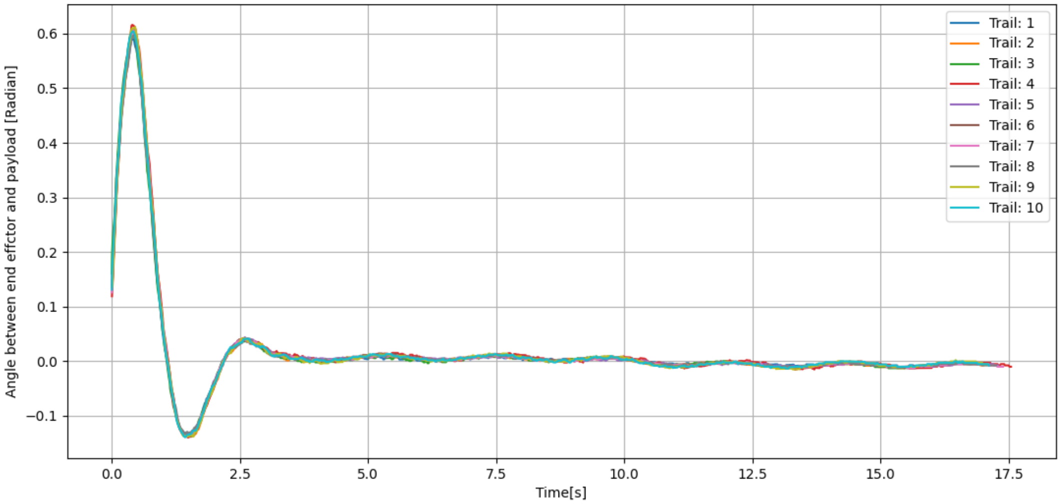


Figure 4.33: Plot of payloads angle in Y-direction.

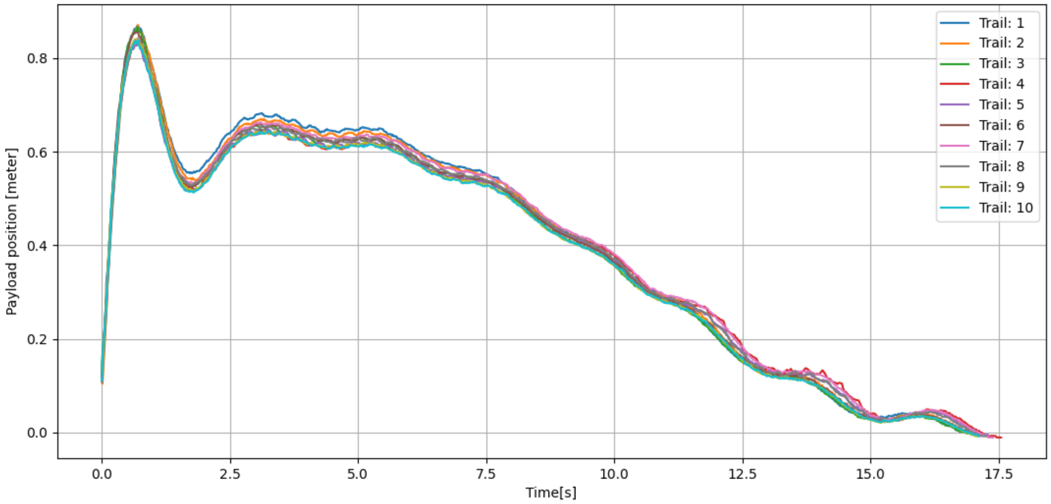


Figure 4.34: Plot of payloads position in X-direction.

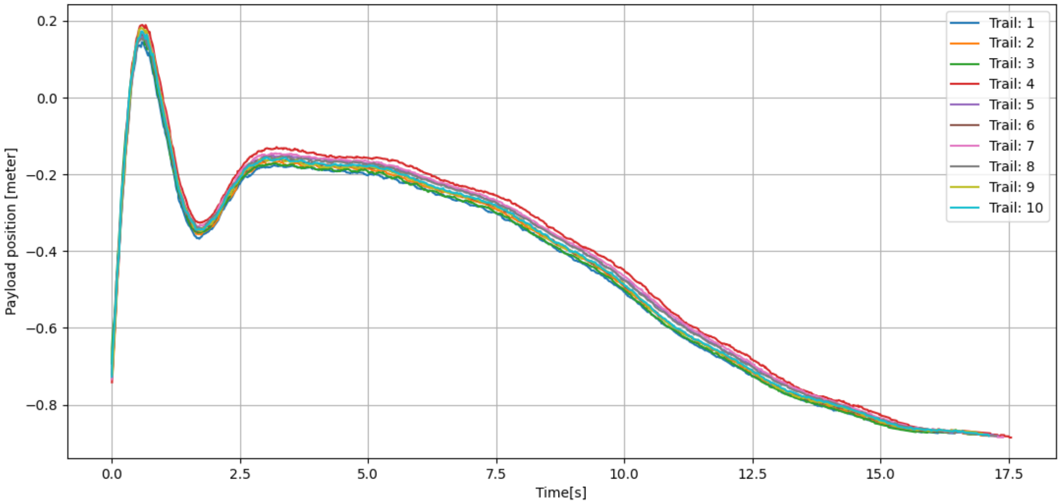


Figure 4.35: Plot of payloads position in Y-direction.

Figure 4.36 and 4.37 shows end effector and payload position, along with the wire angle in X- and Y-direction for one trial.

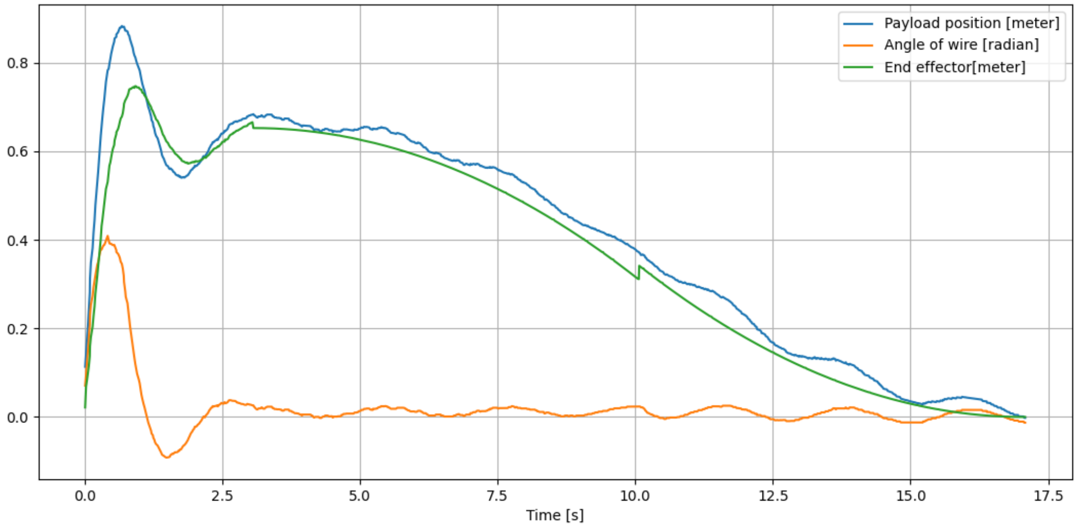


Figure 4.36: Plot of payload position, angle of wire and end effector position in x-direction.

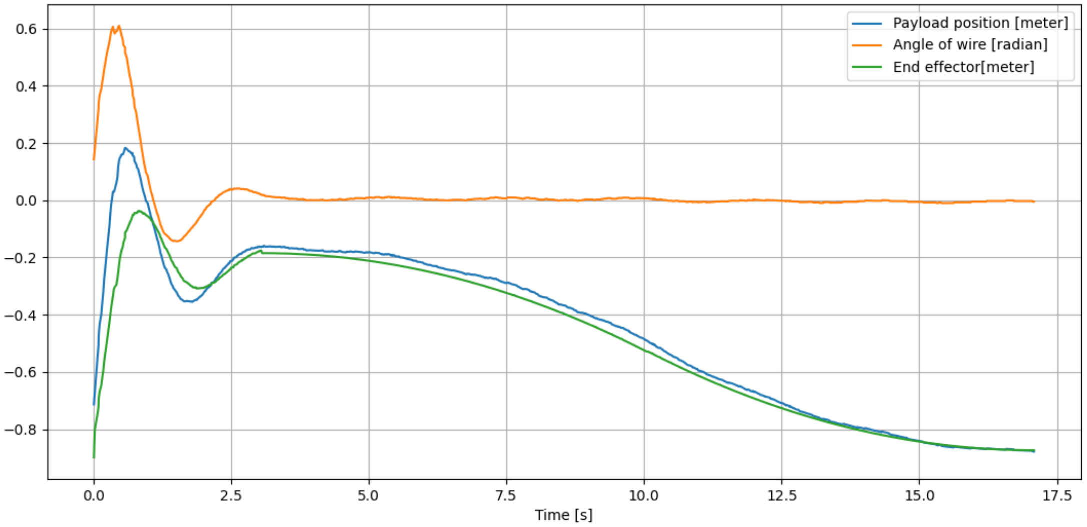


Figure 4.37: Plot of payload position, angle of wire and end effector position in y-direction.

The video below demonstrates the entire system with a PID controller with unrestricted end effector movement.
https://youtu.be/zYnpuX7JBEA

Chapter 5

Discussion

This chapter discuss the reasoning behind the obtained results and how we have experienced the project's execution.

5.1 Model

Simple pendulum

The simple pendulum model has it's limitations. The payload's centroid is aligned with the rope. Consequently, any rotation of the payload will not affect the pendulum dynamics. In the majority of instances, the actual payload of a crane will deviate from the model. The simple pendulum model doesn't account for the ropes mass, wind , friction and other disturbances. Therefore, the model has inaccuracies which don't reflect a real crane or our prototype.

The simple pendulum model is nonlinear. The disadvantage of linearization is that the model's accuracy is limited to small angles. However, for actual crane operations, the angle of the payload is generally low.

The simple pendulum model are only in a single plane, but the actual system exists in three-dimensional space. The method of decomposing the system into XZ and YZ plane proved effective for the simulation. Nevertheless, a spherical pendulum model would most likely be a better option [30].

Undamped model

Figure 4.2 shows the results of the undamped model. The outcome shows that the model deviates from the actual system. However, the frequency of the pendulum's swings show a close match to the real system. This suggests that the length of the rope is accurately estimated.

Damped model

Since the system is non-linearly there is not one unique solution for the damping ratio. The damping impact gradually decreases when the payload angle decreases. The result of the first and second system measurements in figure 4.3 and 4.4, indicate that a shorter data set yields a more precise damping ratio for that time interval. The damping ratio utilized for the project was derived from the first measurements, as they are more accurate for the angle range where compensation is most crucial. However, for a more accurate model, the damping ratio should be updated during the loop.

5.2 Kinematics

First Model

For certain solutions, the findings of the initial kinematic model revealed that the model was erroneous.

Figure 4.12a depicts a robot setup with all joint angles set to 0°. The end effector is described as lying straight and pointing 2 units along the x axis as a result of the forward kinematics. This exact solution is proven valid using inverse kinematics and the accompanying input from the forward kinematics.

Figure 4.12b, on the other hand, makes an intriguing point. The inverse kinematics output does not correspond to the forward kinematics input. Despite this, upon analysing the actual position of the end effector, the solution is correct, although unsuitable. The potential of many solutions is challenging when formulating inverse kinematics for multiple DOF.

In addition, figure 4.14b shows an inaccuracy. The inverse kinematics with the accompanying forward input does not yield a genuine accurate solution. This kinematic model was derived from the one given in paper [28]. However, we did not achieve the same outcomes. One cause for this may be that our model is incorrectly implemented. Perhaps due to the group's lack of robotics knowledge in general.

Another factor that might have a detrimental influence on the outcomes is the robot's actual setup. The basis coordinate system of the UR10 robot is different. In comparison to other robots.

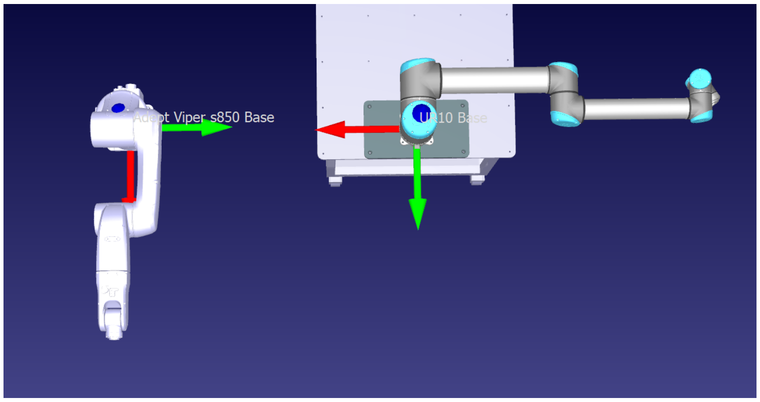


Figure 5.1: Viper s850 and UR10 with different base coordinate systems in RoboDK

The differing base coordinate systems for two robots, the Viper s850 and the UR10, are illustrated in figure 5.1. In zero configuration, the UR10 is pointed away from the positive x-axis. This is, however, a matter of convention. The model was based on literature with a different convention than our system. This is most likely why the model does not represent the real robotic arm accurately.

Second Model

In contrast to the previous model, the findings of the second kinematic model showed that the model was correct in all configurations.

This model was similar to the first, except it took into consideration the UR10's unique base coordinate system. The model produced results that were comparable to the literature [18], of which the inverse kinematics were based on. The simulated kinematic model gives correct solutions, as seen in figure 4.12. The correctness of the analytical computations for both forward and inverse kinematics is shown in figure 4.14.

The video in section 4.3 demonstrates that the simulated model delivers sufficient solutions in the arms workspace.

5.3 Choice of robot arm

For the prototype of the project, a UR10 was used. After completing the project, both its benefits and drawbacks have become evident.

The communication with the UR10 was straightforward to implement and delivered a swift and reliable transfer. The robot controller's PolyScope programming interface was simple to use and operate. There are numerous resources and materials available from UR robotics and those who have already worked with the robot. This was quite helpful when learning how to utilize the robot.

The UR10 is a 6DOF robot arm. Nevertheless, our system is 3DOF. Limiting the UR10 to only 3DOF movement proved challenging. This diminished the functionality of using the robot controller's built-in functionalities. To actuate and control the robot for our intended purpose, we were required to build separate functionalities.

The robot was mounted on a mobile table. The advantages of this is not being limited to one location when developing the prototype. However, not being permanently mounted also caused slight jerks in the tables position when actuating the robot. This is a source of uncertainty in the results for the floor mounted camera in section 4.7.2, since the camera doesn't move along with the table. When documenting results, we were careful to move the table to its original position between each trial. A solution to this issue would be to mount the robot on a permanent surface.

5.4 Graphical simulation

Webots was used to create the graphical simulator. During development, the simulation proved really useful. The simulations were quite accurate and closely mirrored the robot's motions. One drawback of the simulator was the lack of a good mechanism to limit the torque of individual actuators. As a result, no trajectory planning was used in the graphical simulator to depict the robot's motions. This eliminated the prospect of us simulating our designed trajectory planner.

5.5 Control loop simulation

The control simulations indicate that the controllers could function on the prototype. The simulations were slower than anticipated. Probably, because of the restricted working area. The end-effector position PID, negatively affects the performance from the Angle velocity PID controller, when they are combined. This is because, both the angle velocity PID and the end effector position PIDs outputs were nearly the opposite of each other. Figure 4.6b illustrated this. To effectively control the angular velocity of the payload and the location of the end effector. It was essential to optimize the relationship between each PID, as well as modify their parameters separately.

However, on a physical crane system, an fast controller isn't necessarily required. Because a real cranes performance is more limited by the speed of the actuators.

The results of the Modular PID control simulations in section 4.2.1, show a substantial improvement in anti-sway compensation. The payload is stationary after approximately 12 second. To enable the end effector to move more freely and compensate more effectively close to the reference, squaring the error was incorporated. By squaring the error, the controller weights higher gains significantly more than smaller gains. This effectively constrained the controllers impact on the end effector, reducing the allowed working area. With the squared error the end effector PID, the settling time improved to 8 seconds. In other words, improving the settling time with 4 seconds.

The LQR and PID modular controller was simulated and showed a settling time of 10 seconds. Even so, if the end effectors position, could be incorporated in the state space model. A weighted state of the end effectors position could probably provide a better results. This is also advantageous since it allows one to weight the end effector's effort relative to the actual system.

5.6 Controlling the end effector

Originally the simulated controllers were planned to be implemented on the prototype. However, controlling the robot with respect to acceleration, proved to challenging. The system is controlled by moving the position of the end effector. In order to move the end effector the robot arm needs a specific position in either joint angles or cartesian coordinates. Our model and simulations yields acceleration as the parameter to regulate the system. Therefore, there exits a need to express the models acceleration into a explicit position for the end effector. This problem is illustrated in figure 5.2.

