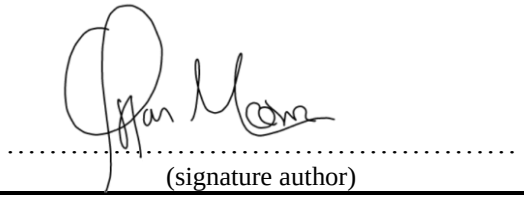
University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: Information Technology - Automation and Signal Processing	Spring semester, 2016 Open / ~~Confidential~~
Author: Ørjan Mæhre	 (signature author)
Instructor: Morten Mossige Supervisors: Ståle Freyer and Karl Skretting	
Title of Master's Thesis: Following Moving Objects Using Externally Guided Motion (EGM). Norwegian title: Følge objekter i bevegelse ved bruk av ‘’Externally Guided Motion’’ (EGM).	
ECTS: 30	
Subject headings: Externally Guided Motion (EGM), ABB, Cognex, In-Sight, Robotics, Object Tracking, Ramp	Pages: 80 + attachments/other: 18 + embedded file Stavanger, June 15/2016 Date/year

This page is intentionally left blank.

Abstract

This thesis presents a sensor guided system using Externally Guided Motion (EGM) to reduce the delay from new sensor data is available, to the robot initiates its movement. A suggested solution to follow a disc sliding down a ramp is presented. The position of the disc is registered by a camera and transmitted to a computer. The data is processed on the computer before it is forwarded to the robot. Without prediction the robot is approximately $200ms$ behind the disc. With prediction, the robot is able to follow the disc. EGM has low latency, enabling the robot follow an object moving with non-constant velocity. Even at velocities upwards to $2m/s$. Using EGM can give more effective robot cells and thereby reducing costs in the manufacturing industry.

This page is intentionally left blank.

Preface

This paper entails a master's thesis carried out at the Department of Electrical Engineering and Computer Science at the University of Stavanger. I would take this opportunity to thank my supervisors Morten Mossige, Ståle Freyer and Karl Skretting for valuable guidance and advice during this thesis. I would also like to thank my fellow students for five good years at the University of Stavanger.

This page is intentionally left blank.

Contents

List of Figures

List of Tables

This page is intentionally left blank.

Code Listings

This page is intentionally left blank.

Acronyms

ABB Ltd.	ABB robotics , Previously Asea Brown Boveri
C#	C-Sharp
EGM	Externally Guided Motion
FOV	Field of View
POM	Polyoxymethylene
ROI	Region Of Interest
RRI	Robot Reference Interface
RW	RobotWare
TCP	Tool Center Point

This page is intentionally left blank.

1. Introduction

Sensor guided robot paths become more and more usual. Where external sensors are used to generate position data for one or several robots. As an initiative to reduce the delay from when the robot recives new sensor data to the motion is started ABB Ltd. have developed a new module with RobotWare (RW) 6 called Externally Guided Motion (EGM). This module can read and write position data to the motion system at a high rate. The robot can fetch new sensor data every 4 ms and can start executing the movement within 20 ms[1]. High rate and low delay is the main advantage of EGM [1].

Reducing the time from new sensor data is available to the robot starts it motion will enable robots to do more in less time. This will give more effective production lines. More effective production can reduce costs in the manufacturing industry.

This thesis is trying to conclude if it is possible to follow an object moving in two dimensions using EGM. A plastic disc will be used as an object. To restrain the movement into two dimensions the disc will slide down a ramp. The velocity is high and non -constant. The motion will be over within 1-2 s. Experiments is conducted to test if EGM gives a quick enough response to follow the disc under the given conditions.

1.1 Related Work

Since EGM is a new module there is no known previous work and no known publications on EGM. There have been done work with sensor guided paths. A sensor guided system for repair of buoyancy module is presented in [2]. A vision system is used to mark defect regions in the buoyancy module. This generates paths for the robot to remove the defect areas.

In [3] a sensor guided system to be used in contour following is suggested. This system uses multiple sensor. Combining vision, force and acceleration for contour following tasks.

Luoa *et al.* [4] suggests an architecture for robotic seam tracing. They use off-the self sensors to compensate the residual errors. The system were able to track both linear and non-linear seams. TCP offset error were within ± 0.1 mm with speed equal to 100 mm/s.

[1] Depending on robot type and robot configuration.

1.2 Robot Cell at the University of Stavanger

The robot cell which the testing has been preformed consists of two Irb 140 ABB industrial robotic arms. Irb 140 is a six axis, small, multifunctional robotic arm. Maximum Tool Center Point (TCP) velcocity is 2.5 m/s. Maximum TCP acceleration is 20 m/s^2, with acceleration time 0-1 m/s equal to 0.15 s. Reach of 5th axis is 810 mm [5]. To be compatible with RW 6 the CPU in the control cabined had to be replaced.

1.3 Thesis Outline

Chapter 2 - Background:
Contains the theoretical foundation which this thesis is built on.

Chapter 3 - Using EGM to Follow Moving Objects:
Describes a suggested solution which enables an industrial robot arm with EGM implemented to track a disc sliding down a ramp.

Chapter 4 - Experiments, Results and Analysis:
Presents all experiments conducted during this thesis. This chapter also presents the results from these experiments and an analysis of the results.

Chapter 5 - Discussion:
Discussions around the implementation, results and problems which have occurred during this thesis.

Chapter 6 - Conclusion and Future Work:
Gives the thesis conclusion based on the results from the experiments. Future work is also presented.

Appendix A - Experiment with Constant Acceleration:
Contain remaining plots with results from the experiment conducted with constant acceleration.

Appendix B - Experiment with Constant Velocity:
Holds the remaining plots with results from the experiments conducted with constant velocity.

Appendix C - Code:
Gives an short introduction to the code which is embedded in this thesis.

2. Background

This chapter contains the theory used to perform the implementation and experiments. EGM, Cognex In-Sight 5400 and necessary physics are some of the subjects described in this chapter.

2.1 Externally Guided Motion (EGM)

This section is based on information found in [1]. EGM is a new module released by ABB with RW 6 and this module gives a better response to external sensor input [6]. EGM gives the possibility to adjust the the path of the robot within 10 - 20 ms[1] [1, p.313]. EGM goes right in to the motor reference generator, and by bypassing the path planner the robot is able react quicker to new sensor data. All filtering, supervision of reference and state handling is handled by EGM. In this section there will be given an introduction to EGM and the EGM sensor protocol for UdpUc communication. There is also possible to use analog communication between sensor and robot, but this will not be covered in this thesis. Interessted readers can find out more about how to set up analog communication in [1].

The reference can either be given as joint values or as pose. Using pose there is a need for reference frames. Measurements from the sensor and direction for position changes can only be given relative to reference frames. Using joint values there is no need for reference frames because both sensor values and position values are axis angles given in degrees relative to the calibration position of each axis [1]. The setup using joint values is quite equal to pose, except for the need of reference frames. Using joint values is covered in this thesis. Information on how to set up the system using joint values can be found in [1].

In RW 6.x the option EGM gives access to:

- Instructions which can be used to activate, set up and reset EGM.
- Instructions which enables the user to initiate EGM movements and stop them.
- Function which can retrieve current EGM state.
- System parameters to configure EGM and assign these parameters default values.

[1, p.314]

[1] Depending on the robot type

2.1.1 Basic Approach

With RW 6.x or higher EGM is already implemented in Robotstudio. Except for setting up communication with the sensor there is no extra work that has needs to be done to run EGM. Own instructions and functions to run EGM are embedded in RAPID.

An EGM movement has to start in a fine point. Therefore it is not possible to order parts of a movement to be conducted using EGM, e.g. have EGM to pop up if something unexpected should happen.

The EGM process can have three different states: EGM_STATE_DISCONNECTED, EGM_STATE_CONNECTED and EGM_STATE_RUNNING. The description of these three states is given in table 2.1. The table is taken from [1, p.315].

Value	**Description**
EGM_STATE_DISCONNECTED	The EGM state of the specific process is undefined. No setup is active.
EGM_STATE_CONNECTED	The specified EGM process is not activated. Setup has been made, but no EGM movement is active.
EGM_STATE_RUNNING	The specified EGM process is running. The EGM movement is active, i.e. the robot is moved.

Table 2.1: Different states of the EGM process [1, p.315].

A basic approach on how to set up EGM using a external device i.e. a sensor to give the target of the movement is shown in Table 2.2. This table is taken from [1, p.315].