Unfortunately, we were unable to accomplish this task. In retrospect a solution to this problem might be found by investigating the Jacobian to describe the relationship between cartesian velocities and the joint velocities[37]. There are probably several solutions to this problem, but

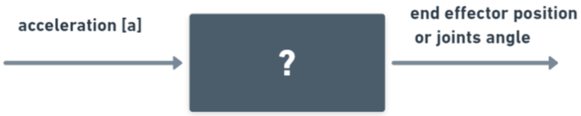


Figure 5.2: Illustration of the problem with using acceleration as input.

they would require more research and experience.

Trajectory with asymmetric constant acceleration

The controller gives a new output for each iterations. The controller cannot pause for trajectory planner's objective. Therefore, a trajectory planner is not an optimal for our control loop. We unsuccessfully attempted to derive the acceleration of the end effector for each iteration. In addition, calculating the problem numerically is computationally expensive. This would most likely significantly reduce the system's cycle time.

The results from section 4.4 show a implemented trajectory with asymmetric constant acceleration. The same technique was used in the PID with unrestricted end effector movement. The trajectory worked adequate for its intended use of simply moving the end effector towards the reference point. Although, the results show some jerk during the trajectory. A reason for this is because the trajectory planning is calculated for the X- and Y- direction independently.

Integrating acceleration

Integrating the acceleration was used in an attempt to move the end effector. However, The integration process moved the end effector discontinuous manner and had a tendency to drift the end effector. We suppose this is because some information is lost during numerical integration and the error is continually double integrated.

5.7 Machine vision

5.7.1 Camera

For the project, we attempted to employ two different cameras. The Intel RealSense d455 camera was used for the finalized version and the majority of the project's duration. The resolution and frame rate of the Intel camera proved adequate for detecting the payload. The Intel camera's documentation, examples, and capabilities expedited the project's machine vision implementation.

5.7.2 Machine vision algorithms

Color detection, ArUco marker and machine learning algorithms can all be used to detect the payload position.

- *The ArUco algorithm* had high precision, but the implementation was difficult when handling scaling, image distortion and rotation of the markers when estimate the payload position. For instance, if the payload rotates and the camera simultaneously detects two markers, it is difficult to determine the payload's center. Another disadvantage occurs when the camera is attached to the end effector. As illustrated in Figure 3.32, in this instance the payload's wire covers the marker, rendering the identification impossible. Using a greater number of tiny markers could be a potential solution to this issue.
- *The Machine learning algorithm Yolov5*, would require creating a large data set for training a model. It is computationally expensive and wasn't pursued further. However, for implementing on a real crane system, it can be practical.
- *The color detection algorithm* was straight forward to implement and provided accurate detection of the payload. Additionally, the algorithm has a low computational cost, allowing for fast cycle time and minimising the cameras bottleneck factor on the system. Therefore, the primary algorithm used with the prototype was the color detecting algorithm. A drawback with color detection is that the algorithm's performance is dependent on the surrounding environments lighting conditions. In our case, when developing the prototype , these can be modified, but not on a real crane. Therefore, it is mainly a stable prototype solution.

5.7.3 Floor mounted camera

Mounting the camera on the floor was easy to implement in the start of the project. Even so, utilizing the 3D reconstruction from the stereo/depth camera on the Intel RealSense introduces disturbance to the depth measurement. This is evident from figure 3.36. We circumvented this by incorporating a filter. Although, it is difficult to estimate how accurate the measurements will be for a more up scaled system, with different surroundings and lighting conditions.

The depth camera is only capable of measuring the distance to a payload's surface, but not it's center. Therefore, to estimate the payloads position in the camera frame, the payloads dimensions must be known, further incorporating a offset to the depth measurement.

The stationary camera view is also a limiting factor for the systems working area. Moving the payload outside of the camera's field of view is problematic. Additionally, the system is halted when individuals or objects pass in front of the camera. A solution to this is to incorporate more cameras to guarantee constant view of the payload.

The results in section 4.5.1, showing the accuracy of estimating global coordinates are dependent on several factors. The detection itself can also be a source of uncertainty. On a millimeter scale, object detection precision is difficult to assess. How we placed the payload in five different location can also impact the results. Again, it was difficult to achieve millimeter precision.

The transformation of global coordinates depends on the description of the camera's precise translation and rotation to its global position. In our case, it was difficult to determine the measured translation in millimeters. Additionally, for the camera mounted on the floor. When the robot actuates, the table it is installed on moves. This results in a tiny shift in the actual translation and rotation that we were unable to account for.

Table 4.4, shows the precision on the finalized payload coordinate estimation, for the prototype with the floor mounted camera. The highest error is 34 millimeters, whereas the majority of measurements have similar errors. Regardless, these results are till accurate enough for our system intended use.

5.7.4 Top Mounted camera

Mounting the camera on the end effector was challenging to implement on the prototype. In literature from similar systems and projects we saw no attempts to mount a camera on the end effector to detect the payload. Therefore, it was especially rewarding to explore and a achieve such a feat.

The top mounted camera requires no depth camera, because the distance to the payload is known from the wire length. It also enables a full working area for the system because, no obstacle can compromise the cameras field of view.

In addition, if this technique is used to a real crane, the camera will provide the operator with an excellent overview of the payload. The camera orientation compensation described in section 3.11.3 necessitated an additional DOF on the prototype. On a physical crane, orientation compensation would require a different solution.

With the camera mounted on the prototypes end effector, the lighting conditions for payload detection were more stable. A downside with this technique, is induced disturbances from the robots actuation. When the robot actuates and the cameras orientation is compensated, slight vibrations occur. These vibrations induce noise on the the cameras measurement. For instance, figure 4.24 shows a plot of a system response for X-direction where the camera is mounted on the end effector. The plot illustrates slight disturbances thought to be induced by vibrations from the end effectors movement.

The results in section 4.5.2, showing the accuracy of estimating global coordinates with the top mounted camera, are dependent on several factors.
Similarly, as with the floor mounted camera, the detection itself is a source of uncertainty.

The transformation of global coordinates depends on the description of the camera's precise translation and rotation to its global position. This was easier with the top mounted camera, because we could retrieve updated translation and rotation from the robot controller.

Furthermore, for the camera mounted on the end effector. The camera moves with the robot, casing no shift in the actual translation and rotation of the camera. However, due to the orientation compensation for the camera, as above mentioned in section 5.7.4. We did observe a slight impact on the cameras measurements.

How we precisely we placed the payload in different places also impact the results. Again, it was difficult to achieve millimeter precision.

Table 4.5 and 4.6, shows the accuracy of the finalized payload detection on the prototype with the top mounted camera. The largest error is 14 mm, with 848x480 resolution. Interestingly, changing the resolution provided no drastic improvement in the estimated coordinates. With both resolutions providing similar errors.
Table 4.7 shows additional results where the robots base rotates, when the payload is stationary positioned. In these results the largest error is 18.6 millimeters. This implies that there is some error associated with rotating the camera frame. However, these errors can result from the nature and distortion in the cameras lens. Because the error increases when measuring further away from the cameras center.

5.8 Control System

On our system, several PID controllers were implemented and tested. Originally the goal was to implement a model based controller. This proved challenging because the model generates gain through acceleration. We adopted a different strategy because we were unable to precisely move the robot end effector based on acceleration. We implemented PID controllers in order to utilize actual system measurements. The output of the PID was cartesian coordinates for the robot's end effector.

The PID's inability to handle constraints is a drawback. This enables the regulator to calculate large gains that can negatively impact the system's constraints. When the gain is outside of the robot's workspace, the system pauses momentarily. Nevertheless, if the gain results in a sudden shift for the robot arm, the arm's speed and safety constraints terminate the system.

Several trials for each controller were executed to document the results. The results demonstrate the persistence of the controller. Consequently, our controller and system are reliable. Increasing the credibility of our results further.

5.8.1 PID Controller with position input

The results in section 4.7.2 and 4.7.3 shows that the PID controller with position input, achieved a settling time around 6 seconds. The results are approximately identical with both top and floor mounted camera. Although, the results in figure 4.21 has more noise due to the cameras measurements. The controller compensate for the sway, whilst also achieving the payloads target reference position. This is possible because the prototype is fast enough to match the payloads moving pace. In practice, regulating with such high angle and speed, is unrealistic for a real crane.

5.8.2 PID Controller with angle input

The results in section 4.7.2 and 4.7.3 shows that the PID controller with angle input, achieved a settling time over 10 seconds. The controller is slow compared to the position PID, but eventually achieves the payloads target reference position. The controller was slower compared to others, because the controller only moves the arms end effector in a restricted area. The controller is suitable in situations where the working space is limited and the systems settling time isn't crucial. For real crane systems this might be more feasible.

In section 3.13.2 it is mentioned how the PIDs output is added to the end effector's position. Restricting the end effector's working area. Although this might not be an ideal implementation, it provides adequate results.

5.8.3 PID with unrestricted end effector movement

The PID with unrestricted end effector movement resulted in a settling time of around 3 seconds for the payloads sway. However, the payload's reference Cartesian position was achieved after a substantial delay, around 17.5 seconds. The payload's settling time is illustrated in figure 4.32 and 4.33, while the payload's reference position is illustrated in figure 4.36 and 4.37. If a physical crane has a limited working area, allowing the end effector to move freely may not be practical. Another thing to note, is that the target position is traced with a planned trajectory. This moves the end effector smoothly towards the reference position regardless of the angle of the payload. This can induce small oscillation to the payload. This is illustrated in figure 4.32 and 4.36. The plot of end effector position in figure 4.36 reveals that when the trajectory planner starts, the robot makes a small jerk at the start and halfway along the path. Figure 4.36 and 4.37 indicates that the jerk is higher in the x-direction than the y-direction.

The PID with unrestricted end effector movement and trajectory to the reference position is the fastest method to regulate the system with respect to the payloads angle. However, for higher payload angles, the end effector could move further away from the reference position. The advantages of this controller is depended on a cranes actuation speed and it having a large working area.

5.8.4 Tuning of PID parameters

Each PID controller utilized in the project is tuned through trial and error. Initially, we intended to utilize Ziegler and Nichols closed loop method to find optimal parameters [46]. However, this proved difficult to execute on the prototype. Ziegler and Nichols method requires the proportional gain set to make the system critically stable [32]. For our prototype which is a stable system by nature this was hard to induce. Even so, we attempted to make the system critically stable. But, increasing the gain eventually made the arm move so abruptly that the inbuilt safety and speed limitations interrupts the system.

Chapter 6

Conclusions and further work

The primary purpose of this project was to develop a prototype to compensate pendulum motion of payload on a maritime crane. A mathematical model of the swaying motion was made, using a simple pendulum model. With a purpose of providing insight in order to simulate and control the payloads oscillations. A robot arm was utilized to emulate actual crane motion, and to make a prototype to implement anti sway regulation. Machine vision is used to detect the payload. A graphical simulation environment is used for verification and development during the project.

In certain intervals, the mathematical model provides a near approximation of the real system. The model is employed when simulating control loops. However, the prototype's implemented controllers are not based on the simulations. As input, the model and simulations utilize acceleration. Unfortunately we were unable to use acceleration to control the prototype. The 3DOF kinematic model and corresponding equations were adequate for actuating the prototype and simulating crane motion. The machine vision yields adequate and consistent results. In the finalized prototype, the known payload is detected via color detection. Mounting the camera on the end effector proved suitable for its intended use. The project's graphical simulation environment is solely used to simulate the motion of the prototype, not the motion of a real system or pendulum motion. The prototype contains three distinct PID control loops. One with input for position, another with input for angle, and a third with unlimited end effector movement and trajectory to the ultimate reference. With position input, the system is both quick and precise. With angle input, the system becomes more constrained and hence oscillates more. The control loop with unrestricted end effect movement and trajectory to reference, is the quickest at reducing swaying motion. It requires the largest operating area and the most time to reach the desired position of the payload.

Considering the project requirements outlined in the preliminary report, along with the problem formulation in section 1.2. It is evident that the thesis addresses the projects main focus areas. Initially, it was intended to employ a digital triplet consisting of a simulated prototype, a virtual crane, and a physical prototype. Due to underestimated development time, we abandoned the digital triplet. Instead, focusing on developing the prototype. We hoped to do considerably more with this project than we managed. However, all things considered we are greatly satisfied with the implemented components, the working prototype and the learning outcomes we achieved. Having developed our own prototype and endeavored to contribute to the academic community, we see the value of reusing and further developing the work of others.

6.1 Further work

Below are suggestions to further works and improvements for the project.

- Manage to control the robots end effector with respect to acceleration from the model.
- The prototype can be placed on a 3DOF or Stewart platform to see how the prototype handles payload compensation when accounting sea waves.
- The system should be subjected to disturbances in order to further valuate the response. For example, impacting the payload with wind, to see how the system performs.
- Other controllers such as LQR and Model Predictive Control, should be implemented to compare different controllers system response and handle constraints on the system.
- Constrained velocity and acceleration control should be implemented in order for the prototype to better compare to a real crane.
- The graphical simulator should be improved and implemented as a digital twin. This allows users to better simulate solutions without the need of the physical prototype.
- The prototype color detection is limited to ideal environments with manageable lighting conditions. Further research into the performance during realistic conditions could be beneficial.
- Machine learning can be pursued further as a means for object detection. Incorporating a hybrid system combining two or more detection algorithms could be useful.
- The trajectory planner should be improved to reduce the occasional jerking from the robots motion. A multi dimensional trajectory should be explored.

Bibliography

[1] Online ArUco markers generator. URL https://chev.me/arucogen/. [Online; accessed April 05 , 2022].

[2] Opc classic, . URL https://opcfoundation.org/about/opc-technologies/opc-classic/. [Online; accessed April 20 , 2022].

[3] Unified architecture, . URL https://opcfoundation.org/about/opc-technologies/opc-ua/. [Online; accessed April 20 , 2022].

[4] PID theory explained. URL https://www.ni.com/en-no/innovations/white-papers/06/pid-theory-explained.html. [Online; accessed May 10 , 2022].

[5] Polyscope manual. URL https://s3-eu-west-1.amazonaws.com/ur-support-site/44018/Software_Manual_en_Global.pdf. [Online; accessed May 8 , 2022].

[6] Introducing the intel RealSense depth camera d455. URL https://www.intelrealsense.com/depth-camera-d455/. [Online; accessed April 1 , 2022].

[7] opcua-asyncio. URL https://github.com/FreeOpcUa/opcua-asyncio/blob/259811d5ad11c47c8d2b2289a8a2783b97c79421/examples/server-minimal.py. [Online; accessed April 29 , 2022].

[8] OpenCV: Thresholding operations using inRange. URL https://docs.opencv.org/3.4/da/d97/tutorial_threshold_inRange.html. [Online; accessed April 8 , 2022].

[9] Getting started with fusion 360, product documentation. URL https://help.autodesk.com/view/fusion360/ENU/?guid=GUID-1C665B4D-7BF7-4FDF-98B0-AA7EE12B5AC2. [Online; accessed April 6 , 2022].

[10] Picture of intel realsense depth camera d455. URL https://www.nidarosdata.no/intel-realsense-depth-camera-d455/cat-p/c/p1001384186?gclid=Cj0KCQjwmPSSBhCNARIsAH3cYgah5H-NJDX41IFL6oQUGDoVH2oiu8jOaiOaqB2Tg-OvrkzUhCaX1fUaAvB9EALw_wcB. [Online; accessed March 18 , 2022].

[11] Elp-usb-camera picture. URL http://www.elpcctv.com/elp-usb-camera-free-driver-5mp-ov5640-fisheye-wide-angle-usb-camera-module-elpusb500w02ml170-p-53.html. [Online; accessed april 19 , 2022].

[12] Viper s850 documentation. URL https://industrial.omron.eu/en/products/viper. [Online; accessed April 10 , 2022].

[13] Comparison between standard and modified denavit-hartenberg methods in robotics modelling. doi: 10.11159/icmie16.118. URL http://avestia.com/MCM2016_Proceedings/files/paper/ICMIE/118.pdf. [Online; accessed May 13 , 2022].

[14] YOLOv5 documentation, . URL https://docs.ultralytics.com/. [Online; accessed may 7 , 2022].

[15] ultralytics/yolov5 example code, . URL https://github.com/ultralytics/yolov5. [Online; accessed April 20 , 2022].

[16] What is opc?, definition, Jun 2017. URL https://opcfoundation.org/about/what-is-opc/. [Online; accessed April 20 , 2022].

[17] Picture of and information of saywer robot, Sep 2018. URL https://www.orix.co.jp/grp/en/newsrelease/180905_ORIXG2.html. [Online; accessed April 6 , 2022].

[18] Adnan Rafi Al Tahtawi, Muhammad Agni, and Trisiani Dewi Hendrawati. Small-scale robot arm design with pick and place mission based on inverse kinematics. *Journal of Robotics and Control (JRC)*, 2(6), Nov 2021. doi: 10.18196/jrc.26124.

[19] Dr. Kostas Alexis. Lqr control. URL http://www.kostasalexis.com/lqr-control.html. [Online; accessed May 18 , 2022].

[20] Aqeel Anwar. What are intrinsic and extrinsic camera parameters in computer vision? URL https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec [Online; accessed February 03 , 2022].

[21] Universal Robots AS. The urscript programming language. URL https://www.siemens-pro.ru/docs/ur/scriptManual.pdf. [Online; accessed February 16 , 2022].