To move between the different states a set of specific RAPID instructions are used. The transitions between the states can be found in Figure 2.1, this figure has been taken from [1, p.316]. EGM can be run in RAPID by using three different instructions . These three are: One "*setup*" instruction, one "*Act*" instruction and one "*Run*" instruction. Two of these instructions are shown in Figure 2.1, but one is left out. The "*Act*" instruction is where all the conditions of the movement is set, like which coordinate system the movement is related to, the convergence criteria, maximum admitted joint speed, etc. It is the "*Act*" instruction most of the choices regarding the EGM movement are defined.

	Action
1	Move the robot to a fine point.
2	Register an EGM client and get an EGM identity. This identity is then used to link setup, activation, movement, deactivation etc. to a certain EGM usage. The EGM state is still EGM_STATE_DISCONNECTED.
3	Call an EGM setup instruction to set up the position data source using UdpUc protocol connection. The EGM state changes to EGM_STATE_CONNECTED.
4	Give the position convergence criteria, i.e. when the position is considered to be reached.
5	Define which frames are used to define the target position and in which frame the movement is to be applied.
6	Give the stop mode, an optional time-out and perform the movement itself. Now the EGM state is EGM_STATE_RUNNING. This is when the robot is moving.
7	The EGM movement will stop when the position is considered to be reached, i.e. the convergence criteria is fulfilled. Now the EGM state has changed back to EGM_STATE_CONNECTED.

Table 2.2: Basic approach for setting up EGM using an external device to give the target of the movement. This table is taken from [1, p.315].

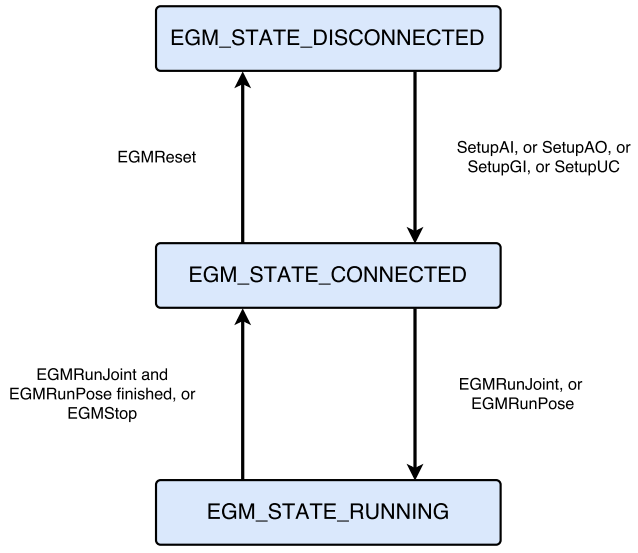


Figure 2.1: Transition between the states and which instructions to use to move from one state to another. This figure is taken from[1, p.316].

A block diagram representing a simplified view of the EGM control system is found in Figure 2.2. This figure is taken from [1, p.318]. There are two impor-

tant parameters that influence the EGM control loop, *Default proportional Position Gain* and *Default Low Pass Filter Bandwith Time* [1]. Position gain influences the responsiveness moving to the target which the sensor points at, in relation to the current robot position. The higher the value, the faster response. The low pass filter filters the speed contribution from EGM [1].

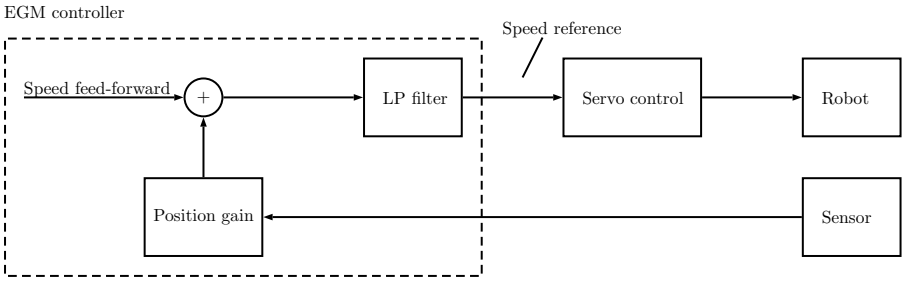


Figure 2.2: Block diagram of the EGM control system. This figure is taken from [1, p.318].

A block diagram of how the data flow is using the UdpUc protocol is found in Figure 2.3 and Table 2.3 explain the different steps. Both are taken from [1, p.317]. Something worth noticing is that in step 5. If the there is no new position data available, motion control continuous to use the latest position data written by EGM.

1	Motion control calls EGM
2	EGM reads feedback data from motion control
3	EGM sends feedback to the sensor
4	EGM checks the UDP queue for messages from the sensor
5	If there is a message, EGM reads the message and step 5 writes the position data to motion control. If no position data have been sent, motion control continues to use the latest position data previously written by EGM.

Table 2.3: Description of the different stages in the data flow using the UdpUc interface. This table is taken from [1, p.317].

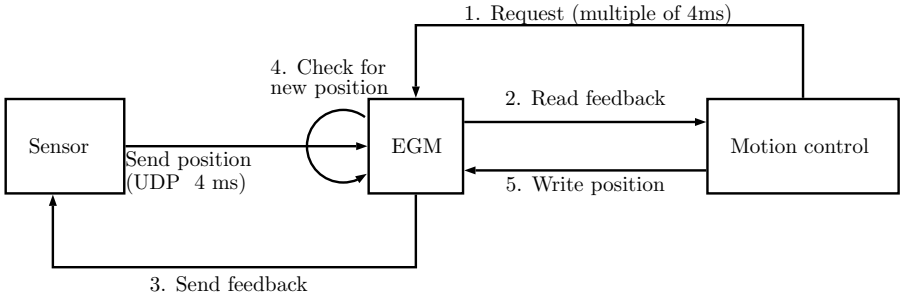


Figure 2.3: Block diagram representation of the data flow using the UdpUc interface. This figure is taken from [1, p.317].

2.1.2 Sensor Protocol

The protocol is designed for high speed communication between sensor and robot controller. The protocol is using *Google Protocol Buffers* for encoding and UDP as a transport protocol. The sensor protocol data structures is defined by the EGM proto file [1]. When this file is compiled it generates serialized/de-serialized code which can be used by the application. The robot controller and sensor uses a *server - client* setup to communicate, where the sensor acts like the server. This means that the robot controller has to initiate the communication. The sensor can not send anything before it has received a first message from the robot controller. After this first message data can flow independently of each other in both directions [1]. There is no special *connect / disconnect* message and there are no built in synchronization, request responses or supervision of lost messages. There is also no supervision over the queue of the receiver. This means that the sender will keep sending out data and it is the receivers job to make sure that the queue is emptied. The robot will send and receive data every four millisecond by default. This cycle time can be changed by using the optional argument `\SampleRate` of the RAPID instructions `EGMActJoint` or `EGMActPose`[1].

In the protocol there are two main data structures, `EGMRobot`, which is sent from the robot and `EGMSensor`, which is sent from the sensor. The header is the same for both data structures and is shown below in Listing 2.1[1].

```
1  message EgmHeader
2  {
3  optional uint32 seqno = 1; // Sequence number
4  optional unit32 tm = 2; // Time stamp in milliseconds
5
6  enum MessageType
7  {
8  MSGTYPE_UNDEFIND = 0;
9  MSGTYPE_COMMAND = 1; // For future use
10 MSGTYPE_DATA = 2; // Sent by robot controller
11 MSGTYPE_CORRECTION = 3; // Sent by sensor
12 }
13
14 optional MessageType mtype = 3 [default = MSGTYPE_UNDEFINED];
15 }
```

Listing 2.1: EgmHeader for the two main data structures, `EgmRobot` and `EgmSensor`[1].