[22] Luigi Biagiotti and Claudio Melchiorri. *Trajectory Planning for automatic Machines and Robots*. Springer, 2008. ISBN 978-3-540-85628-3 978-3-540-85629-0.

[23] Ilja Boginskis and Hmdoun Abker Ibrahim Hmdoun. Modeling, Simulation and Control for Marine Crane Operations. page 8.

[24] John J Craig. *Mechanics and Control Third Edition*. Pearson, inc, Upper Saddle River, NJ, 2005. ISBN 0-13-123629-6.

[25] Cyberbotics. Simulate robot applications. URL https://cyberbotics.com/. [Online; accessed April 3 , 2022].

[26] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, 22(2), 1955. doi: 10.1115/1.4011045.

[27] Didrik Fjeld Elset. *Crane Payload Stabilization using Lagrangian Kinematics and Euler Angles*. NTNU, 2019.

[28] Madiha Farman, Muneera Al-Shaibah, Zoha Aoraiath, and Firas Jarrar. Design of a Three Degrees of Freedom Robotic Arm. *International Journal of Computer Applications*, 179 (37):12–17, April 2018. ISSN 09758887. doi: 10.5120/ijca2018916848. URL http://www.ijcaonline.org/archives/volume179/number37/farman-2018-ijca-916848.pdf.

[29] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. 47(6):2280–2292. ISSN 00313203. doi: 10.1016/j.patcog.2014.01.005. URL https://linkinghub.elsevier.com/retrieve/pii/S0031320314000235. [Online; accessed April 22 , 2022].

[30] J Gea-Banacloche. *University Physics I: Classical Mechanics*. Open Educational Resources, 2019. URL https://scholarworks.uark.edu/oer/3. [Online; accessed April 17 , 2022].

[31] Mario E.Salgando Graham C. Goodwin, Stefan F.Graebe. *Control system design*. Prentice Hall, 2002. ISBN 0-13-958653-9.

[32] Finn Haugen. *Praktisk reguleringsteknikk*. Tapir akademisk forl, Trondheim, 2. utg. edition, 2003. ISBN 8251918871.

[33] Ville Hirvonen. Lagrangian mechanics. URL https://profoundphysics.com/lagrangian-mechanics-for-beginners/. [Online; accessed May 18 , 2022].

[34] D. J. Inman. *Vibration with Control*. John Wiley Sons, 2006. ISBN 0-470-01051-7.

[35] Inbae Jeong. Ur10 model in webots example files. URL https://github.com/ibjeong/webots_ur. [Online; accessed April 19 , 2022].

[36] Jia Luo and C. Edward Lan. Determination of weighting matrices of a linear quadratic regulator. 18(6):1462–1463. ISSN 0731-5090, 1533-3884. doi: 10.2514/3.21569. URL https://arc.aiaa.org/doi/10.2514/3.21569. [Online; accessed May 05 , 2022].

[37] Kevin M. Lynch and Frank C. Park. *Modern robotics: mechanics, planning, and control*. Cambridge University Press, Cambridge, UK, 2017. ISBN 978-1-107-15630-2 978-1-316-60984-2. OCLC: ocn983881868.

[38] David Morin. *Introduction to Classical Mechanics With Problems and Solutions*. Cambridge University Press, 2008. ISBN 978-0-521-87622-3.

[39] RoboDK. Robodk software documentation. URL https://robodk.com/. [Online; accessed February 03 , 2022].

[40] RobotWorx. Ur10 robotic arm, 2022. URL https://www.robots.com/robots/universal-robots-ur10. [Online; accessed February 03 , 2022].

[41] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. 30(1):32–46. ISSN 0734-189X. doi: 10.1016/0734-189X(85)90016-7. URL https://www.sciencedirect.com/science/article/pii/0734189X85900167. [Online; accessed April 11 , 2022].

[42] Unity Technologies. Unity - manual: Creating and using scripts. URL https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html. [Online; accessed 2022-05-06].

[43] Universial robots. Real-time data exchange (rtde) guide, 2019. URL https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/. [Online; accessed Mars 28 , 2022].

[44] Lisa Ann Williams. Modelling, Simulation and Control of offshore crane. pages 0–103, 2018.

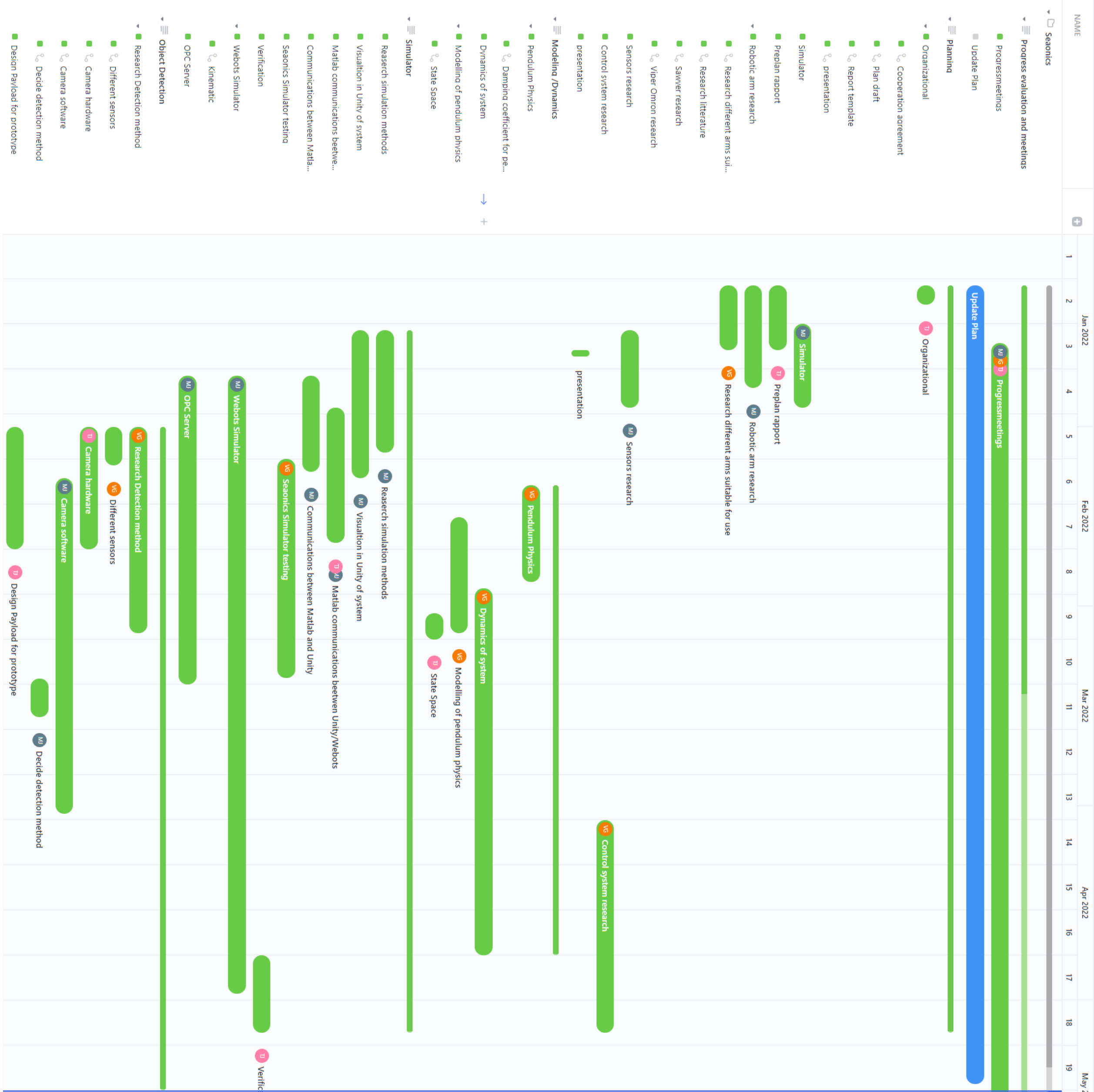
[45] Jana Kosecka S. Shankar Sastry Yi Ma, Stefano Soatto. *An invitation to 3-D Vision*. Springer Science + buisness meida, LLC, 2004. ISBN 978-0-387-21779-6.

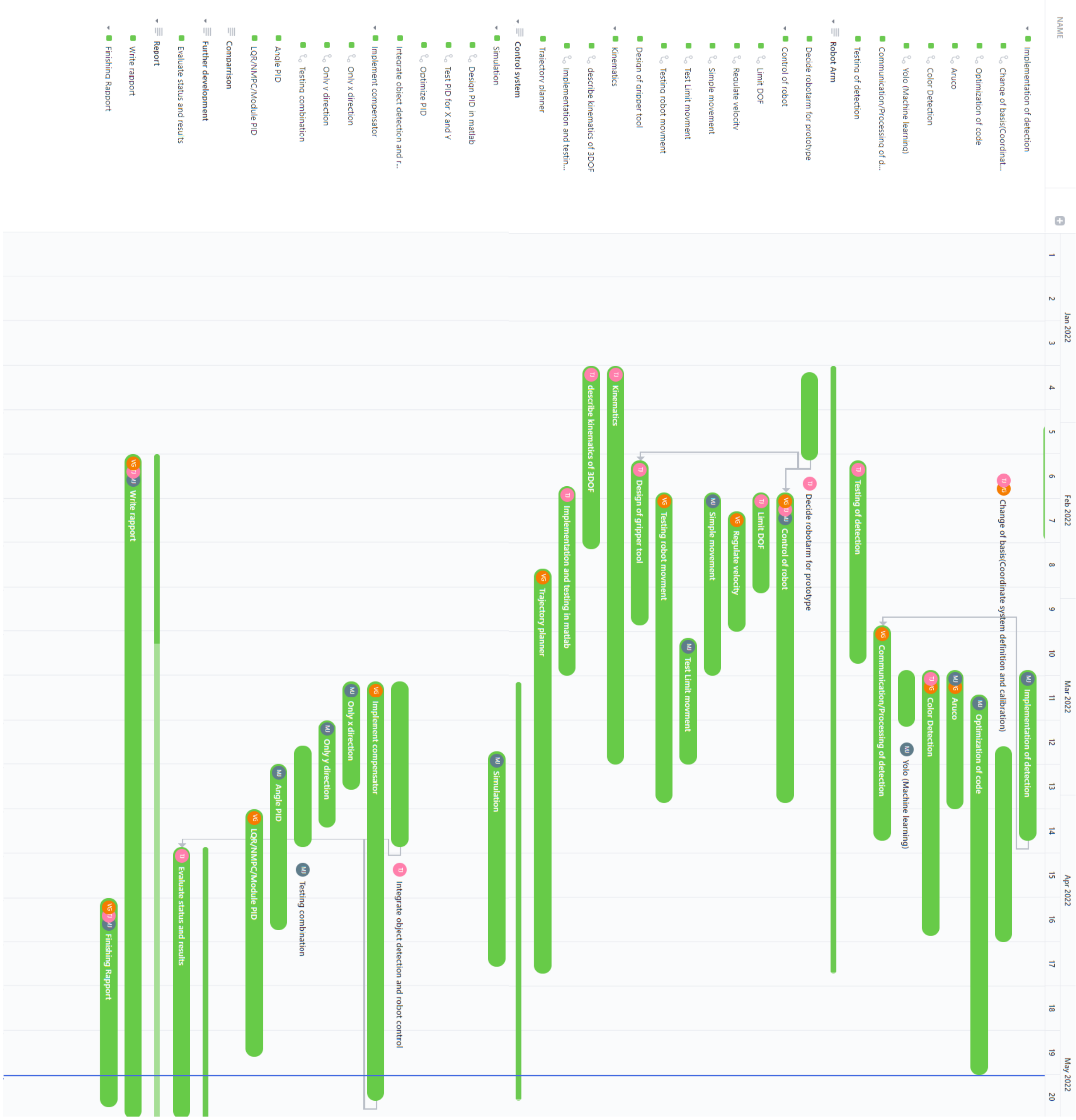
[46] J. G Ziegler and N. B Nichols. Optimum settings for automatic controllers. *Journal of dynamic systems, measurement, and control*, 115(2 B):220–222, 1993. ISSN 0022-0434.

Appendix A

Appendices

A.1 Gantt diagram



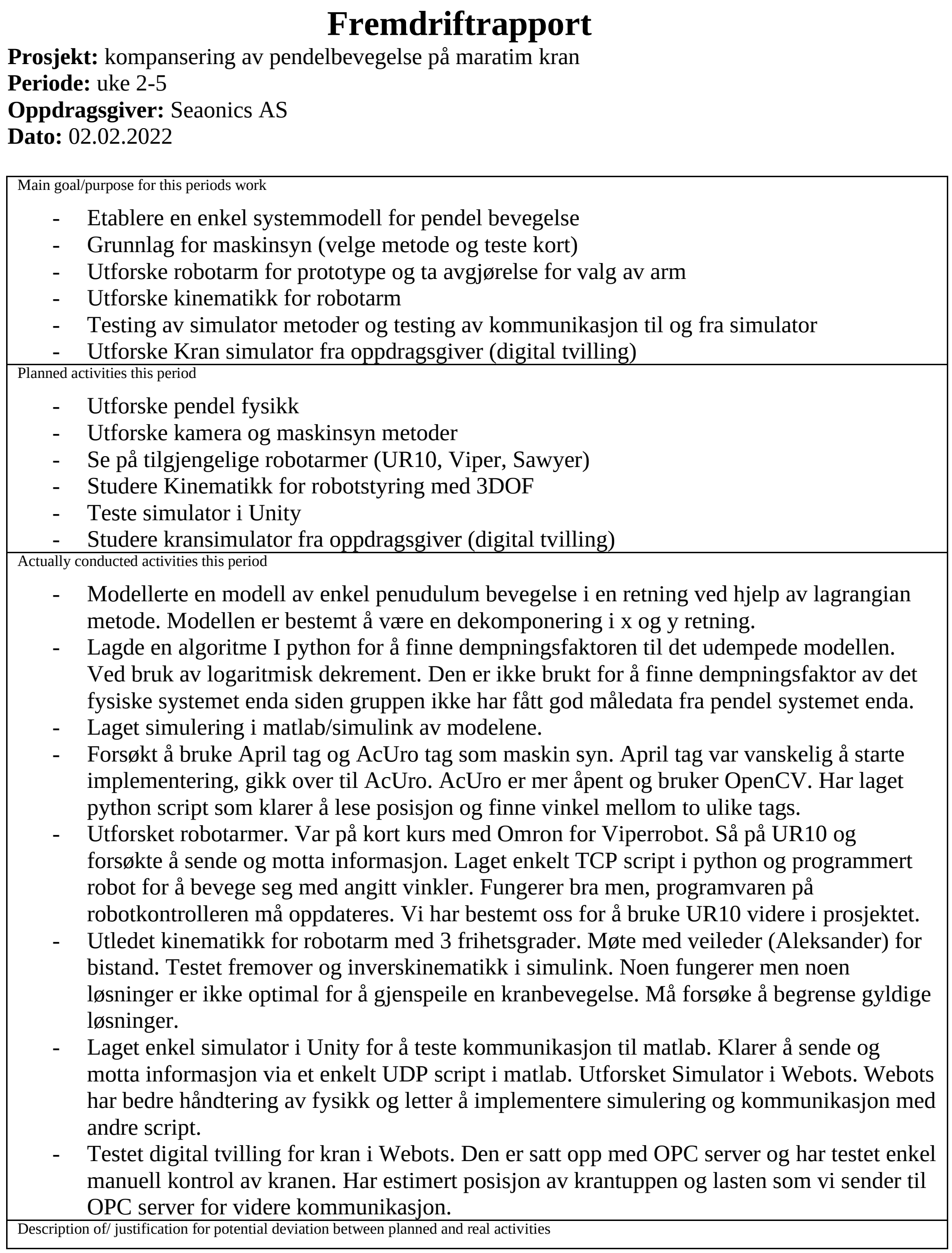

NAME
Implementation of detection
Change of basis(Coordinat...
Optimization of code
Aruco
Color Detection
Yolo (Machine learning)
Communication/Processing of d...
Testing of detection
Robot Arm
Decide robotarm for prototype
Control of robot
Limit DOF
Regulate velocity
Simple movement
Test Limit movment
Testing robot movment
Design of gripper tool
Kinematics
describe kinematics of 3DOF
Implementation and testin...
Trajectory planner
Control system
Simulation
Design PID in matlab
Test PID for X and Y
Optimize PID
Integrate object detection and r...
Implement compensator
Only x direction
Only y direction
Testing combination
Angle PID
LQR/NMPC/Module PID
Comparrison
Further development
Evaluate status and results
Report
Write rapport
Finishing Rapport
Jan 2022
Feb 2022
Mar 2022
Apr 2022
May 2022
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Change of basis(Coordinate system definition and calibration)
Integrate object detection and robot control
Implementation and testing in matlab
TJ
VG
MJ

A.2 Source code

Uploaded as zip file on Inspera. Source code contains:

- Scripts for the implemented PID controllers
- Scripts to log result for each controller
- Polyscope UR10 script
- Data base of measurements
- Script to plot results
- Matlab simulation for kinematics
- Script to estimate damping ratio
- Webots simulator