The packet being sent from sensor to robot contains less data than the packet going the opposite way, See Listings 2.3 and 2.2. From these code snippets it clear

that the robot exchange more information about its state than the sensor. This is natural, since the sensor only tells the robot where to go, but it is good to know for the sensor what the position of the robot is, the next planned position and that everything is working properly.

```
header {
 seqno: 385
 tm: 4275119
 mtype: MSGTYPE_DATA
}
feedBack {
 joints {
  joints: -1.1714538335800171
  joints: 0.82521051168441772
  joints: 0.48566141724586487
  joints: -0.016466835513710976
  joints: 0.27424958348274231
  joints: -2.6035432815551758
 }
 cartesian {
  pos {
   x: 166.29362487792969
   y: -396.09942626953125
   z: 49.759349822998047
  }
  orient {
   u0: 0.00016568797582294792
   u1: -0.6623578667640686
   u2: 0.74915015697479248
   u3: 0.0074828267097473145
  }
 }
}
planned {
 joints {
  joints: -1.1378754377365112
  joints: 0.82992005348205566
  joints: 0.46713745594024658
  joints: -0.013733416795730591
  joints: 0.28822517395019531
  joints: -2.5726895332336426
 }
 cartesian {
  pos {
   x: 166.29362487792969
   y: -396.09942626953125
   z: 49.759349822998047
  }
  orient {
   u0: 0.00020525175204966217
   u1: -0.66237151622772217
   u2: 0.74913811683654785
   u3: 0.0074862390756607056
  }
 }
}
motorState {
 state: MOTORS_ON
}
mciState {
 state: MCI_RUNNING
}
mciConvergenceMet: false
testSignals {
 signals: 0
 signals: 0
 signals: 0
 signals: 0
 signals: 0
 signals: 0
}
rapidExecState {
 state: RAPID_RUNNING
}
```

Listing 2.2: Example of packet going from robot to sensor

```
header {
 seqno: 520
 tm: 278273743
 mtype: MSGTYPE_CORRECTION
}
planned {
 cartesian {
  pos {
   x: 250
   y: -400
   z: 50
  }
  orient {
   u0: 0
   u1: 0
   u2: 0
   u3: 0
  }
 }
}
```

Listing 2.3: Example of a packet going from sensor to robot

To configure the UDP - device the following parameters has to be in order; name, type, serial port, remote address and remote port number, see Table 2.4 These parameters are set in Robotstudio under **Controller - Configuration Editor - Transmission Protocol**.

Name	Type	Serial port	Remote address	Remote port number
EGMsensor	UDPUC	N/A	Computers IP-address	6510

Table 2.4: Example on how to set the parameters for UDPUC - communication [1].

2.1.3 Disadvantages

Since EGM by passes the path planner one can not expect a linear movement or order a movement to take a specific time or with a specific speed. Other limitations are:

- Can only be used on six axis robots.
- Can only be used in RAPID tasks with a robot, i.e. it is not possible to use it in a task that contains only additional axis, i.e. in robtargets there are values in the pose portion of the data.
- An EGM movement has to start in a fine point.
- For each robot a maximum of four external devices may be used to define position data. It is the important that each data component has one and only one source. This is not checked by EGM.
- It is not possible to perform a linear movement using EGM, due to the lack of interpolation functionality. The actual path of the robot will depend in the robot configuration, the start position, and the generated position data.
- EGM does not support MultiMove.

- It is not possible to use EGM to guide a mechanical unit in a moving work object.

These limitations are listed in [1, p.314].

2.2 Vision

As an external sensor a Cognex In-Sight 5400 smart camera is used to locate and produce the position data of the disc. This camera communicates with the PC through Ethernet and uses Udp as transport protocol.

2.2.1 Smart Camera

In-Sight 5400 is a smart camera produced by Cognex. The camera has a 640x480 resolution and can acquire new images with a frame rate up to 60 frames/second[2] [7]. This is a rigid camera with an IP67 grading, which means that the camera can be deployed in rough environments. What separates a Cognex smart camera from regular cameras is that all the image processing happens inside the camera. Instead of outputting an image, the camera can output a *string* with the position data. Cognex has its own library of finished algorithms that can be used, varying from object tracking algorithms to mathematical- and logic- tools [7].

Resolution:	640 x 480
IP -grading:	67
Color:	Greyscale
Frame rate:	60 fps

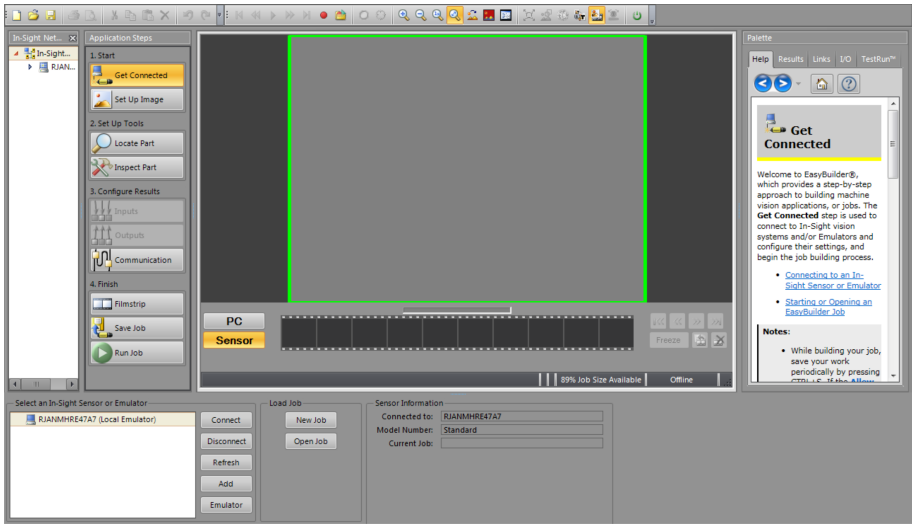
Table 2.5: Summary of Cognex 5400 specifications [8].

2.2.2 In-Sight Explorer

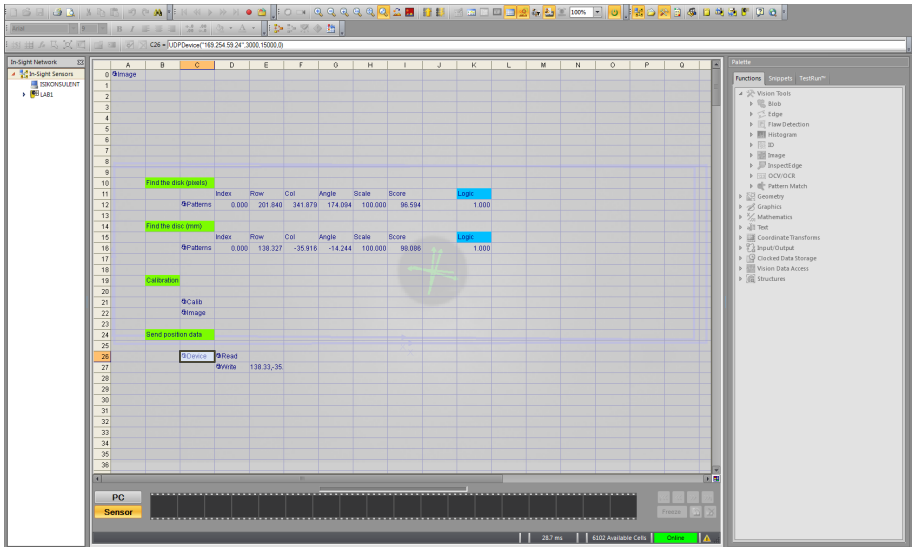
Cognex has its own GUI based programming software called *In-Sight Explorer*. The user can choose between two environments, *Easy builder*, a step by step environment and *Spreadsheet* which is a *Excel* like, drag and drop environment, see Figure 2.4. An Easy builder solution eventually get converted to spreadsheet code. In spreadsheet the user can drag and drop desired tools into empty cells and fill in the parameters specified for the particular tools.

Under the spreadsheet is the image produced by the camera. The spreadsheet can be hidden by pressing the spreadsheet button in tool bar. This enable the user to view the image captured with or without overlays. Typical overlays can be an axis cross on the object which is to be recognize or Region Of Interest (ROI). These

[2]Acquisition rate is based on minimum exposure, and full image frame capture.



(a) Easy Builder.



(b) Spreadsheet.

Figure 2.4: The two programming environments in In-Sight Explorer. *Easybuilder* is the simplest of the two where the camera can be programmed by following the steps under "Application steps". *Spreadsheet* is a more "drag and drop" environment where tools are dragged from the menu on the right and dropped into empty cells.

overlay can be hidden, this is done in the settings of the tool which produce the overlay, e.g. in the `Patterns` tool.

If a new camera gets connected to the network and the PC have not used this camera before it has to be added to the sensor network by In-Sight Explorer. Unless this is done the camera is not be displayed in In-Sight Explorer.

The camera is added by going into the **Add Sensor/Device to Network** option under the **System** menu. This will display a list over devices which are connected to the same network, but is not connected into the In-Sight Network. The user can here do some changes to the camera, such as name and IP-address.