A.3 Progress report



Fremdriftrapport

Prosjekt: kompansering av pendelbevegelse på maratim kran
Periode: uke 2-5
Oppdragsgiver: Seaonics AS
Dato: 02.02.2022

Main goal/purpose for this periods work

- Etablere en enkel systemmodell for pendel bevegelse
- Grunnlag for maskinsyn (velge metode og teste kort)
- Utforske robotarm for prototype og ta avgjørelse for valg av arm
- Utforske kinematikk for robotarm
- Testing av simulator metoder og testing av kommunikasjon til og fra simulator
- Utforske Kran simulator fra oppdragsgiver (digital tvilling)

Planned activities this period

- Utforske pendel fysikk
- Utforske kamera og maskinsyn metoder
- Se på tilgjengelige robotarmer (UR10, Viper, Sawyer)
- Studere Kinematikk for robotstyring med 3DOF
- Teste simulator i Unity
- Studere kransimulator fra oppdragsgiver (digital tvilling)

Actually conducted activities this period

- Modellerte en modell av enkel penudulum bevegelse i en retning ved hjelp av lagrangian metode. Modellen er bestemt å være en dekomponering i x og y retning.
- Lagde en algoritme I python for å finne dempningsfaktoren til det udempede modellen. Ved bruk av logaritmisk dekrement. Den er ikke brukt for å finne dempningsfaktor av det fysiske systemet enda siden gruppen ikke har fått god måledata fra pendel systemet enda.
- Laget simulering i matlab/simulink av modelene.
- Forsøkt å bruke April tag og AcUro tag som maskin syn. April tag var vanskelig å starte implementering, gikk over til AcUro. AcUro er mer åpent og bruker OpenCV. Har laget python script som klarer å lese posisjon og finne vinkel mellom to ulike tags.
- Utforsket robotarmer. Var på kort kurs med Omron for Viperrobot. Så på UR10 og forsøkte å sende og motta informasjon. Laget enkelt TCP script i python og programmert robot for å bevege seg med angitt vinkler. Fungerer bra men, programvaren på robotkontrolleren må oppdateres. Vi har bestemt oss for å bruke UR10 videre i prosjektet.
- Utledet kinematikk for robotarm med 3 frihetsgrader. Møte med veileder (Aleksander) for bistand. Testet fremover og inverskinematikk i simulink. Noen fungerer men noen løsninger er ikke optimal for å gjenspeile en kranbevegelse. Må forsøke å begrense gyldige løsninger.
- Laget enkel simulator i Unity for å teste kommunikasjon til matlab. Klarer å sende og motta informasjon via et enkelt UDP script i matlab. Utforsket Simulator i Webots. Webots har bedre håndtering av fysikk og letter å implementere simulering og kommunikasjon med andre script.
- Testet digital tvilling for kran i Webots. Den er satt opp med OPC server og har testet enkel manuell kontrol av kranen. Har estimert posisjon av krantuppen og lasten som vi sender til OPC server for videre kommunikasjon.

Description of/ justification for potential deviation between planned and real activities

- En algoritme for å finne dempningsfaktoren var ikke planlagt, men noe gruppen så som nødvendig for å få en mer persis model.
- Laget prototype av last til robotarmen.
- Fokuserer på similator i Webots. Ettersom den fungerer bedre til vårt formål en Unity
- Enkelte ting har gått raskere en forventet, vi har vært litt dårlig på å oppdatere planen deretter.

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

- Inverskinematikken må vi utforske og teste mer etter hvert som vi får opp en fysisk prototype.
-

Main experience from this period

- Modellering med lagrange metode.
- Forståelse for invers og forward kinematikk
- Implementering av ArCuro Tag
- OPC server og Webots
- UR10 styring og programmering

Main purpose/focus next period

- Optimalisering av maskinsyn og evaluering
- Oppdatering UR10, bedre styring med threads for effektiv kommunikasjon
- Bedre på det organitatoriske.

Planned activities next period

- Optimalisere koden for objektgjennkjenning.
- Oppdatere firmware til UR10, planene er å kunne programere eksternt med RobotDK for lettere testing.
- Få bedre struktur og bruke threads for kommunikasjon med robotarm. Den må være hurtig for å kunne bevege seg raskt nokk og oppdatere verdier.
- Tilpasse inverskinematikk for robotarm og teste
- Finne parameter for modellen (Vekt, center of mass, dempningsfaktor, lengde på tråd)
- Lage teststasjon for maskinsyn

Other

Wish/need for counceling

- Samtale med Aleksander angående kinematikk og optimalisering av løsninger

Approval/signature group leader	Signature other group participants Mateusz Zedynak K Grandal

Fremdriftrapport

Prosjekt: Kompensering av pendelbevegelse på maritim kran
Periode: Uke 5-9
Oppdragsgiver: Seaonics AS
Dato: 04.03.2022

Main goal/purpose for this periods work

- Optimalisering av maskinsyn og evaluering
- Oppdatering UR10, bedre styring med tråder for effektiv kommunikasjon
- Bedre på det organisatoriske.

Planned activities this period

- Invers Kinematikk
- Tilpasse inverskinematikk for robotarm og teste
- Finne parameter for modellen (Vekt, center massen, dempingsfaktor, lengde på tråd)
- Lage teststasjon for maskinsyn

Actually conducted activities this period

- Testing av Inverskinematikk og løsninger, Løsninger er begrenset til bedre intervall. Likevel er det feil og må oppklares.
- Teststasjon for maskinsyn er ferdig.
- Webots – laget 2 prototype av UR10 for å teste ulike løsninger fra kinematikk matrisen
- Matlab visualisering for DH parameters (Kinematikk)
- Styring av robotarm med tråder og sending mottak via OPC. Fungerer greit og klarer å sende og motta vinkler og ønskelig informasjon. Videre må det undersøkes spesifikke styringskommandoer for robotarmen.

Description of/ justification for potential deviation between planned and real activities

- Grunnet annet emne som kjøres parallelt og intensivt har tidsforbruk for prosjektet blitt redusert betydelig.
- Original Kinematikk fungerte ikke som tiltenkt. Feil utgangspunkt i koordinatreferanse gir feil løsning og må undersøkes.
- Utfordrende å styre robotarm og kommunisere samtidig. Enkel kommando for styring av robotvinkler kan ikke kjøres parallelt ettersom roboten må fullføre en kommando før den gjennomfører neste. Videre skal vi undersøke inkrementerende kommando for å styre hver servo i robotarmen individuelt.

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

-

Main experience from this period

- Hvordan robotarmer planlegger en path. Bruk av jacobian.
- Bedrekjennskap til UR10 software og aktuelle pythonscript
- Kjennskap til OPC og Weebots.
- Bedre forståelse over koordinatsystem til vårt system.

Main purpose/focus next period

<table>
<tr><td colspan="2">- Ferdigstille kinematikken og få full kontroll over robotarm. Både i simulator og fysisk.
- Sammenslåing av robotstyring og objekt gjenkjenning.</td></tr>
<tr><td colspan="2">Planned activities next period
- Begrense antall løsninger fra kinematikken
- Kommunikasjon mellom Matlab og fysisk robot arm/prototype for å teste resultat fra kinematikken må ferdigstilles.
- Akselerasjon av endeffector
- Starte å implementere pid regulator for x-retning</td></tr>
<tr><td colspan="2">Other</td></tr>
<tr><td colspan="2">Wish/need for counceling
- Samtale med Aleksander og Agus angående kinematikk.</td></tr>
<tr><td>Approval/signature group leader
Trygve N Johannessen</td><td>Signature other group participants
Mateusz Zedynak
Viktor K Grandal</td></tr>
</table>

Fremdriftrapport

Prosjekt: kompansering av pendelbevegelse på maratim kran
Periode: uke 13-16
Oppdragsgiver: Seaonics AS
Dato: 22.04.2022

Main goal/purpose for this periods work

- Regulering i X(pixel) og Y(meter) retning
- Estimere tilstandsverdier vinkel, vinkelhastighet
- Optimalisering av løsning
- Utforske andre reguleringsmetoder

Planned activities this period

- Transformasjon av kamera koordinater til globale.
- Implementering av kamera på top.
- Kompansering av kameraorienting på top.
- Plots av regulering.
- Skriving av rapport.

Actually conducted activities this period

- Utledet og implementert transformasjon av kamerakoordinater til globale koordinater fra både kamera motert på tcp og på gulvet.
- Designet og konstruert kamerafeste på tcp.
- Utledet og implementert orienterings kompansering ved bruk av ledd 6.
- Logget og plottet grafer.
- Dokumenter en del implementeringer I metode delen på rapporten.

Description of/ justification for potential deviation between planned and real activities

- Kamera på toppen var ikke planlagt, men etter anbefalering av veileder og oppdagsgiver ble det igangsatt.
- Det ble utforsket en del LQR, men ikke implementert enda.

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

-

Main experience from this period

- Bytting av koordinatsystem
- Intrinsic og extrinsic matrise (pixel til meter)
- LQR

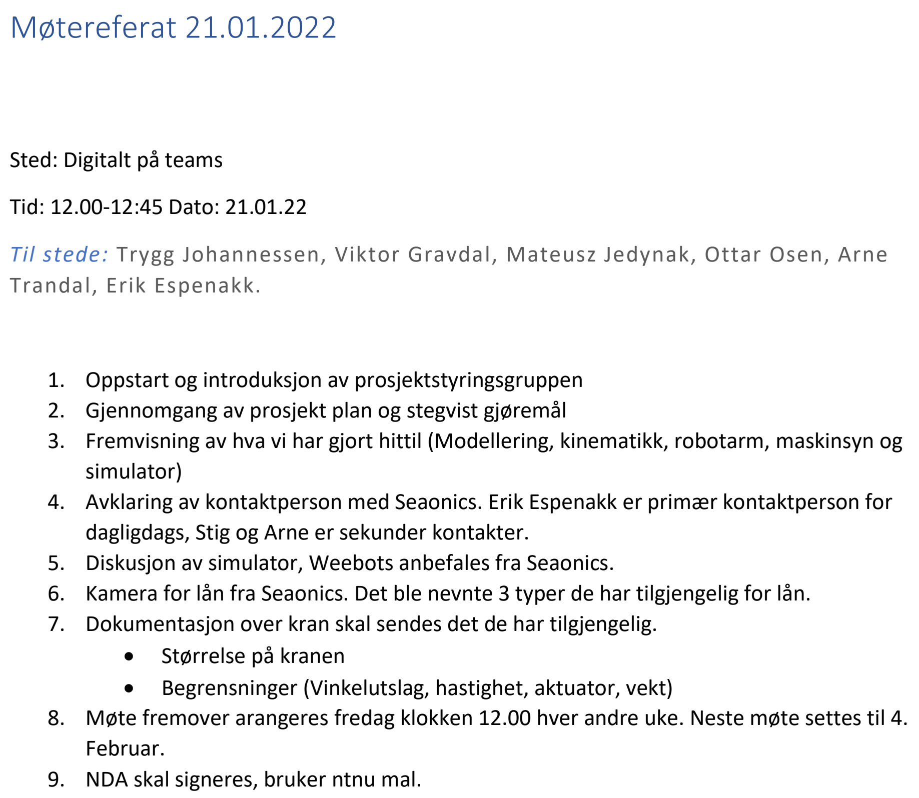
Main purpose/focus next period

- Prioritering liste
- 1. Nøyaktighet i målinger (kamera på golve, kamera på toppen)
- 2. Rapport
- 3. Optimalisering/Endring av PID i vinkel retning
- 3. Linear–quadratic regulator (LQR)
- 4. Testing av Knuckle boom crane og UR10 i webots
- 5. Eventuelt Nonlinear model predictive controllers (NMPC)

Planned activities next period

Other	
Wish/need for counceling - Ottar modul-regulering	
Approval/signature group leader Trygve M Johannessen	Signature other group participants Mateusz Zedynak Viktor K Grandal

A.4 Meeting summary



Møtereferat 21.01.2022

Sted: Digitalt på teams

Tid: 12.00-12:45 Dato: 21.01.22

Til stede: Trygg Johannessen, Viktor Gravdal, Mateusz Jedynak, Ottar Osen, Arne Trandal, Erik Espenakk.

1. Oppstart og introduksjon av prosjektstyringsgruppen
2. Gjennomgang av prosjekt plan og stegvist gjøremål
3. Fremvisning av hva vi har gjort hittil (Modellering, kinematikk, robotarm, maskinsyn og simulator)
4. Avklaring av kontaktperson med Seaonics. Erik Espenakk er primær kontaktperson for dagligdags, Stig og Arne er sekunder kontakter.
5. Diskusjon av simulator, Weebots anbefales fra Seaonics.
6. Kamera for lån fra Seaonics. Det ble nevnte 3 typer de har tilgjengelig for lån.
7. Dokumentasjon over kran skal sendes det de har tilgjengelig.
    - Størrelse på kranen
    - Begrensninger (Vinkelutslag, hastighet, aktuator, vekt)
8. Møte fremover arangeres fredag klokken 12.00 hver andre uke. Neste møte settes til 4. Februar.
9. NDA skal signeres, bruker ntnu mal.

Møtereferat 04.02.2022

Sted: Digitalt på teams

Tid: 12.00-12:45 Dato: 04.02.22

Til stede: Trygg Johannessen, Viktor Gravdal, Mateusz Jedynak, Ottar Osen, Aleksander l. Skrede, Arne Trandal, Erik Espenakk.

1. Gjennomgang av fremdrift og hva gruppen har oppnådd så langt
    - Bra fremgang
    - Ligger godt ant i forhold til planen

2. Diskusjon rundt implementering av maskinsyn
    - Se nærmere på lettere metoder for å finne vinkel og kunne se bort ifra posisjonen til objektet.

3. Anslår mindre tid til prosjektet i noen uker fremover pga annet emne.

Møtereferat 18.02.2022

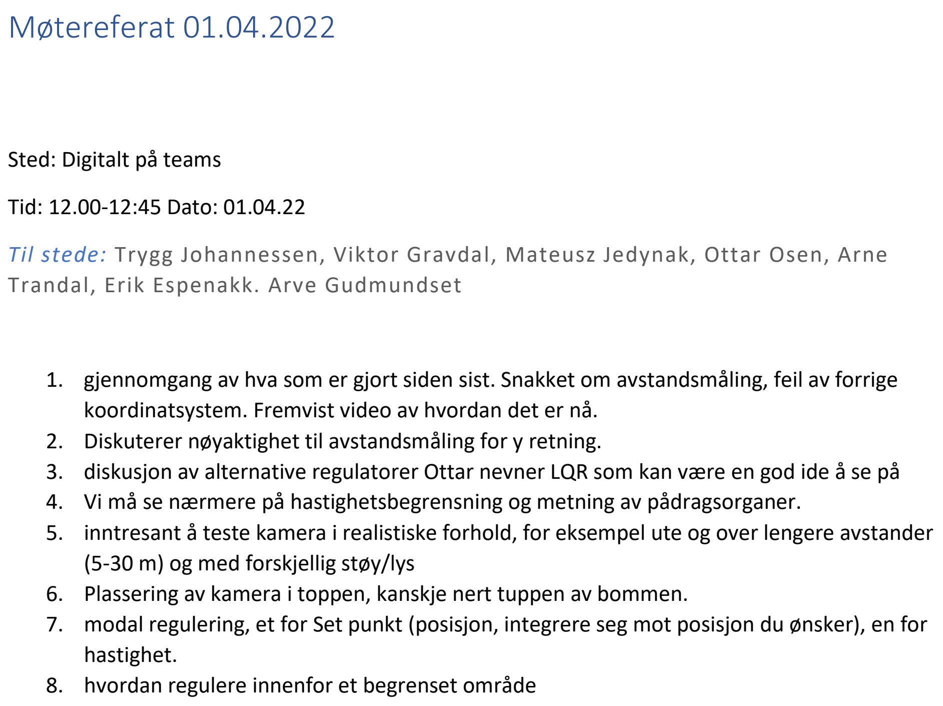
Sted: Digitalt på teams

Tid: 12.00-13:00 Dato: 18.02.22

Til stede: Trygg Johannessen, Viktor Gravdal, Mateusz Jedynak, Ottar Osen, Arne Trandal, Stig Espeseth, Aleksander l. Skrede.

- Presenterte hvor langt vi er kommet i fasene I prosjektet.
- Presenterte forksjellige metoder for bruk av maskinsyn.
  - Acuro
  - Yolov5
  - Fargedeteksjon
  - Konkluderte med at alle er nyttig for rapporten, men holder oss til farge deteksjon enn så lenge.
- Muligheter å feste lasten I taket for å se på lavere svingningsfrekvenser.
- Teste ut kinematikk I robotDK
- Diskusjoner angående pådraget (robotarm)
  - Optimalisering invers kinematikken
  - Gyldige løsninger for kranbevegelse og behandling av ikke gyldige løsninger
  - Begrense ledd, men la resterende ledd gi pådrag.
  - Jacobi matrise for å gi estimering av accerelation pådrag.
- Diskuterte muligheter for å være på seaonics sine lokaler.

Bra oppmøte og fint engasjement.