Cameras already connected to the In-sight network can be found in the down left corner in Easy builder or in the left column in Spreadsheet, see Figure 2.4.

2.2.3 Edge Detection

Edges are detected by examining ROI for discontinuities in the pixel polarity. The ROI is scanned in a systematic manner, usually line by line starting at the top. Tracking one edge pixel to a neighbouring one and circumnavigating regions is called edge following. Binary images is required to do this [9]. A boundary is determinated by finding one pixel, thereafter the boundary is tracked by passing from one pixel to one of its neighbouring pixels until a junction or an endpoint is found. Tracking must then return to the start point and the boundary is tracked in the reversed direction [9]. The edge strength is measured, typical the gradient magnitude. A threshold value is used to decide if whether edges are present or not at an image point. To link edges together and avoid marking an edge at every noisy pixel thresholding with hysteresis can be used. This uses the assumption that edges are likely to be in a continuous curve. The upper threshold is used to find the start of an edge. From the start point the edge is traced, marking an edge when the value is over the lower threshold. The marking is stopped when the value of a pixel is below the lower threshold [10].

2.3 Physics

This section is based on information found in [11]. Different forces are acting on an object sliding down a ramp, see Figure 2.5. If the friction is constant throughout the motion the object will slide down the ramp with constant acceleration. The distance and velocity can than be found by applying the equations of motion for constant acceleration, see Equations 2.1, 2.2, 2.3 and 2.4[11]. Where v is velocity, v_0 is start velocity, a is acceleration, d is distance and t is time. These are to be used when the motion is in a straight line. In cases where the motion is in a 2D-plan the motions can be decomposed and the equations applied in each direction.

$$v = v_0 + a \cdot t \tag{2.1}$$

$$d = \frac{v_0 + v}{2} \cdot t \tag{2.2}$$

$$d = v_0 \cdot t + \frac{1}{2} a \cdot t^2 \tag{2.3}$$

$$v^2 - v_0^2 = 2 \cdot a \cdot d \tag{2.4}$$

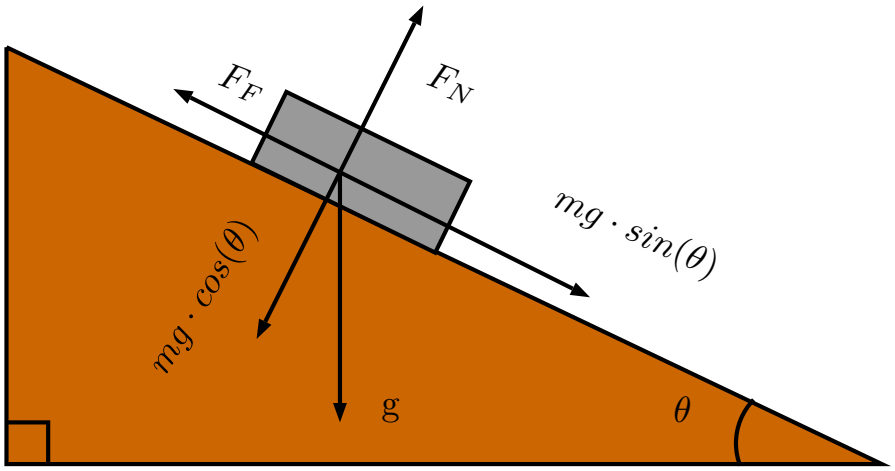


Figure 2.5: Forces acting on a object sliding down a ramp.

To find the Euclidean distance between two points in 2D, Equation 2.5 can be used. Δd is the combined distance in both x- and y-direction.

$$\Delta d = \sqrt{\Delta x^2 + \Delta y^2} \tag{2.5}$$

The velocity is found by differentiate the position with respect to time and the acceleration is the second differentiate of the position with respect to time. To find the velocity and acceleration between two points in 2D, Equations 2.6 and 2.7 can be used [11].

$$\Delta v = \sqrt{\Delta v_x^2 + \Delta {v_y}^2} \tag{2.6}$$

$$\Delta a = \sqrt{\Delta a_x^2 + \Delta {a_y}^2} \tag{2.7}$$

Friction force is defined by Equation 2.8, where μ_s is the friction coefficient between the object and the surface[11].

$$F_F = \mu_s mg(cos(\theta)) \tag{2.8}$$

2.4 Homogeneous Transformation

This section is based on information found in [12]. If the same point in space is viewed from two different coordinate systems it is necessary to transform the point to a common coordinate system. This is to be able to compare the two views, i.e the same point in space is viewed from the robots - and the cameras - coordinate system. This transformation is done by using rotation and translation matrices, see Equations 2.9 and 2.10. When rotating it is important to keep in mind the order in which the rotation matrices are multiplied, since this will affect the product [12].

$$
\begin{aligned}
R_{XYZ} &= R_{z,\phi} R_{y,\theta} R_{x,\psi} \\
&= \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\psi) & -sin(\psi) \\ 0 & sin(\psi) & cos(\psi) \end{bmatrix}
\end{aligned} \tag{2.9}
$$

The displacement vector is the displacement between the origin of the two coordinate systems in Euclidean space.

$$
d = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{2.10}
$$

Combining Equations 2.9 and 2.10 gives the homogeneous transformations matrix which is used to transform points from one coordinate system to another, see Equation 2.11.

$$
H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}; R \in SO(3), \quad d \in \mathbb{R}^3 \tag{2.11}
$$

3. Using EGM to Follow Moving Objects

This chapter describes how the system is built up to enable a robot with EGM implemented to follow a disc sliding down a ramp. The communication is set up like client-server communication, where the PC in both links acts like the server. All communication happens over Ethernet with UDP as the message protocol.

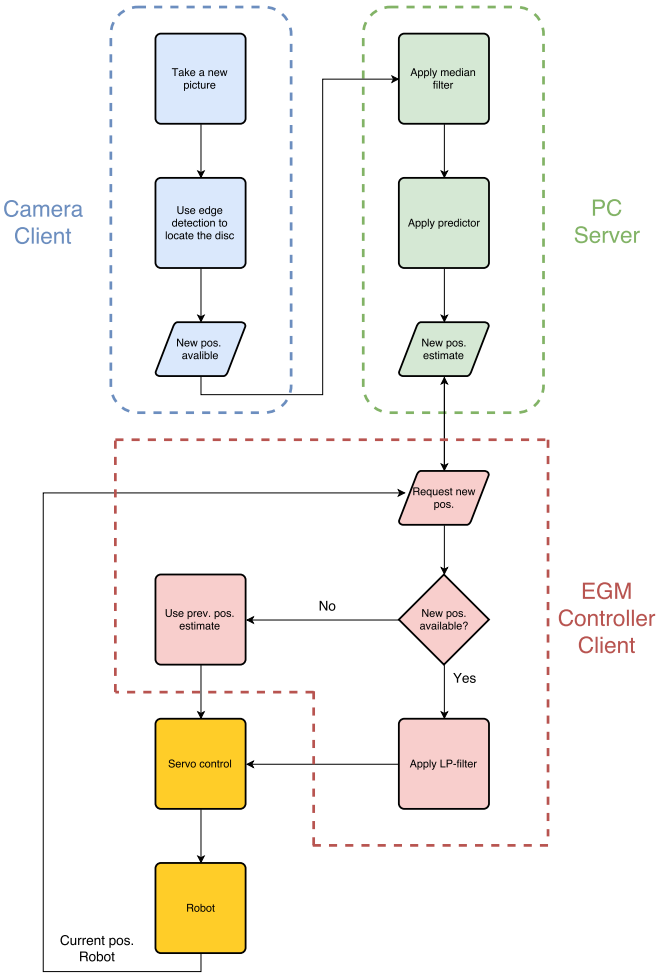


Figure 3.1: The flow of position data in the system. There is a server - client communication between camera - PC and PC - Egm controller.

3.1 Virtual Environment

The main focus in this section is how the solution is implemented in Robotstudio. This covers both the RAPID code and how the virtual environment is set up so to be used for simulations. RobotStudio 6.02 and RW 6.01 were used throughout the whole thesis.

3.1.1 Work Station

The virtual environment is set up like the lab at the University of Stavanger. With two IRB 140 ABB robotic arms and a conveyor placed between them. The ramp is not implemented, and there is no disc. The basis of this environment is made by Karl Skretting. There are done some small changes in this thesis.

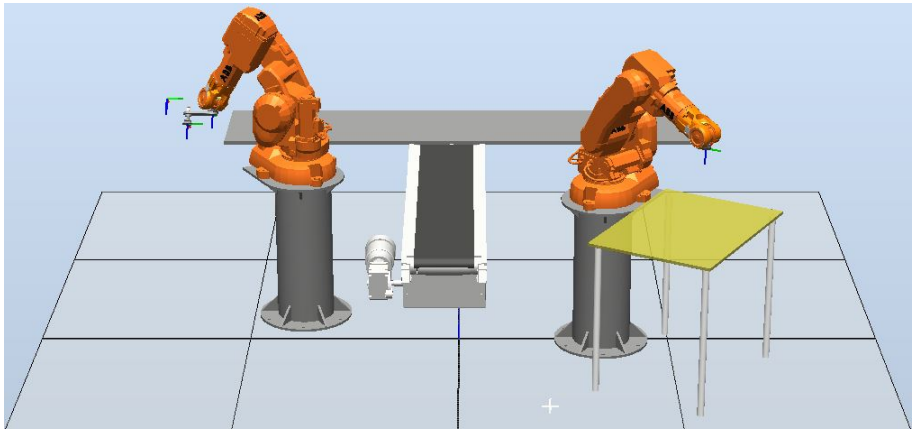


Figure 3.2: The virtual environment is setup to be as alike to the physical environment as possible.

3.1.2 RAPID

In RAPID there is a program which sets up EGM and asks for new positions every X ms.[1] To enable the same rotation on all `robtargets`, `workobjects` and `poses`, `OrientZYX` is used. This takes Euler angles as input and outputs an orient type variable which is expressed in the form of quaternion. The x-axis is located along the ramp, y-axis is across the ramp and z-axis is orthogonal to the ramp, see Figure 3.3. The coordinates for these data types are defined in world coordninates[2].

The workobjects x- and y-axis have the same crossing as the cameras coordinate system. The z axis has its origin placed above the ramp. This means that the robot moves in a plane above and parallel to the ramp. `Pose` is used, this means that there is need for a reference system which all movements are relative to. The `pose` data type is named `posecorTopOfRamp` and has the same origin and rotation as the workobject.

[1] X is equal to the processing time of the camera. This is set manually in `Samplerate` under `EgmRunPose` and depend on the camera setup.

[2] Which has its origin in the robots base.

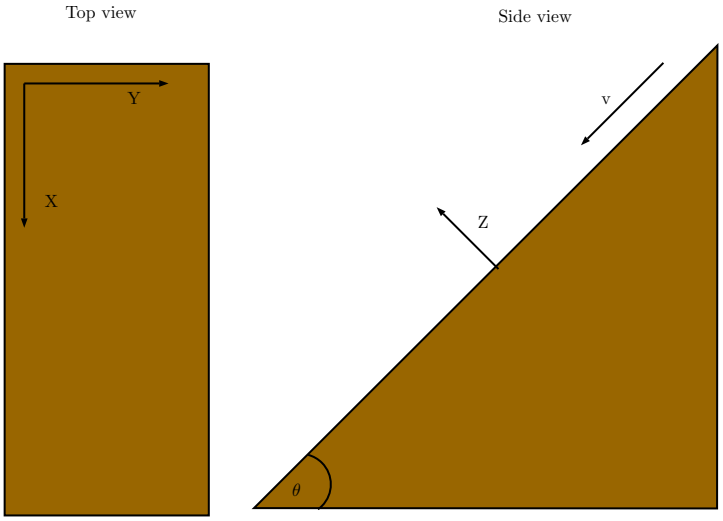


Figure 3.3: Overview of the simulated ramp is set up.

The robot is moved to a `fine` point, named `startPoint`. When the robot reaches this point, if EGM is not connected, the program sets up EGM using `EGMSetupUC`. Then EGM is run by using `EGMActPose` and `EGMRunPose`, see Listings 3.1 and 3.2.

```
1 EGMActPose egmID1\Tool:=tSuctionCup\WObj:=TopOfRamp,posecorTopOfRamp,
2 EGM_FRAME_WORLD, posecorTopOfRamp, EGM_FRAME_WORLD
3 \x:=egm_minmax_lin1 \y:=egm_minmax_lin1 \z:=egm_minmax_lin1
4 \rx:=egm_minmax_rot1 \ry:=egm_minmax_rot1 \rz:=egm_minmax_rot1
5 \LpFilter:=3\Samplerate:=12\MaxSpeedDeviation:= 5000;
```

Listing 3.1: `EGMActPose`

Some relevant information in Listing 3.1 are:

- Both poses are in relation to world coordinates.
- Convergence criteria, `emg_minmax_lin1` = [-1,1] and `egm_minmax_rot1` = [-2,2] is set earlier in the program.
- The low-pass filters bandwidth are set to $3Hz$.
- The robot should request new position data every 12th ms (`Samplerate`)
- Maximum admitted joint speed change is set to 5000 degrees/second.[3]

The bandwidth of the low-pass filter is set to a small value to avoid "Error: 50375: Dynamic load too high" which occurs when the speed reference changes too fast. Using a smaller bandwidth give a smoother transition from high speed to immediate stop and vice versa. If the bandwidth is set too small this will give a overshoot. Through experiments $3Hz$ were found to be a reasonable value which eliminate the error and gives a minimum of overshoot.

[3]This is set to a random high value, because trimming the acceleration is not desired

```
1 EGMRunPose egmID1, EGM_STOP_HOLD\x \y \z\CondTime:=20
2  \RampInTime:=0.05 \RampOutTime:=0.5;
```

Listing 3.2: `EGMRunPose`

In `EGMRunPose` the `CondTime` is set to 20 s, which means that if the convergence criteria is not within this time the program will continue to the next instruction. This is never a issue in this program since the motion is done within a few seconds. The `RampInTime` is how fast the movement is started in seconds and `RampOutTime` has no meaning since `Mode` is set to `EGM_STOP_HOLD` [1].

3.2 Real Time Sensor Data Processing

All code is written in C-Sharp (C#), which is a programming language in the .Net family. Several different classes are made and they are presented in separate sections below. All the code is placed under the solution `ExternalGuidedMotion`.

3.2.1 Structuring of the Data, (`EGM.cs`)

This class is auto generated by *ProtoGen*. Protobuf -csharp binaries can be downloaded from [13]. The zip-file should be unpacked and placed in a suitable directory. Windows console is opened in the tools directory and the command in Listing 3.3 is written in the console window. This generated an EGM C# (`Egm.cs`) file from the `egm.proto` file.

```
protogen .\egm\egm.proto --proto_path=.\egm
```

Listing 3.3: Command to generate the EGM C# file

From the NuGet packet manager protobuf -csharp and Google.ProtocolBuffers is installed in the solution.

The egm.proto file can be obtained from the PC or the IRC5 controller.

- **On the PC where RobotWare is installed:**
  %LocalAppData%\ABB Industrial IT\Robotics IT\RobotWare\
  <RobotWare_xx.xx.xxxx> utility\Template\EGM\
- **In the IRC5 Controller:**
  <SystemName>\PRODUCTS\<RobotWare_xx.xx.xxxx>\utility\
  Template\EGM\

Egm.proto defines the data contract between the sensor and the robot [1].

3.2.2 Choosing which Position Data to use, (`Program.cs`)

This is where the `Main` method is located. The arguments in `Main` is used to decide which class to be used to generate the position data, `Camera.cs` or `SimDisc.cs`.

"Camera" is given as an argument to use the camera and to use the simulated disc "Simulate" is given. The argument given decide what `mode` is set to and this sets which methods to be run later in the program.

3.2.3 Receiving Data From the Camera, (`Camera.cs`)

This class takes in the x- and y- position of the disc. Data packets from the camera are sent as `strings`. This class convert the data into `double` to be able to do mathematical operations. The x- and -y positions are filtered using a *Median filter*. This filters out noise peaks which may be present due to false accepts in the camera. Filtered values are stored in `X` and `Y` respectively. The class calls the `NewPrediction` method in the `Predictor` class with `X` and `TimeStamp` as parameters. All this happens in its own thread to minimize delay.