Møtereferat 01.04.2022

Sted: Digitalt på teams

Tid: 12.00-12:45 Dato: 01.04.22

Til stede: Trygg Johannessen, Viktor Gravdal, Mateusz Jedynak, Ottar Osen, Arne Trandal, Erik Espenakk. Arve Gudmundset

1. gjennomgang av hva som er gjort siden sist. Snakket om avstandsmåling, feil av forrige koordinatsystem. Fremvist video av hvordan det er nå.
2. Diskuterer nøyaktighet til avstandsmåling for y retning.
3. diskusjon av alternative regulatorer Ottar nevner LQR som kan være en god ide å se på
4. Vi må se nærmere på hastighetsbegrensning og metning av pådragsorganer.
5. inntresant å teste kamera i realistiske forhold, for eksempel ute og over lengere avstander (5-30 m) og med forskjellig støy/lys
6. Plassering av kamera i toppen, kanskje nert tuppen av bommen.
7. modal regulering, et for Set punkt (posisjon, integrere seg mot posisjon du ønsker), en for hastighet.
8. hvordan regulere innenfor et begrenset område

Møtereferat 22.04.2022

Sted: Seaonics AS

Tid: 12.00-12:45 Dato: 22.04.22

Til stede: Trygg Johannessen, Viktor Gravdal, Mateusz Jedynak, Ottar Osen, Aleksander L. Skrede, Arne Trandal, Erik Espenakk. Arve Gudmundset

1. Gjennomgang av utvikling frem til nå. Vist frem foreløpig resultat fra kamera på siden og kamera på top.

2. Prioritering av rapport og resultat sankning fremover.
3.

A.5 Meeting presentations

Uploaded as zip file on Inspera.

A.6 Proposed bachelor thesis

Forslag til bachelor oppgave

Viktor Karl Gravdal, Mateusz Jedynak, Trygg Meyer Johannessen.

January 14, 2022

Introduksjon

Dokumentet presenterer forslag for bachelor oppgave våren 2022. Forslaget står som et førsteutkast, endringer kan forekomme.

Beskrivelse

Under proskjekt skal vi utforske stabilisering av last på en kran. Flytting av last medfører pendling og oscillering avhengig av bevegelse. Oppgaven baserer seg på å minimere uønsket bevegelse på lasten. Vi skal utforske anvendelse av maskinsyn og diverse sensorer. Videre, ser vi på muligheten for å regulere pendelbevegelse. Ideen er å bruke en robotarm med redusert frihetsgrader som prototype for å simulere kranbevegelse. Eventuelt bruke en gantryplatform. Se figur 1 for illustrasjon

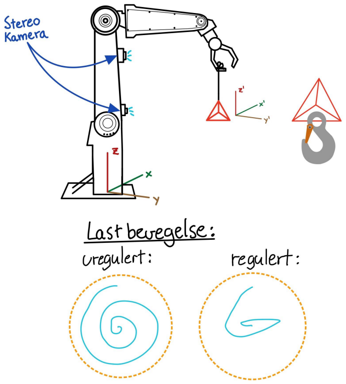


Figure 1: Illustrasjon av konsept

Mål:

Objekt-detektering av løftehodet

For å identifisere løftehodet/åket undersøker vi anvendelse av stereo kamera. Målet er å detektere og hente ut objektets koordinater x-, y- ,z- retning i sanntid.

Identifisering av pendelbevegelse

Målet er å konkret forutse retningsvektoren for pendel-bevegelsen

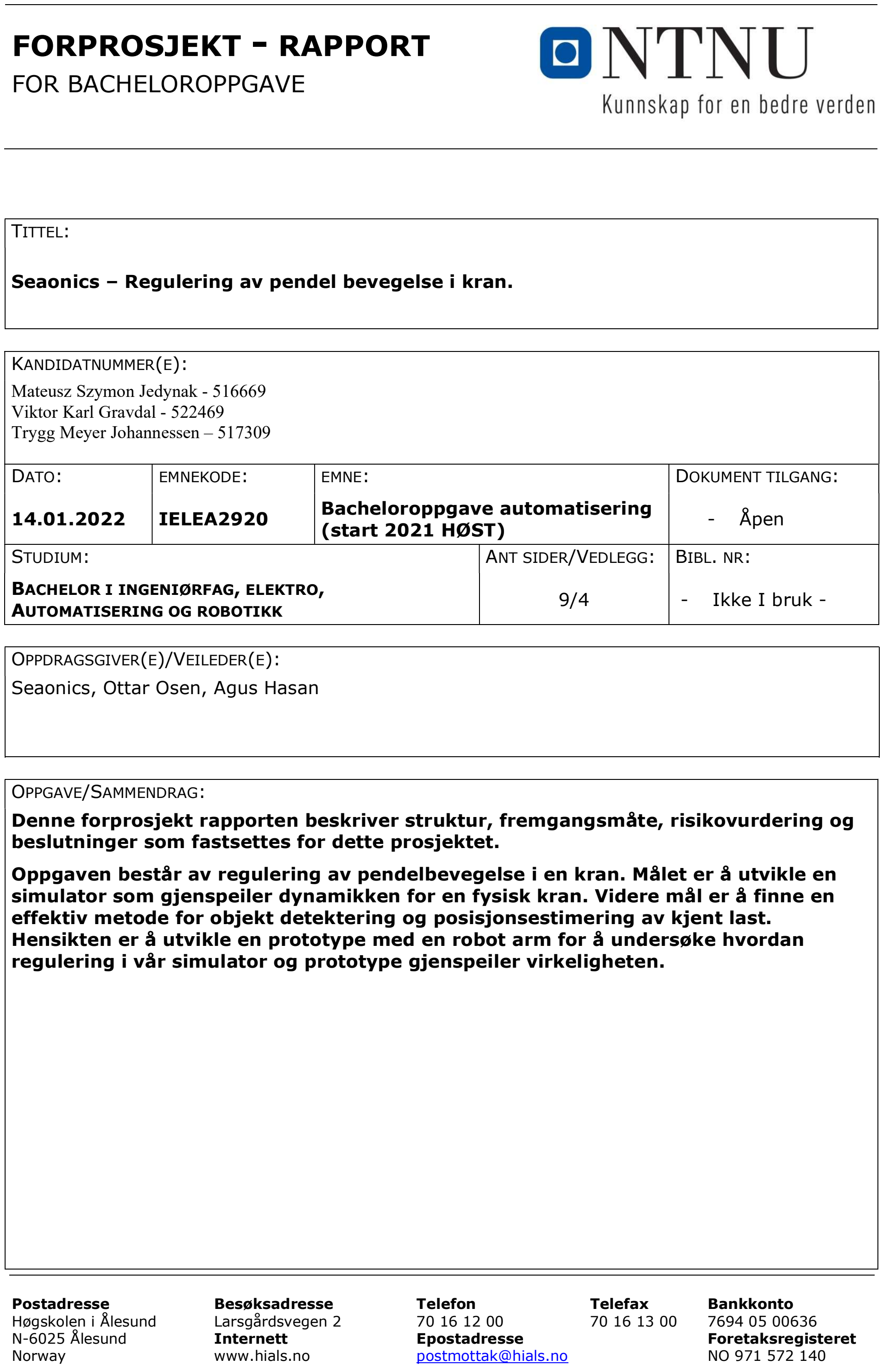
Kompensering for pendelbevegelse

Målet er å minimere pendelbevegelsen ved hjelp av moderne reguleringsmetoder.

Eventuelt

Videre tenker vi å utvide modellen slik at den kan ta høyde for bølgebevegelser, ved å bruke en 3D platform for å simulere bølger.

A.7 Pre-project - report



FORPROSJEKT - RAPPORT

FOR BACHELOROPPGAVE

NTNU
Kunnskap for en bedre verden

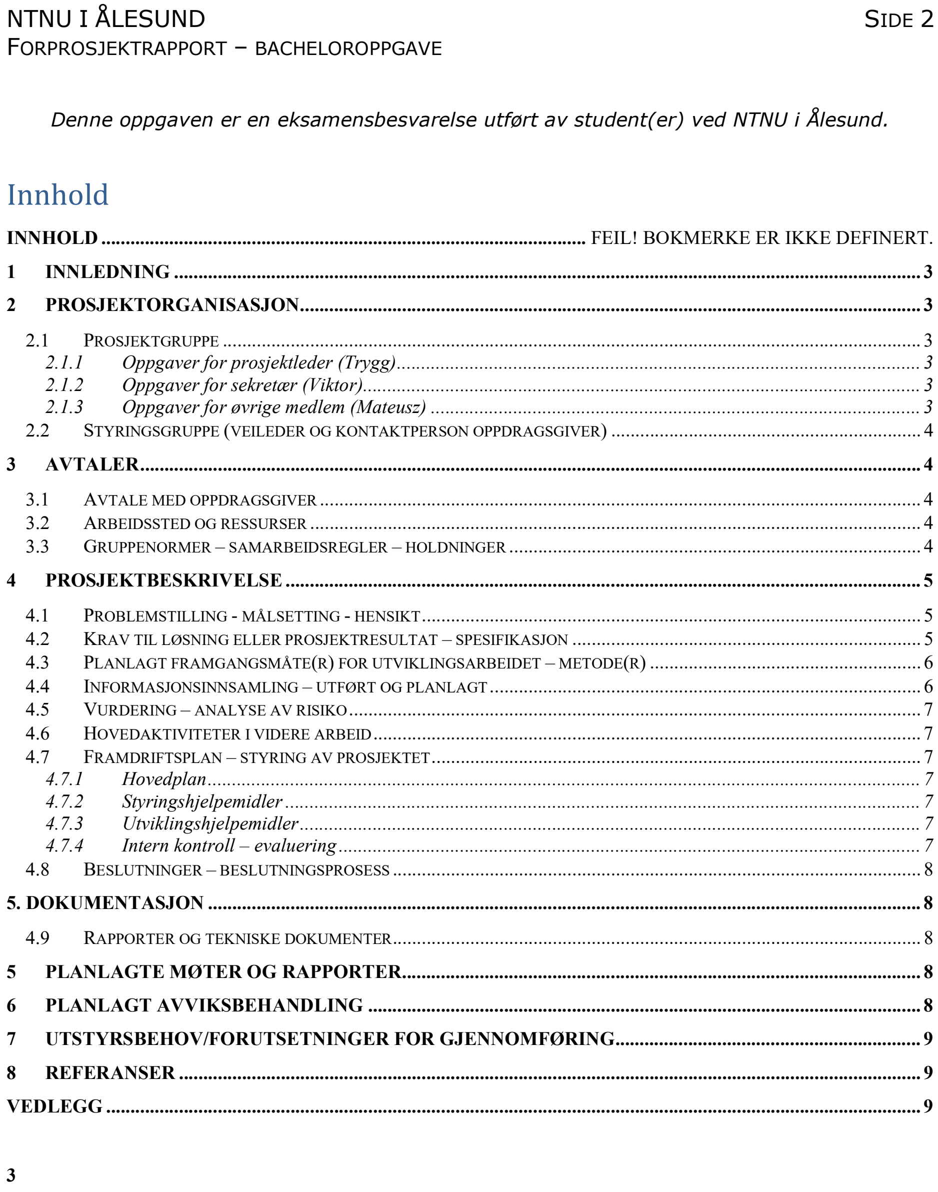
TITTEL: **Seaonics – Regulering av pendel bevegelse i kran.**			
KANDIDATNUMMER(E): Mateusz Szymon Jedynak - 516669 Viktor Karl Gravdal - 522469 Trygg Meyer Johannessen – 517309			
DATO: **14.01.2022**	EMNEKODE: **IELEA2920**	EMNE: **Bacheloroppgave automatisering (start 2021 HØST)**	DOKUMENT TILGANG: - Åpen
STUDIUM: **BACHELOR I INGENIØRFAG, ELEKTRO, AUTOMATISERING OG ROBOTIKK**		ANT SIDER/VEDLEGG: 9/4	BIBL. NR: - Ikke I bruk -
OPPDRAGSGIVER(E)/VEILEDER(E): Seaonics, Ottar Osen, Agus Hasan			

OPPGAVE/SAMMENDRAG:

Denne forprosjekt rapporten beskriver struktur, fremgangsmåte, risikovurdering og beslutninger som fastsettes for dette prosjektet.

Oppgaven består av regulering av pendelbevegelse i en kran. Målet er å utvikle en simulator som gjenspeiler dynamikken for en fysisk kran. Videre mål er å finne en effektiv metode for objekt detektering og posisjonsestimering av kjent last. Hensikten er å utvikle en prototype med en robot arm for å undersøke hvordan regulering i vår simulator og prototype gjenspeiler virkeligheten.

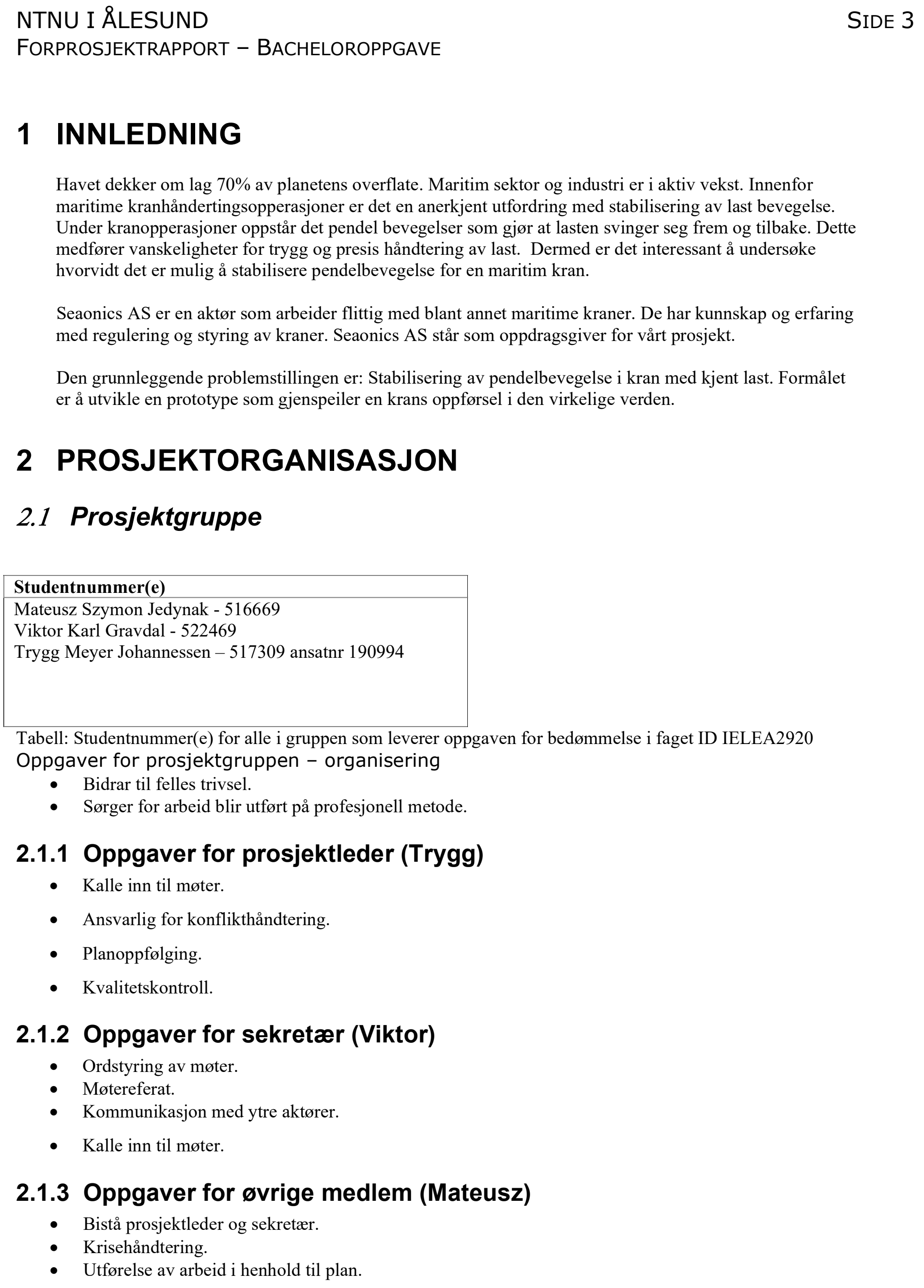
Postadresse	**Besøksadresse**	**Telefon**	**Telefax**	**Bankkonto**
Høgskolen i Ålesund	Larsgårdsvegen 2	70 16 12 00	70 16 13 00	7694 05 00636
N-6025 Ålesund	**Internett**	**Epostadresse**		**Foretaksregisteret**
Norway	www.hials.no	postmottak@hials.no		NO 971 572 140



Denne oppgaven er en eksamensbesvarelse utført av student(er) ved NTNU i Ålesund.

Innhold

3



1 INNLEDNING

Havet dekker om lag 70% av planetens overflate. Maritim sektor og industri er i aktiv vekst. Innenfor maritime kranhåndertingsopperasjoner er det en anerkjent utfordring med stabilisering av last bevegelse. Under kranopperasjoner oppstår det pendel bevegelser som gjør at lasten svinger seg frem og tilbake. Dette medfører vanskeligheter for trygg og presis håndtering av last. Dermed er det interessant å undersøke hvorvidt det er mulig å stabilisere pendelbevegelse for en maritim kran.