3.2.4 Predicting Future Positions of the Disc, (`Predictor.cs`)

This predictor uses previous readings to predict future positions. The current velocity is found by taking the displacement between two consecutive samples, divided by the change in time, see Equation 3.1. To find the acceleration it would be possible to derive the velocity, but this were found to give unnecessary noise. Therefore a model of the acceleration is used, see Equation 3.2. Were θ is the angle of the ramp in radians, g is the gravitational acceleration in mm/ms^2 and μ is the frictional coefficient.

$$v = \frac{x_i - x_{i-1}}{\Delta t} \tag{3.1}$$

$$a = g \cdot sin(\theta) - \mu \cdot g \cdot cos(\theta) \tag{3.2}$$

From the experiments done in Chapter 4 the delay where found to be between 200 and 300 ms. The predictor therefore has to predict the discs position 200 - 300 ms ahead in time. The predicted value is found by using Equation 3.3. Where x is current position in mm, v is velocity in mm/ms, t is the time step in ms, c is a constant, a is the acceleration in mm/ms^2 and $\hat{x}$ is the predicted value which is stored in the variable `PredictedPosition`. The time step is the time between two consecutive samples. The predictor must predict the position around 250 ms ahead in time. Since the sampling frequency is about 12 ms[4], c is set to a constant between 20 and 30. The frictional coefficient, μ is set equal to 0.3. This value where found trough the experiments in Chapter 4 and confirmed by [14] where 0.3 is in the middle scope of the friction coefficient for wood - wood and plastic - plastic.

$$\hat{x} = x + v \cdot t \cdot c + \frac{1}{2} a \cdot (t \cdot c)^2 \tag{3.3}$$

[4]The time between readings depend on where the camera is placed, light conditions and other factors. Therefore this may vary from experiment to experiment.

3.2.5 Constructing Data Packets, (`Sensor.cs`)

The skeleton of this class is made by ABB Ltd. in `EgmSensorApp.cpp` which is given as an example application to EGM. This class constructs the data packets which is being sent to the robots motion controller. It also handles the data packets coming from the robot. Since there are two different places where position data can be generated, this class uses the argument in `Main` to decide if the position data is coming from the camera or the simulated disc. Further in this section all data comes from the camera. There is a own thread, `SensorThread`, which sets up the communication between PC and robot. This thread handles the requests from the robot and updates the Cartesian coordinates to the latest predicted position estimate. When the robots current position is received the latest position estimates is sent to the robot. Estimates are updated by the method `CameraSetPos`. This method fetch the predicted x-position from `PredictedPosition` in the `Predictor` class. The y-position comes from `Camera.Y`. These values are used to set the Cartesian position in the `CreateSensorMessage` method. This is were packet being sent to the robot is created. The data packet is sent to the robot using `UdpClient` which is a class in the .Net framework.

3.3 Vision

The vision program is created using the spreadsheet environment in In-Sight Explorer, see figure 3.4. In the image cell (which is located by default in `A0`) the parameters is set to restrict the Field of View (FOV) of the camera to only cover the ramp. Restricting the FOV gives faster recognition, reduces noise and false accepts. Only by restricting the FOV the processing time of the camera went down from $16ms$ to about $12ms$.[5] The `Expose time` is also set to a minimum to keep the processing time down to a minimum. Depending on the light conditions in the lab the expose time was set to 1 or $2ms$.

To recognise the disc, an edge detection tool located in `Patterns` is used. The calibrated image transforms pixels into mm, this is done by the `Calib` tool located in `C16`. The `calib` tool is set up to use a chequerboard with $20mm$ spacing without a fiducial, see Figure 3.5. Four poses are used to get an accurate calibration along the whole ramp. The origin is the top right cross seen in Figure 3.5a. The x-axis is defined along, the y-axis is across and the z-axis is perpendicular to the ramp.

`Device` located in `C21` sets up the UDP communication between the camera and the PC. `Host Name` is the computers IP-address, `Port` is which port to use, in this case it is set to 3000. `Timeout` is set to a high value, because it is not desired that the camera times out if there is no communication. `Binary Packet` is left unchecked.

`Read` in `D21` points to the `Device` structure in `C21`. The `Write` cell points to the image in `A0`, the `device` structure in `C21`, and the to the `data` structure in `E22`. The data structure in `E22` is the position in both x-and y-direction, timestamps and sequence number in a `String` format. It is this data which is being sent to the PC. The timestamps is the processing time of the camera. This is used in the predictor

[5] Depending on the setup

and for debugging. Sequence number is only used for debugging. The position is in mm and the timestamps are in ms, therefore the data is restricted to zero decimal points since mm and ms gives an acceptable precision.

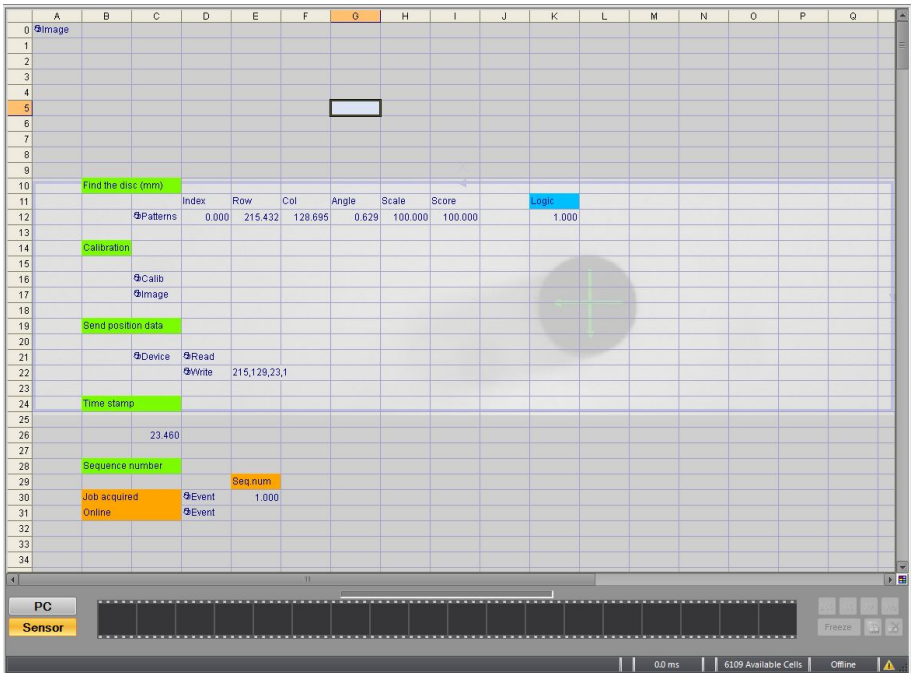
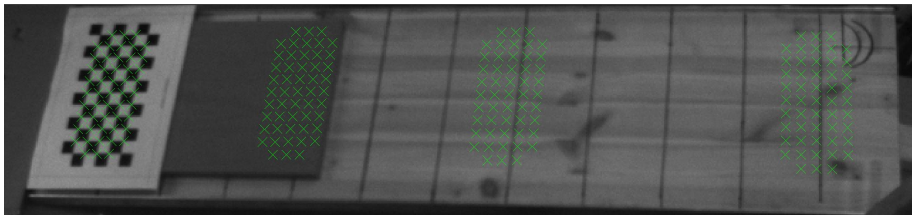
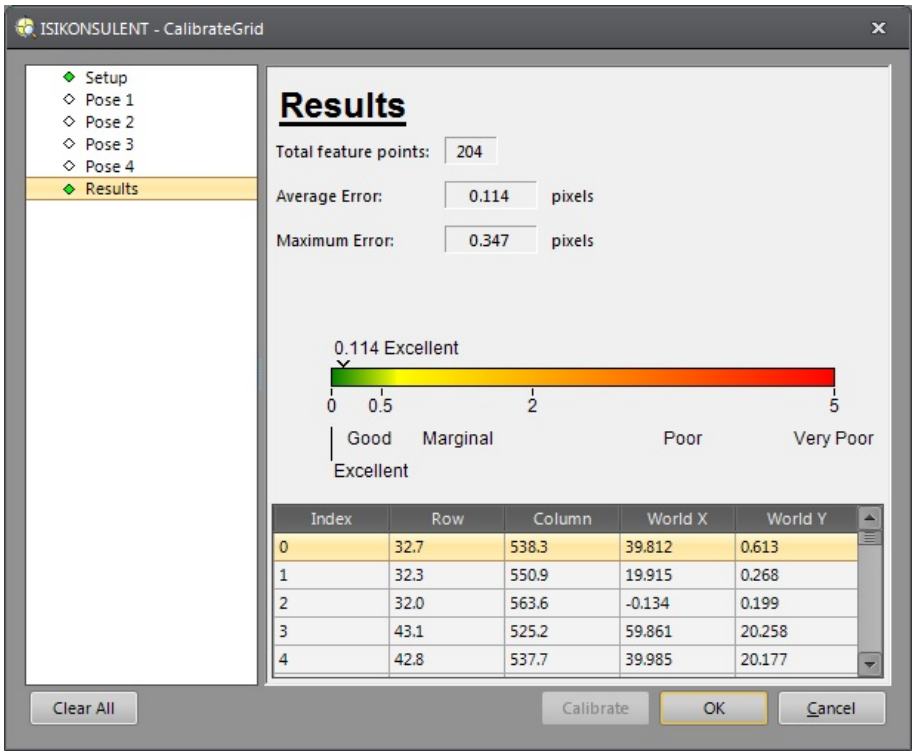


Figure 3.4: The spreadsheet program used to detect the disc. Behind the spreadsheet live image of what is being recorded is found. In this image it is possible to see the disc behind the spreadsheet, with an overhead displaying the axis cross for the disc.

There are two `events` locate in `D30` and `D31`. `D30` is used to update the sequence number for each iteration. `D31` is used to reset the sequence number when the camera goes from online to offline. So that each session starts with sequence number 0.



(a) Calibrated image.



(b) Result after calibrating.

Figure 3.5: Calibration environment after calibrating. After calibrating In-Sight Explorer gives a score on how good the calibration was. In this case the score was 0.114 and this is defined as "Excellent".

3.4 Offline Processing

A Matlab script was created to do offline processing. This script uses the position and time data from the *.txt*-files generated by the C# program. Processing this data creates plots of the position, velocity and acceleration with respect to time. The position data from the robot is in relation to the base, and the position data of the disc is related to the workobject. This script transforms the position data to a common coordinate system, so the data from the two sources can be compared. This transformation is done by rotation and translation matrices.

The distance that the disc and has travelled is found by using the Algorithm 1 (based on Equation 2.5).

Algorithm 1 How to calculate distance in a 3D-space.

for i = 1: Step = 1: i $\leq$ length position of data **do**
distance[i] = $\sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2 + (z_0 - z_i)^2}$
end for

The velocity is found from the derivative of the position. First the derivative is found in all three axis, and then to find the total velocity an algorithm similar to Algorithm 1 is used.

And last the acceleration is found by using the second derivative of the position. As for the velocity a similar algorithm to Algorithm 1 is used to find the combined acceleration.

With these data in place it is possible compare travelled distance, velocity and acceleration for both disc and robot.

This page is intentionally left blank.

4. Experiments, Results and Analysis

This chapter describes the setup for all experiments conducted during this thesis. Results and an analysis of the results are also given. All experiments were conducted in the robotics lab at the University of Stavanger, see Section 1.2.

4.1 Tracing Capability of the Robot

These two experiment were done to test the tracing capability of the robot. This was to test how far the robot was behind the disc with constant velocity and constant acceleration. Results from these experiments gave a pinpoint on how far ahead the predictor had to predict.

4.1.1 Constant Acceleration

The purpose of this experiment was to see how well the robot could follow the path of a simulated disc. The disc moved with constant acceleration. Position data for both disc and robot were stored in a *.txt*- file for offline processing. This experiment was conducted to try and answer the following questions:

- Can the robot follow an object with non- constant velocity?
- Is the difference in the robot and disc's position constant throughout the motion?
- Will the difference in the robot and disc's position be different with high acceleration versus low acceleration?

Difference in position means how long after the disc have travelled a certain distance have the robot travelled the same distance, or visa versa.

Setup

The disc was created in C#. This disc slides frictionless down a ramp according to the laws of physics, see Equations 4.1, 4.3 and 4.2. The robot tried to copy the motion of the disc. The robots start point was [496, -526, 650].[1]. The disc moved

[1] In world coordinates

in the work objects x-direction. The movement was started by pressing "s" in the console window. This started the timer which was used to update the x-position of the disc and assigned the y-position to a constant equal $-125mm$. Before "s" were typed in the position data were set to [0,0,0] which gave the origin of the work object as the target.

$$a = g \cdot sin(\theta) \tag{4.1}$$

Where g is the gravitational force and θ is the angle of the ramp.

$$v = a \cdot t \tag{4.2}$$

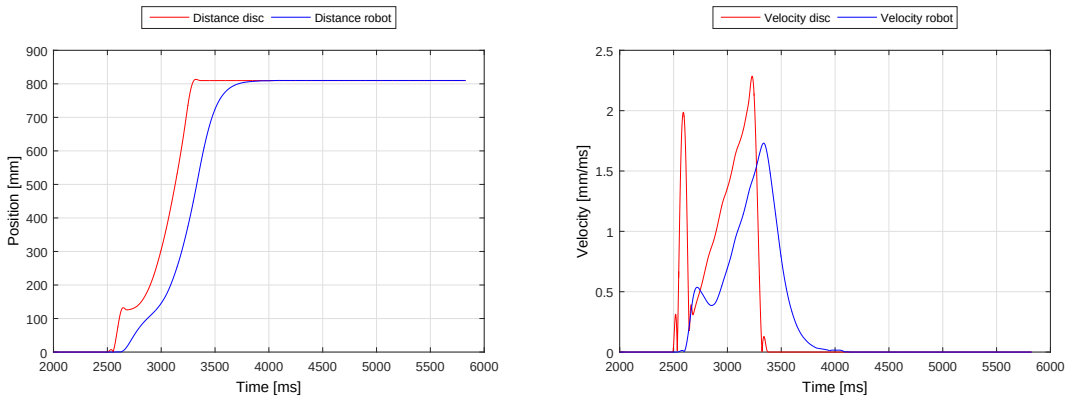
$$s = \frac{1}{2} \cdot v \cdot t \tag{4.3}$$

Five different angles were tested, 20, 30, 40, 50 and 60 degrees. Distance, velocity and acceleration were calculated using Algorithm 1, found in Section 3.4.

Results

This section presents the results from the experiments done with 20 degrees and 60 degrees. Plots from the remaining experiments can be found in Appendix A. This appendix also holds plots displaying the movement and velocity in all three directions. A summary of the results can be found in Tables 4.1 and 4.2.

By setting the simulated ramps angle to 20 degrees, the theoretical acceleration for the disc is $3.36 \cdot 10^{-3} mm/ms^2$, which gives a top velocity equal to $2.78m/s$. The whole motion took $828ms$. Measured top velocity for the disc was $2.29m/s$. The disc had a higher top velocity than the robot, the difference was $0.56m/s$, see Figure 4.1b. The disc pulled away from the robot during the motion, but not by much. At distance equal $270mm$ the robot was $190ms$ behind the disc. At distance equal $540mm$ the difference was $201ms$, see Figure 4.1a.

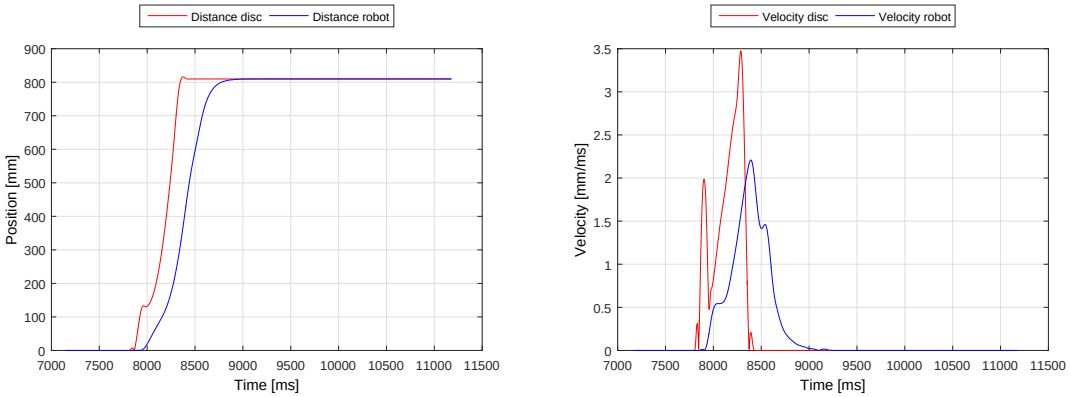


(a) Distance.

(b) Velocity.

Figure 4.1: Robot and discs distance and velocity. 20 degrees angle on the simulated ramp.