Seaonics AS er en aktør som arbeider flittig med blant annet maritime kraner. De har kunnskap og erfaring med regulering og styring av kraner. Seaonics AS står som oppdragsgiver for vårt prosjekt.

Den grunnleggende problemstillingen er: Stabilisering av pendelbevegelse i kran med kjent last. Formålet er å utvikle en prototype som gjenspeiler en krans oppførsel i den virkelige verden.

2 PROSJEKTORGANISASJON

2.1 Prosjektgruppe

Studentnummer(e)
Mateusz Szymon Jedynak - 516669 Viktor Karl Gravdal - 522469 Trygg Meyer Johannessen – 517309 ansatnr 190994

Tabell: Studentnummer(e) for alle i gruppen som leverer oppgaven for bedømmelse i faget ID IELEA2920

Oppgaver for prosjektgruppen – organisering

- Bidrar til felles trivsel.
- Sørger for arbeid blir utført på profesjonell metode.

2.1.1 Oppgaver for prosjektleder (Trygg)

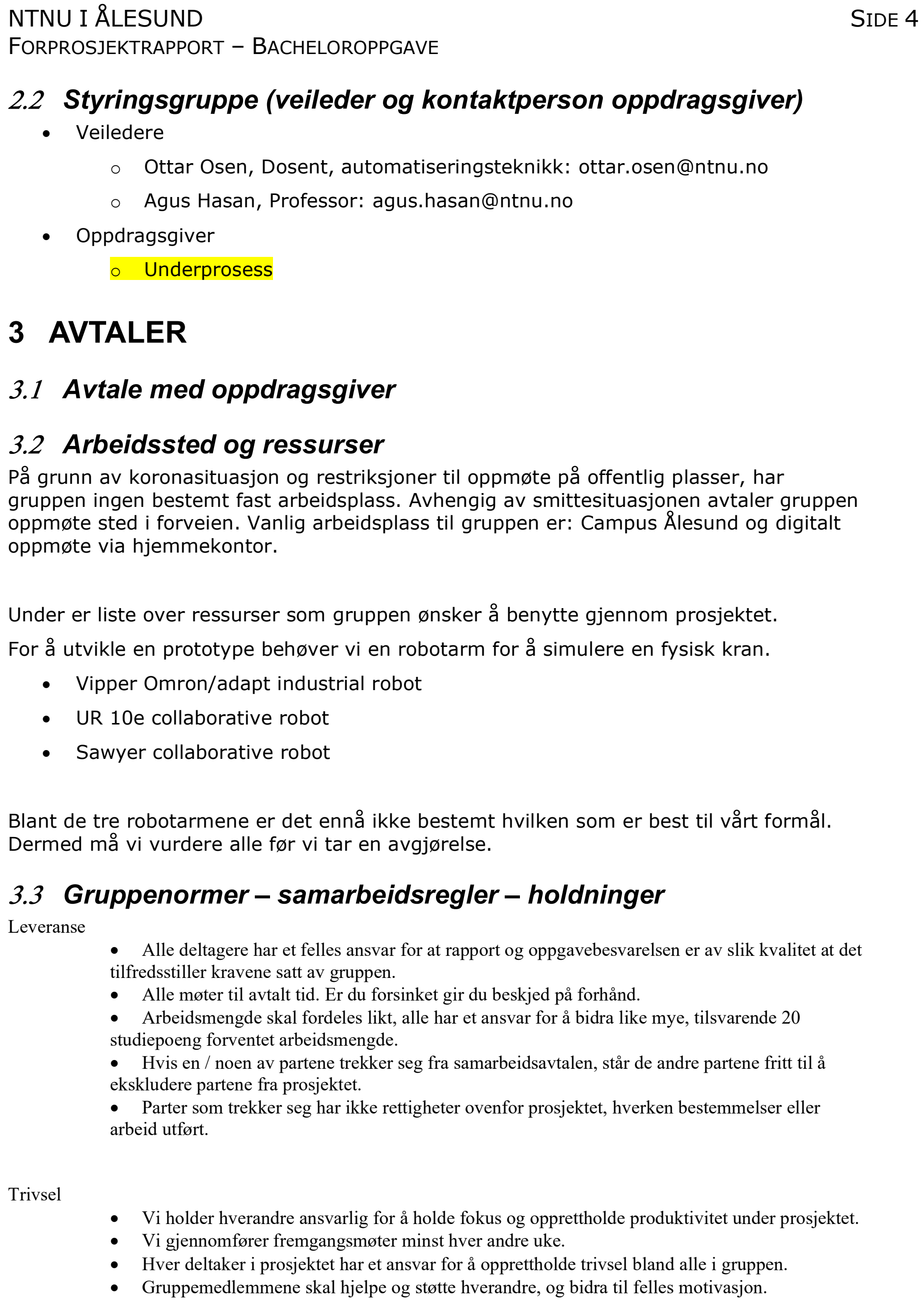
- Kalle inn til møter.
- Ansvarlig for konflikthåndtering.
- Planoppfølging.
- Kvalitetskontroll.

2.1.2 Oppgaver for sekretær (Viktor)

- Ordstyring av møter.
- Møtereferat.
- Kommunikasjon med ytre aktører.
- Kalle inn til møter.

2.1.3 Oppgaver for øvrige medlem (Mateusz)

- Bistå prosjektleder og sekretær.
- Krisehåndtering.
- Utførelse av arbeid i henhold til plan.



2.2 Styringsgruppe (veileder og kontaktperson oppdragsgiver)

- Veiledere
  - Ottar Osen, Dosent, automatiseringsteknikk: ottar.osen@ntnu.no
  - Agus Hasan, Professor: agus.hasan@ntnu.no
- Oppdragsgiver
  - Underprosess

3 AVTALER

3.1 Avtale med oppdragsgiver

3.2 Arbeidssted og ressurser

På grunn av koronasituasjon og restriksjoner til oppmøte på offentlig plasser, har gruppen ingen bestemt fast arbeidsplass. Avhengig av smittesituasjonen avtaler gruppen oppmøte sted i forveien. Vanlig arbeidsplass til gruppen er: Campus Ålesund og digitalt oppmøte via hjemmekontor.

Under er liste over ressurser som gruppen ønsker å benytte gjennom prosjektet.

For å utvikle en prototype behøver vi en robotarm for å simulere en fysisk kran.

- Vipper Omron/adapt industrial robot
- UR 10e collaborative robot
- Sawyer collaborative robot

Blant de tre robotarmene er det ennå ikke bestemt hvilken som er best til vårt formål. Dermed må vi vurdere alle før vi tar en avgjørelse.

3.3 Gruppenormer – samarbeidsregler – holdninger

Leveranse

- Alle deltagere har et felles ansvar for at rapport og oppgavebesvarelsen er av slik kvalitet at det tilfredsstiller kravene satt av gruppen.
- Alle møter til avtalt tid. Er du forsinket gir du beskjed på forhånd.
- Arbeidsmengde skal fordeles likt, alle har et ansvar for å bidra like mye, tilsvarende 20 studiepoeng forventet arbeidsmengde.
- Hvis en / noen av partene trekker seg fra samarbeidsavtalen, står de andre partene fritt til å ekskludere partene fra prosjektet.
- Parter som trekker seg har ikke rettigheter ovenfor prosjektet, hverken bestemmelser eller arbeid utført.

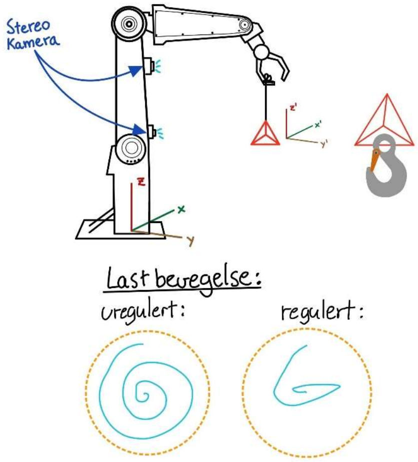
Trivsel

- Vi holder hverandre ansvarlig for å holde fokus og opprettholde produktivitet under prosjektet.
- Vi gjennomfører fremgangsmøter minst hver andre uke.
- Hver deltaker i prosjektet har et ansvar for å opprettholde trivsel bland alle i gruppen.
- Gruppemedlemmene skal hjelpe og støtte hverandre, og bidra til felles motivasjon.

4 PROSJEKTBESKRIVELSE

4.1 Problemstilling - målsetting - hensikt

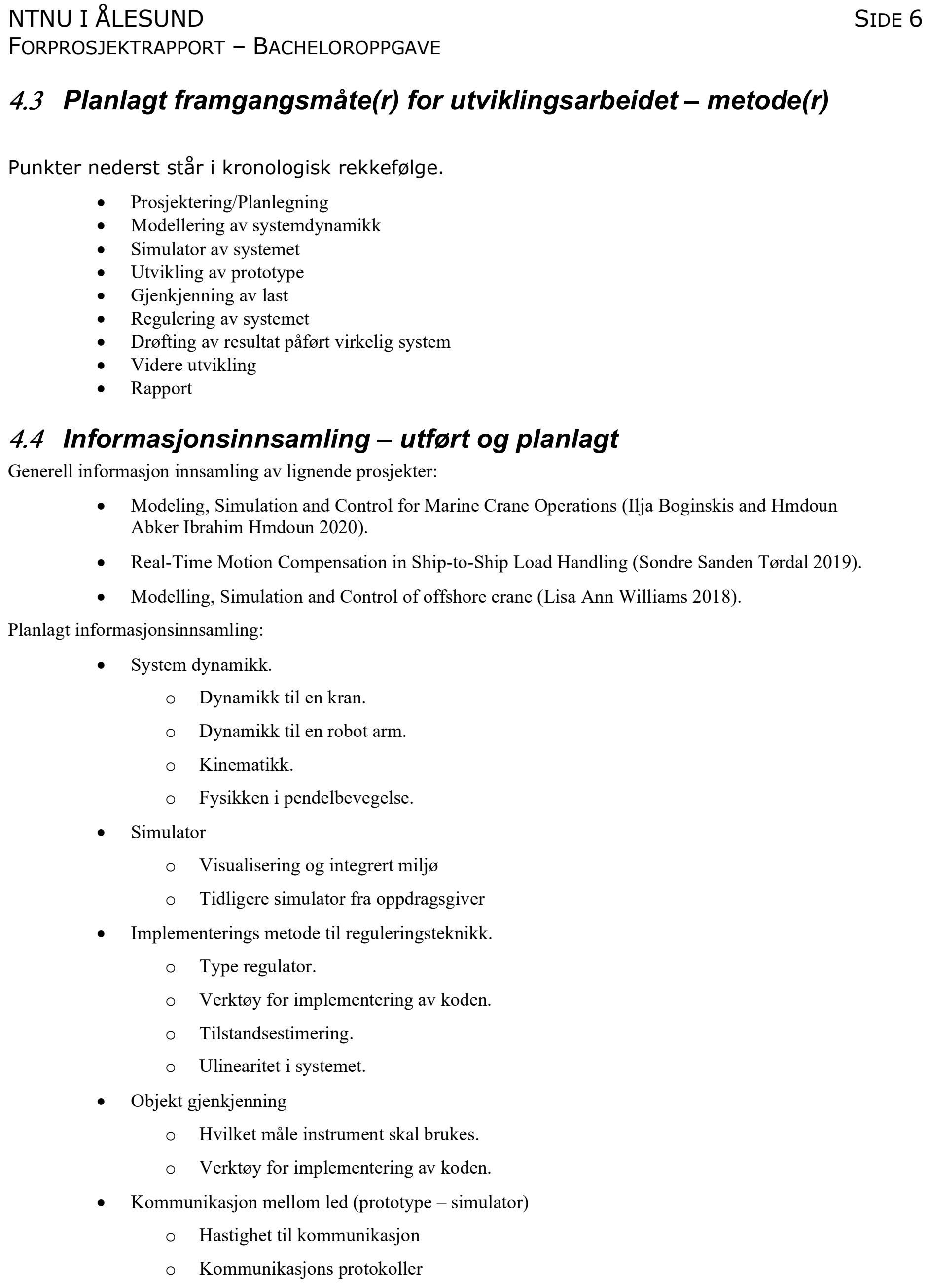
Grunnleggende problemstilling er regulering av pendelbevegelse for kjent last på en kran. For å utføre dette skal vi produsere en simulator. Med simulatoren skal vi teste ulike reguleringsmetoder e.g. PID, MPC. Videre skal vi benytte en fysisk robotarm, med redusert frihetsgrad for å etterligne kranbevegelse til en fysisk kran. Dette danner en prototype. For å regulere prototypen må vi detektere lasten som en form for posisjon estimering. Posisjonsestimeringen utføres enten med bruk av sensorer eller kamera (Maskinsyn). Hensikten er å bruke kunnskapen fra prototypen for å vurdere hvordan det gjenspeiler seg i en fysisk kran. Dette gjør vi ved å bruke en digital tvilling til en fysisk kran. Slik kan vi drøfte kontroll systemets frekvensrespons til en fysisk kran og for vår prototype, for å kunne konkludere hvordan en praktisk løsning kan implementeres på et virkelig system. Figur 1 viser en illustrasjon av prototypen



Figur 1 Illustrasjon av prototype

4.2 Krav til løsning eller prosjektresultat – spesifikasjon

- Stabilisering av posisjon til last som er i bevegelse.
- Simulator som tilnærmer seg virkeligheten.
- Utforskning innom ulike regulerings metoder og drøfte resultater.
- Detektering av last ved bruk av ulike sensorer.
- Drøfte problemstillinger og begrensninger til kran i forhold til prototype.
- Drøfting av systemdynamikk til prototypen (Robot arm) i forhold til en kran.



4.3 Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)

Punkter nederst står i kronologisk rekkefølge.

- Prosjektering/Planlegning
- Modellering av systemdynamikk
- Simulator av systemet
- Utvikling av prototype
- Gjenkjenning av last
- Regulering av systemet
- Drøfting av resultat påført virkelig system
- Videre utvikling
- Rapport

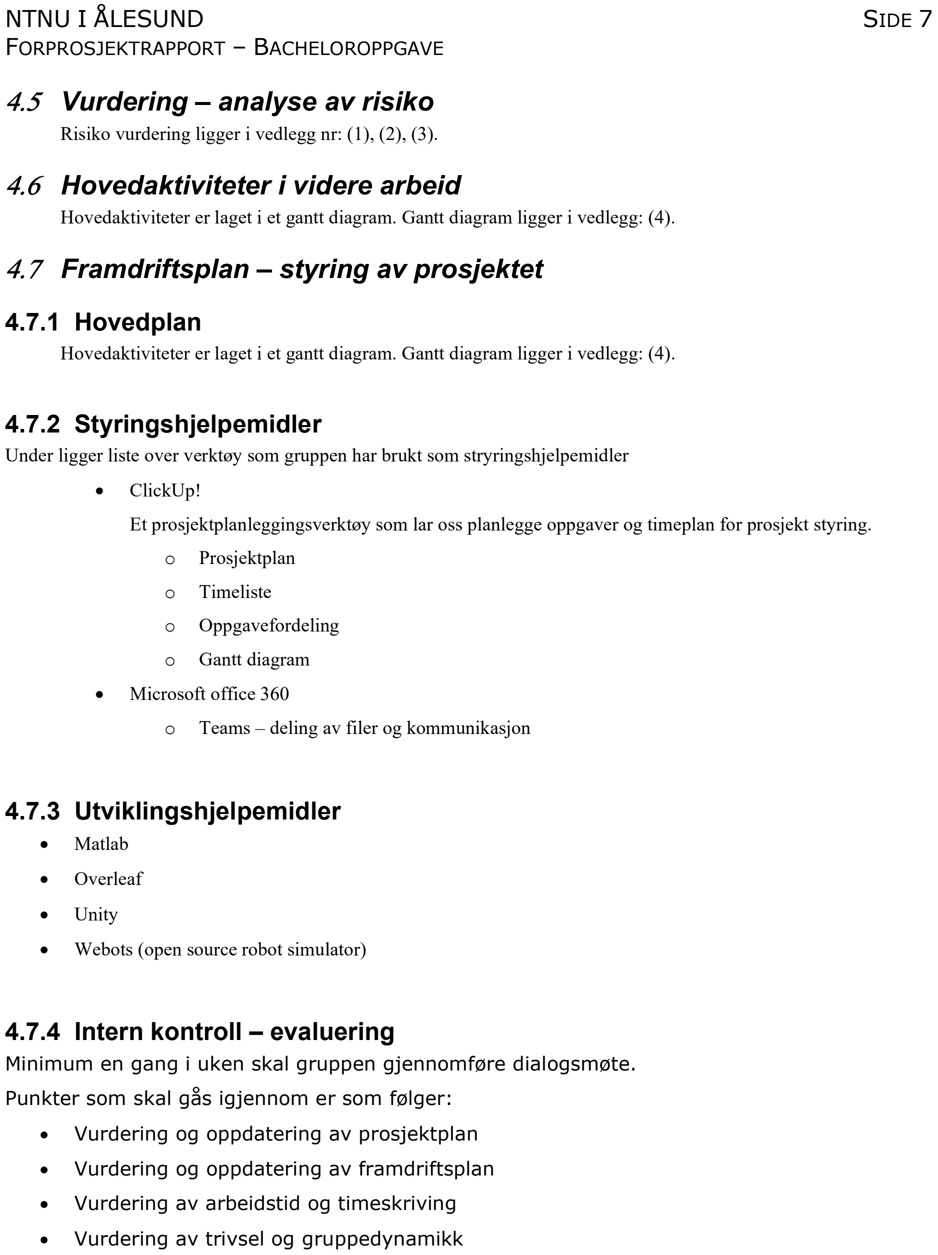
4.4 Informasjonsinnsamling – utført og planlagt

Generell informasjon innsamling av lignende prosjekter:

- Modeling, Simulation and Control for Marine Crane Operations (Ilja Boginskis and Hmdoun Abker Ibrahim Hmdoun 2020).
- Real-Time Motion Compensation in Ship-to-Ship Load Handling (Sondre Sanden Tørdal 2019).
- Modelling, Simulation and Control of offshore crane (Lisa Ann Williams 2018).

Planlagt informasjonsinnsamling:

- System dynamikk.
  - Dynamikk til en kran.
  - Dynamikk til en robot arm.
  - Kinematikk.
  - Fysikken i pendelbevegelse.
- Simulator
  - Visualisering og integrert miljø
  - Tidligere simulator fra oppdragsgiver
- Implementerings metode til reguleringsteknikk.
  - Type regulator.
  - Verktøy for implementering av koden.
  - Tilstandsestimering.
  - Ulinearitet i systemet.
- Objekt gjenkjenning
  - Hvilket måle instrument skal brukes.
  - Verktøy for implementering av koden.
- Kommunikasjon mellom led (prototype – simulator)
  - Hastighet til kommunikasjon
  - Kommunikasjons protokoller



4.5 Vurdering – analyse av risiko

Risiko vurdering ligger i vedlegg nr: (1), (2), (3).

4.6 Hovedaktiviteter i videre arbeid

Hovedaktiviteter er laget i et gantt diagram. Gantt diagram ligger i vedlegg: (4).

4.7 Framdriftsplan – styring av prosjektet

4.7.1 Hovedplan

Hovedaktiviteter er laget i et gantt diagram. Gantt diagram ligger i vedlegg: (4).

4.7.2 Styringshjelpemidler

Under ligger liste over verktøy som gruppen har brukt som stryringshjelpemidler

- ClickUp!

  Et prosjektplanleggingsverktøy som lar oss planlegge oppgaver og timeplan for prosjekt styring.
  - Prosjektplan
  - Timeliste
  - Oppgavefordeling
  - Gantt diagram
- Microsoft office 360
  - Teams – deling av filer og kommunikasjon

4.7.3 Utviklingshjelpemidler

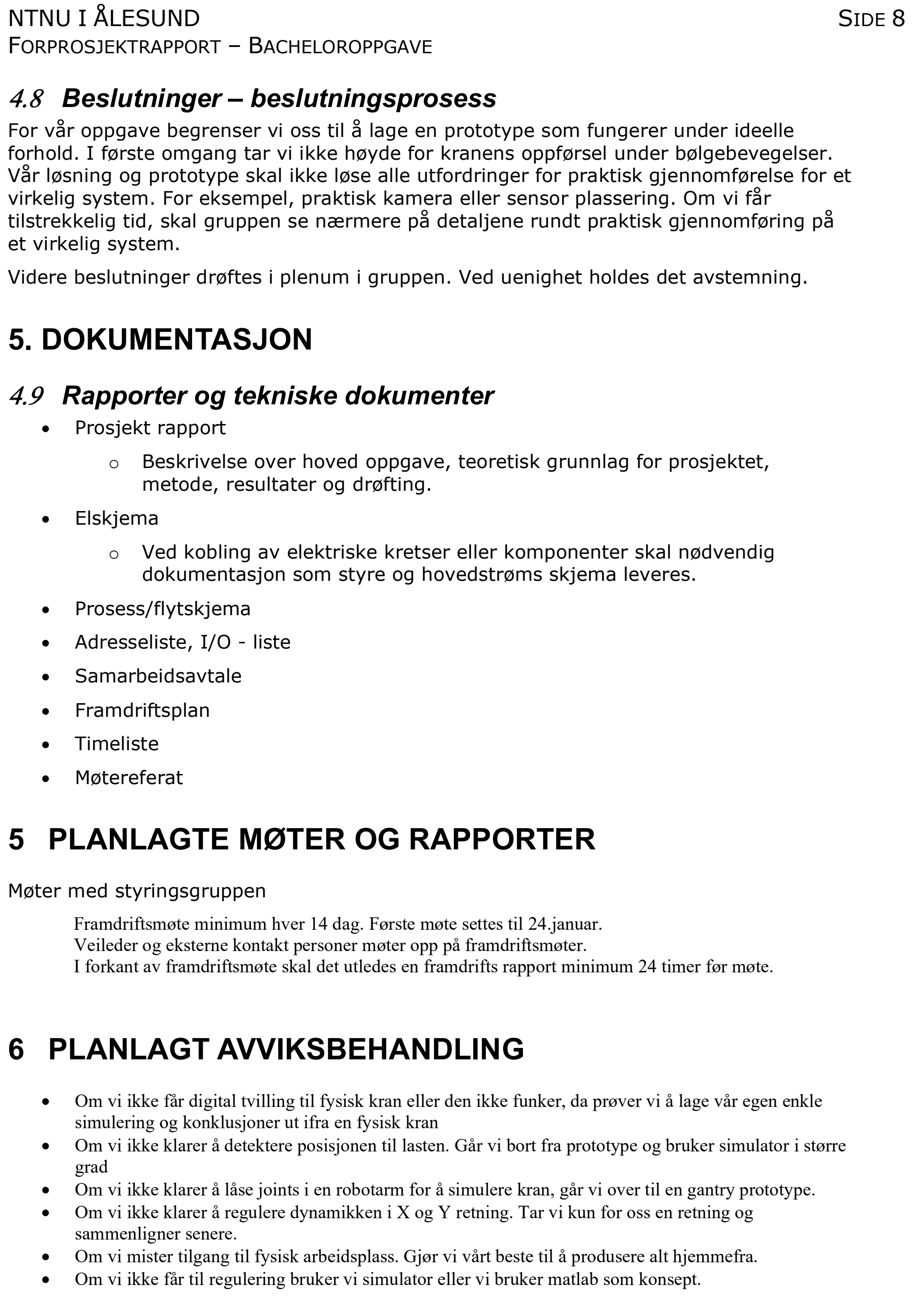
- Matlab
- Overleaf
- Unity
- Webots (open source robot simulator)

4.7.4 Intern kontroll – evaluering

Minimum en gang i uken skal gruppen gjennomføre dialogsmøte.

Punkter som skal gås igjennom er som følger:

- Vurdering og oppdatering av prosjektplan
- Vurdering og oppdatering av framdriftsplan
- Vurdering av arbeidstid og timeskriving
- Vurdering av trivsel og gruppedynamikk



4.8 Beslutninger – beslutningsprosess

For vår oppgave begrenser vi oss til å lage en prototype som fungerer under ideelle forhold. I første omgang tar vi ikke høyde for kranens oppførsel under bølgebevegelser. Vår løsning og prototype skal ikke løse alle utfordringer for praktisk gjennomførelse for et virkelig system. For eksempel, praktisk kamera eller sensor plassering. Om vi får tilstrekkelig tid, skal gruppen se nærmere på detaljene rundt praktisk gjennomføring på et virkelig system.

Videre beslutninger drøftes i plenum i gruppen. Ved uenighet holdes det avstemning.

5. DOKUMENTASJON

4.9 Rapporter og tekniske dokumenter

- Prosjekt rapport
  - Beskrivelse over hoved oppgave, teoretisk grunnlag for prosjektet, metode, resultater og drøfting.
- Elskjema
  - Ved kobling av elektriske kretser eller komponenter skal nødvendig dokumentasjon som styre og hovedstrøms skjema leveres.
- Prosess/flytskjema
- Adresseliste, I/O - liste
- Samarbeidsavtale
- Framdriftsplan
- Timeliste
- Møtereferat

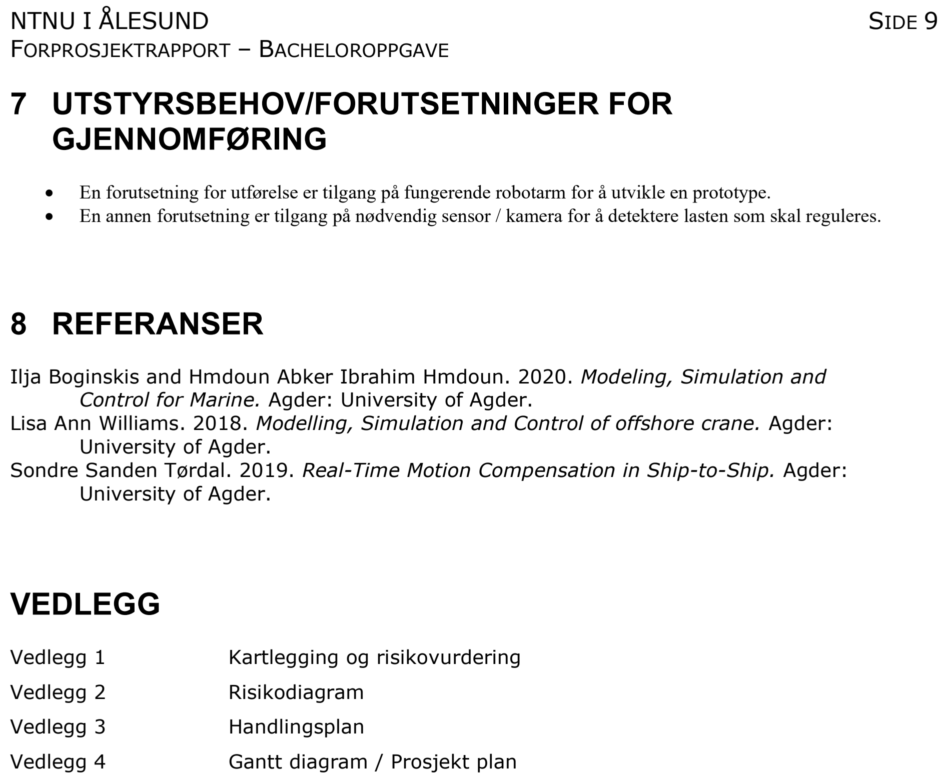
5 PLANLAGTE MØTER OG RAPPORTER

Møter med styringsgruppen

Framdriftsmøte minimum hver 14 dag. Første møte settes til 24.januar.
Veileder og eksterne kontakt personer møter opp på framdriftsmøter.
I forkant av framdriftsmøte skal det utledes en framdrifts rapport minimum 24 timer før møte.

6 PLANLAGT AVVIKSBEHANDLING

- Om vi ikke får digital tvilling til fysisk kran eller den ikke funker, da prøver vi å lage vår egen enkle simulering og konklusjoner ut ifra en fysisk kran
- Om vi ikke klarer å detektere posisjonen til lasten. Går vi bort fra prototype og bruker simulator i større grad
- Om vi ikke klarer å låse joints i en robotarm for å simulere kran, går vi over til en gantry prototype.
- Om vi ikke klarer å regulere dynamikken i X og Y retning. Tar vi kun for oss en retning og sammenligner senere.
- Om vi mister tilgang til fysisk arbeidsplass. Gjør vi vårt beste til å produsere alt hjemmefra.
- Om vi ikke får til regulering bruker vi simulator eller vi bruker matlab som konsept.



7 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING

- En forutsetning for utførelse er tilgang på fungerende robotarm for å utvikle en prototype.
- En annen forutsetning er tilgang på nødvendig sensor / kamera for å detektere lasten som skal reguleres.

8 REFERANSER

Ilja Boginskis and Hmdoun Abker Ibrahim Hmdoun. 2020. *Modeling, Simulation and Control for Marine.* Agder: University of Agder.

Lisa Ann Williams. 2018. *Modelling, Simulation and Control of offshore crane.* Agder: University of Agder.

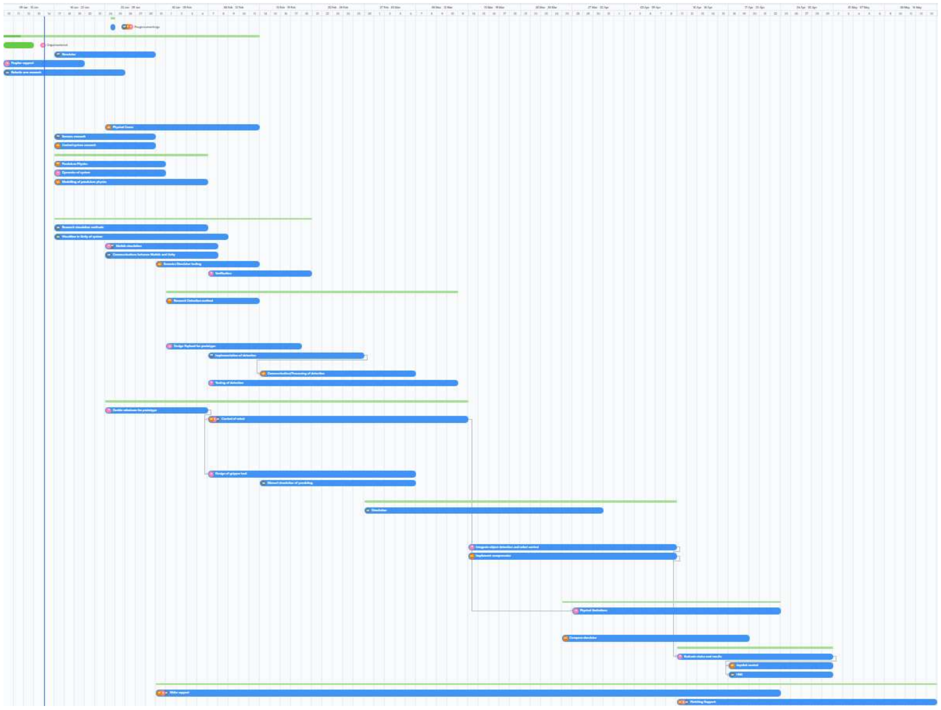
Sondre Sanden Tørdal. 2019. *Real-Time Motion Compensation in Ship-to-Ship.* Agder: University of Agder.

VEDLEGG

Vedlegg 1	Kartlegging og risikovurdering
Vedlegg 2	Risikodiagram
Vedlegg 3	Handlingsplan
Vedlegg 4	Gantt diagram / Prosjekt plan

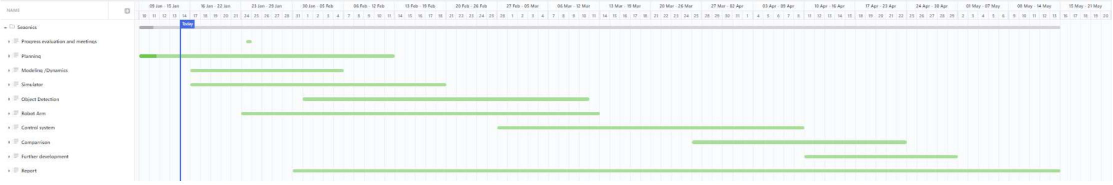
A.8 First draft of gantt diagram

Oversikt over gantt diagram



Side 1 for Planlegging

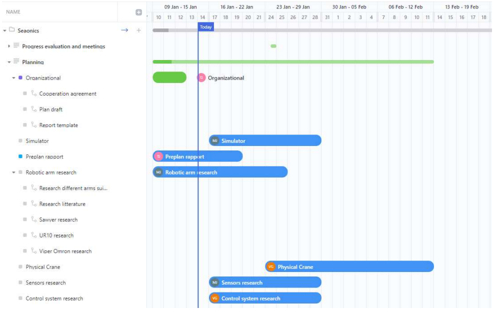
Oversikt over faser



NAME

- Seaonics
  - Progress evaluation and meetings
  - Planning
  - Modeling /Dynamics
  - Simulator
  - Object Detection
  - Robot Arm
  - Control system
  - Comparrison
  - Further development
  - Report

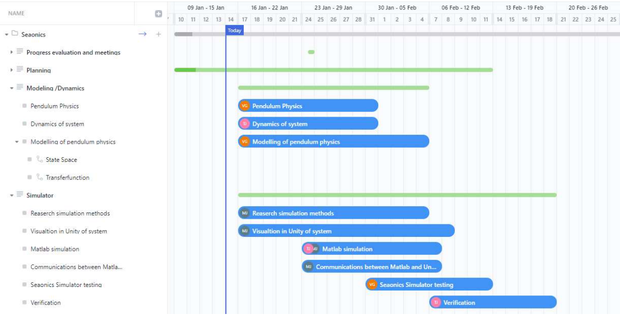
1. Planlegning fase



NAME

- Seaonics
  - Progress evaluation and meetings
  - Planning
    - Organizational
      - Cooperation agreement
      - Plan draft
      - Report template
    - Simulator
    - Preplan rapport
    - Robotic arm research
      - Research different arms sui...
      - Research litterature
      - Sawyer research
      - UR10 research
      - Viper Omron research
    - Physical Crane
    - Sensors research
    - Control system research

2. Modellering og dynamikk

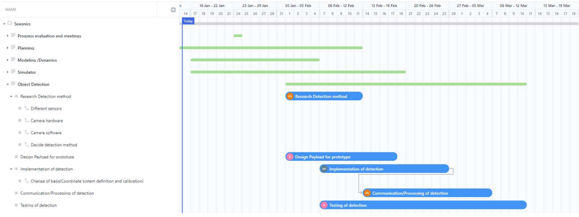


NAME

- Seaonics
  - Progress evaluation and meetings
  - Planning
  - Modeling /Dynamics
    - Pendulum Physics
    - Dynamics of system
    - Modelling of pendulum physics
      - State Space
      - Transferfunction
  - Simulator
    - Reaserch simulation methods
    - Visualtion in Unity of system
    - Matlab simulation
    - Communications between Matla...
    - Seaonics Simulator testing
    - Verification

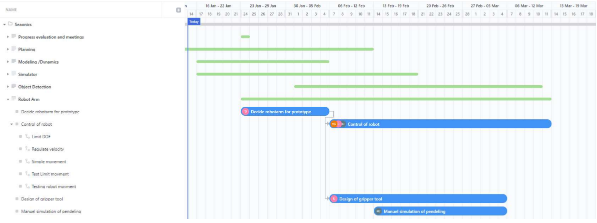
Side 2 for Planlegging

3. Objekt gjenkjenning


NAME

- Seaonics
  - Progress evaluation and meetings
  - Planning
  - Modeling /Dynamics
  - Simulator
  - Object Detection
    - Research Detection method
      - Different sensors
      - Camera hardware
      - Camera software
      - Decide detection method
    - Design Payload for prototype
    - Implementation of detection
      - Change of basis(Coordinate system definition and calibration)
    - Communication/Processing of detection
    - Testing of detection

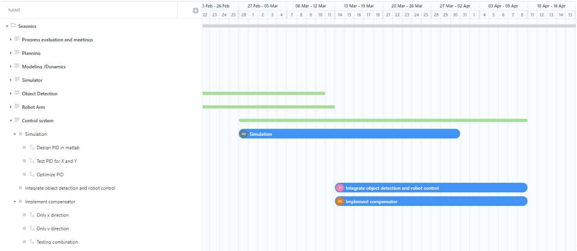
4. Robot arm


NAME

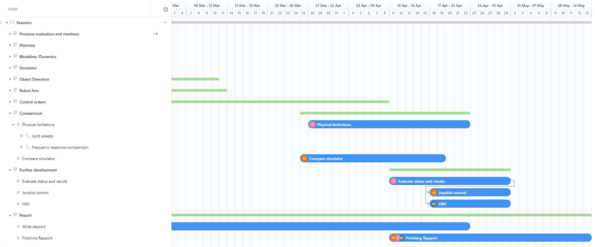
- Seaonics
  - Progress evaluation and meetings
  - Planning
  - Modeling /Dynamics
  - Simulator
  - Object Detection
  - Robot Arm
    - Decide robotarm for prototype
    - Control of robot
      - Limit DOF
      - Regulate velocity
      - Simple movement
      - Test Limit movment
      - Testing robot movment
    - Design of gripper tool
    - Manuel simulation of pendeling

5. Regulering

6. Sammenligning, Utviklingsfase Rapport


NAME

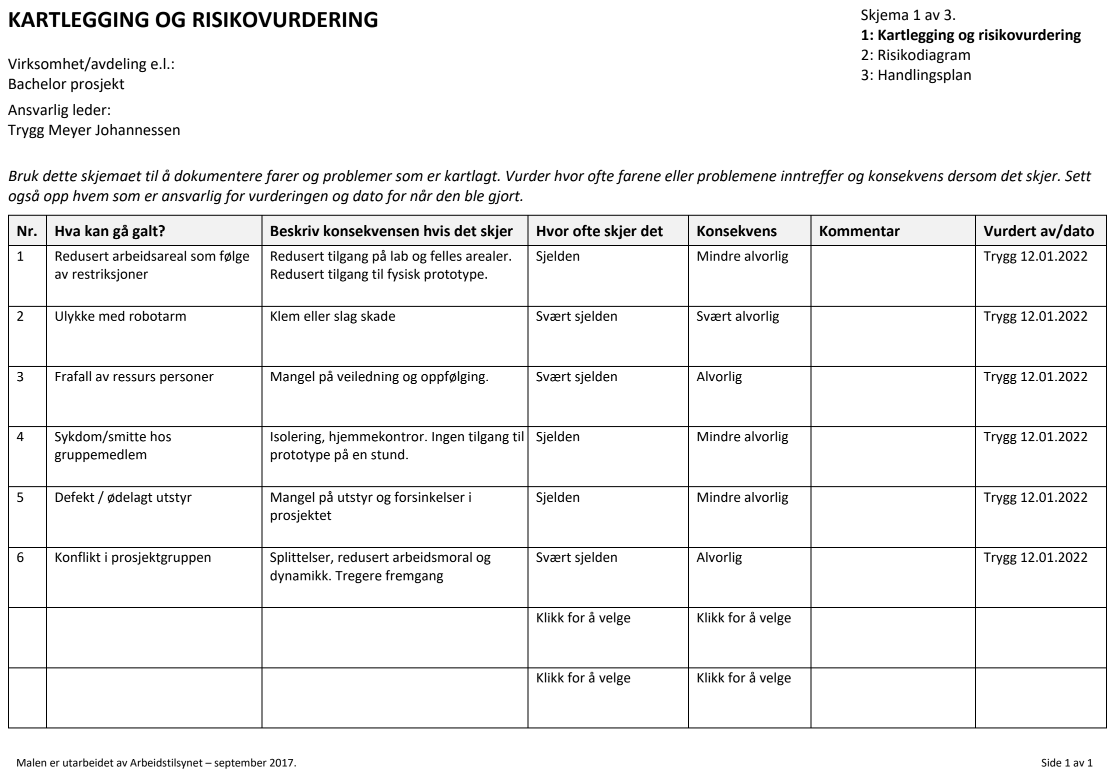
- Seaonics
  - Progress evaluation and meetings
  - Planning
  - Modeling /Dynamics
  - Simulator
  - Object Detection
  - Robot Arm
  - Control system
    - Simulation
      - Design PID in matlab
      - Test PID for X and Y
      - Optimize PID
    - Integrate object detection and robot control
    - Implement compensator
      - Only x direction
      - Only y direction
      - Testing combination


NAME

- Seaonics
  - Progress evaluation and meetings
  - Planning
  - Modeling /Dynamics
  - Simulator
  - Object Detection
  - Robot Arm
  - Control system
  - Comparrison
    - Physical limitations
      - Joint speeds
      - Frequency response comparrison
    - Compare simulator
  - Further development
    - Evaluate status and results
    - Joystick control
    - HMI
  - Report
    - Write rapport
    - Finishing Rapport

Side 3 for Planlegging

A.9 Risk assessment



KARTLEGGING OG RISIKOVURDERING

Skjema 1 av 3.
1: Kartlegging og risikovurdering
2: Risikodiagram
3: Handlingsplan

Virksomhet/avdeling e.l.:
Bachelor prosjekt

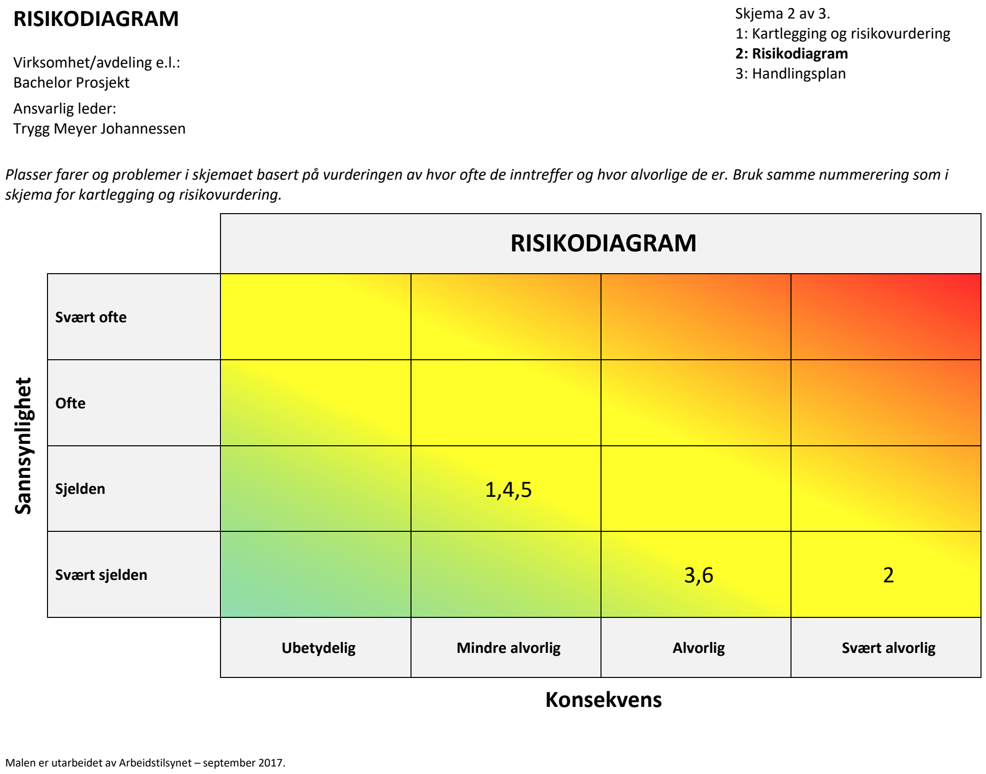
Ansvarlig leder:
Trygg Meyer Johannessen

Bruk dette skjemaet til å dokumentere farer og problemer som er kartlagt. Vurder hvor ofte farene eller problemene inntreffer og konsekvens dersom det skjer. Sett også opp hvem som er ansvarlig for vurderingen og dato for når den ble gjort.

Nr.	Hva kan gå galt?	Beskriv konsekvensen hvis det skjer	Hvor ofte skjer det	Konsekvens	Kommentar	Vurdert av/dato
1	Redusert arbeidsareal som følge av restriksjoner	Redusert tilgang på lab og felles arealer. Redusert tilgang til fysisk prototype.	Sjelden	Mindre alvorlig		Trygg 12.01.2022
2	Ulykke med robotarm	Klem eller slag skade	Svært sjelden	Svært alvorlig		Trygg 12.01.2022
3	Frafall av ressurs personer	Mangel på veiledning og oppfølging.	Svært sjelden	Alvorlig		Trygg 12.01.2022
4	Sykdom/smitte hos gruppemedlem	Isolering, hjemmekontror. Ingen tilgang til prototype på en stund.	Sjelden	Mindre alvorlig		Trygg 12.01.2022
5	Defekt / ødelagt utstyr	Mangel på utstyr og forsinkelser i prosjektet	Sjelden	Mindre alvorlig		Trygg 12.01.2022
6	Konflikt i prosjektgruppen	Splittelser, redusert arbeidsmoral og dynamikk. Tregere fremgang	Svært sjelden	Alvorlig		Trygg 12.01.2022
			Klikk for å velge	Klikk for å velge		
			Klikk for å velge	Klikk for å velge		

Malen er utarbeidet av Arbeidstilsynet – september 2017.

Side 1 av 1



RISIKODIAGRAM

Skjema 2 av 3.
1: Kartlegging og risikovurdering
2: Risikodiagram
3: Handlingsplan

Virksomhet/avdeling e.l.:
Bachelor Prosjekt

Ansvarlig leder:
Trygg Meyer Johannessen

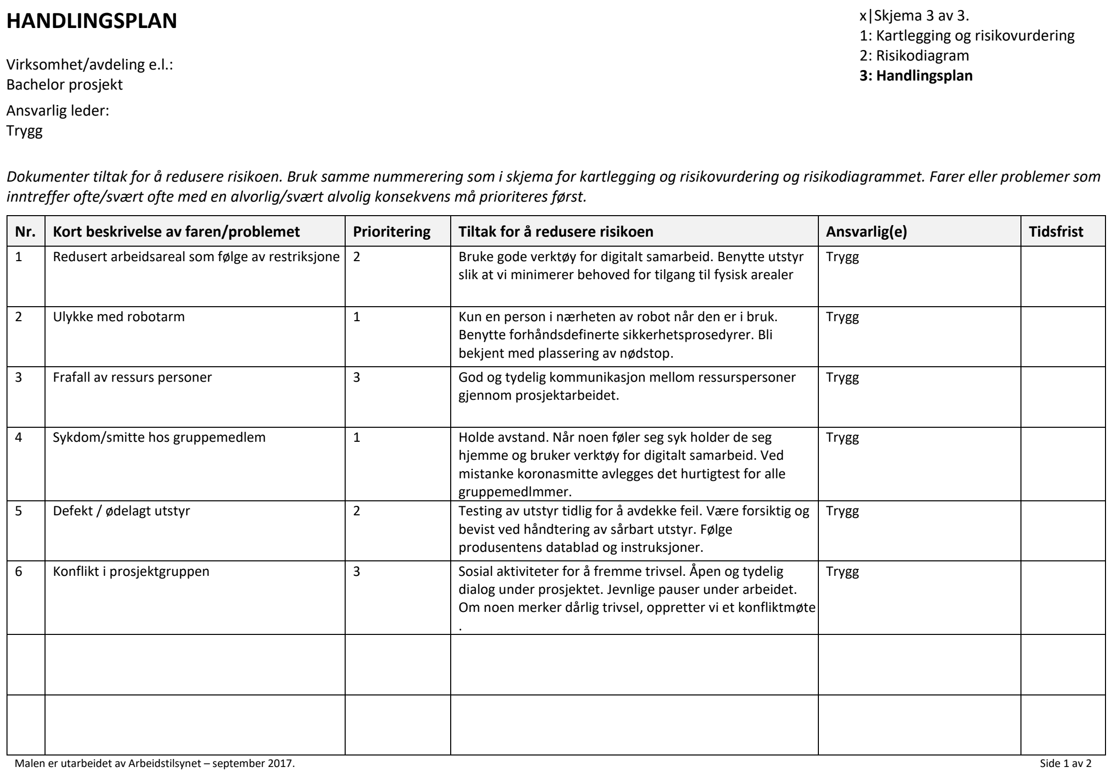
Plasser farer og problemer i skjemaet basert på vurderingen av hvor ofte de inntreffer og hvor alvorlige de er. Bruk samme nummerering som i skjema for kartlegging og risikovurdering.

Sannsynlighet

	RISIKODIAGRAM			
Svært ofte				
Ofte				
Sjelden		1,4,5		
Svært sjelden			3,6	2
	Ubetydelig	**Mindre alvorlig**	**Alvorlig**	**Svært alvorlig**

Konsekvens

Malen er utarbeidet av Arbeidstilsynet – september 2017.



HANDLINGSPLAN

x|Skjema 3 av 3.
1: Kartlegging og risikovurdering
2: Risikodiagram
3: Handlingsplan

Virksomhet/avdeling e.l.:
Bachelor prosjekt

Ansvarlig leder:
Trygg

Dokumenter tiltak for å redusere risikoen. Bruk samme nummerering som i skjema for kartlegging og risikovurdering og risikodiagrammet. Farer eller problemer som inntreffer ofte/svært ofte med en alvorlig/svært alvolig konsekvens må prioriteres først.

Nr.	**Kort beskrivelse av faren/problemet**	**Prioritering**	**Tiltak for å redusere risikoen**	**Ansvarlig(e)**	**Tidsfrist**
1	Redusert arbeidsareal som følge av restriksjone	2	Bruke gode verktøy for digitalt samarbeid. Benytte utstyr slik at vi minimerer behoved for tilgang til fysisk arealer	Trygg	
2	Ulykke med robotarm	1	Kun en person i nærheten av robot når den er i bruk. Benytte forhåndsdefinerte sikkerhetsprosedyrer. Bli bekjent med plassering av nødstop.	Trygg	
3	Frafall av ressurs personer	3	God og tydelig kommunikasjon mellom ressurspersoner gjennom prosjektarbeidet.	Trygg	
4	Sykdom/smitte hos gruppemedlem	1	Holde avstand. Når noen føler seg syk holder de seg hjemme og bruker verktøy for digitalt samarbeid. Ved mistanke koronasmitte avlegges det hurtigtest for alle gruppemedlmmer.	Trygg	
5	Defekt / ødelagt utstyr	2	Testing av utstyr tidlig for å avdekke feil. Være forsiktig og bevist ved håndtering av sårbart utstyr. Følge produsentens datablad og instruksjoner.	Trygg	
6	Konflikt i prosjektgruppen	3	Sosial aktiviteter for å fremme trivsel. Åpen og tydelig dialog under prosjektet. Jevnlige pauser under arbeidet. Om noen merker dårlig trivsel, oppretter vi et konfliktmøte .	Trygg	

Malen er utarbeidet av Arbeidstilsynet – september 2017.

Mateusz J, Viktor G, Trygg J

NTNU

Kunnskap for en bedre verden