Angle equal to 60 degrees will give a theoretical acceleration equal $8.50{\cdot}10^{-3} mm/ms^2$, which gives a top velocity equal to $4.84 m/s$. The whole motion was over after $569 ms$. Measured top velocity for the disc was $3.48 m/s$, and for the robot $2.21 m/s$, see Figure 4.2b. Difference in top velocity was $1.27 m/s$. During the motion the disc pulled away from the robot. At distance equal $270 mm$ the difference in time was $183 ms$ and at $540 mm$ the robot was $209 ms$ behind the disc, see Figure 4.2a.



(a) Distance.

(b) Velocity.

Figure 4.2: Robot and discs distance and velocity. 60 degrees angle on the simulated ramp.

Angle	270 [mm]	540 [mm]
20	190 [ms]	201 [ms]
30	189 [ms]	203 [ms]
40	186 [ms]	203 [ms]
50	184 [ms]	205 [ms]
60	183 [ms]	209 [ms]

Table 4.1: Difference in time between the disc and robot at two given points, when the distance travelled was equal to $270 mm$ and $540 mm$

Analysis

The peaks at the start of the velocity plots were because the y- position was set to $-125 mm$. Difference in theoretical and measured top velocity can be due to the smoothing of the plots. Example of unfiltered plots are found in Appendix A. The robot is able to follow the disc. Distance between disc and robot were not constant. From the numbers it looks like the acceleration affected how far behind the robot was. Already at 30 degrees the discs top velocity were higher than the robots max TCP velocity. This is why the robot fell farther and farther behind the disc as the ramps angle was increased.

	Top velocity	
Angle	**Disc**	**Robot**
20	2.29 [m/s]	1.73 [m/s]
30	2.70 [m/s]	1.90 [m/s]
40	3.07 [m/s]	2.06 [m/s]
50	3.27 [m/s]	2.14 [m/s]
60	3.48 [m/s]	2.21 [m/s]

Table 4.2: Top velocity for the disc and robot.

4.1.2 Constant Velocity

The purpose of this experiment was to see if the robot could follow an object moving with constant velocity. The experiment was conducted to try and answer the following questions:

- Can the robot follow an object moving with constant velocity?
- Is the difference in the discs and robots position constant throughout the motion?
- Does different velocities influence the distance between disc and robot?
- What is the processing time of the camera? And is this affected by the objects velocity?

Setup

This test were done with the setup seen in Figure 4.3. The robot on the right had a disc mounted to its tool. This robot moved linear and with constant velocity over the white area seen in Figure 4.3. This robot was programmed to move to a point, wait for $2s$ before returning to the start point with the same velocity. Looking down on the white area was a Cognex 5400 smart camera. This camera was set up to locate the disc and send its position over to a program in C#. The PC forwarded the position data to the robot on the left. This robot copied the discs motion. The disc was set up to move only in the x-direction (there were some movement in the y-direction as well due to inaccuracy in the calibration). There were no movement in the z-direction of the disc. Six different velocities were tested, from $100mm/s$ to $2500mm/s$.

The camera used an edge detection algorithm to locate the disc. This algorithm is built into the Cognex In-Sight Explorer software. A detailed description on how the camera was programmed is found in Section 2.2.2.

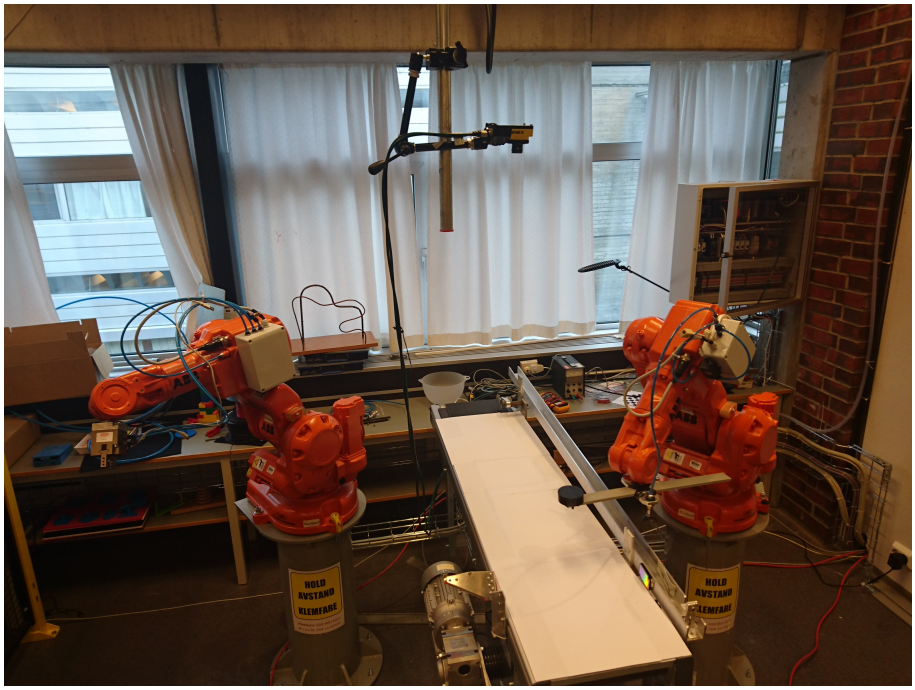


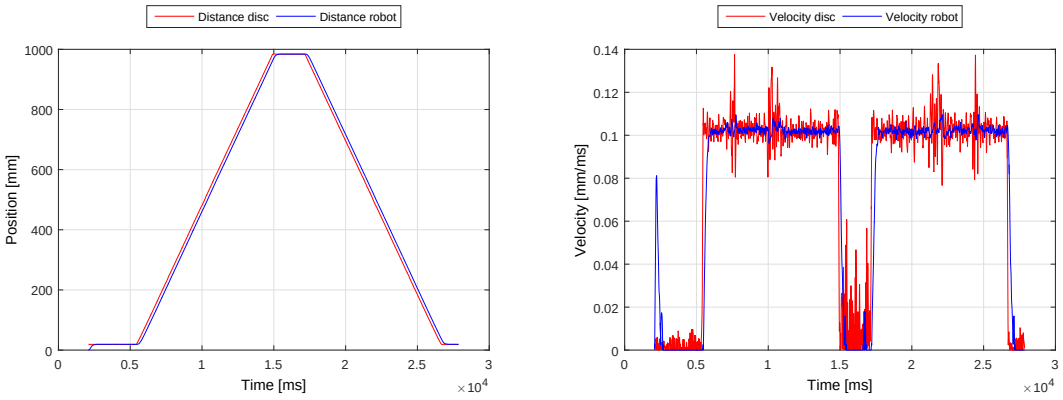
Figure 4.3: Physical setup experiment with constant velocity. The robot on the right moved the disc linear and with constant velocity in the FOV of the camera. The robot on the left copied the discs motion.

Results

The plots presented in this section is for the velocity equal $100mm/s$, $1000mm/s$ and $2500mm/s$. The remaining plots are found in Appendix B. Table 4.3 summarizes the results for all velocities tested.

In all results there were a lot of noise. Reason for this noise is discussed in Section 5.2. All data presented is smoothed using `smooth` in Matlab.

When the discs velocity was $100mm/s$ the robot was approximately $200ms$ behind the disc, except for in the acceleration and deceleration phase, see Figure 4.4. How far behind the robot was compared to the disc was approximately constant throughout the motion. When distance travelled was equal to $300mm$ the robot was $209ms$ behind the disc and at $700mm$ the difference was $210ms$, see Figure 4.4. The cameras processing time was not constant. 1032 out of 3461 samples the camera used $9ms$ and the remaining 2429 samples the camera used $10ms$, see Figure 4.5.



(a) Distance. (b) Velocity.

Figure 4.4: Distance and velocity with the disc travelling at $100mm/s$.

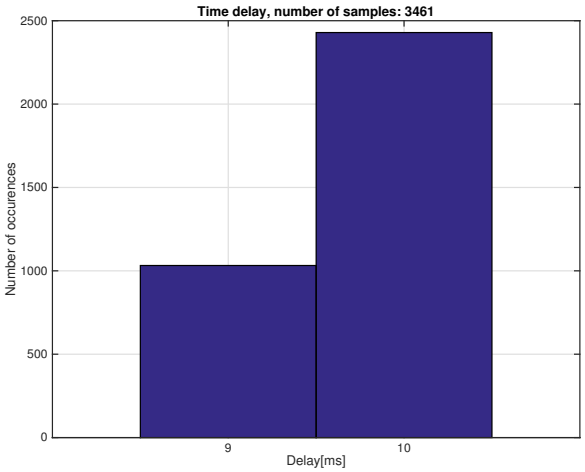
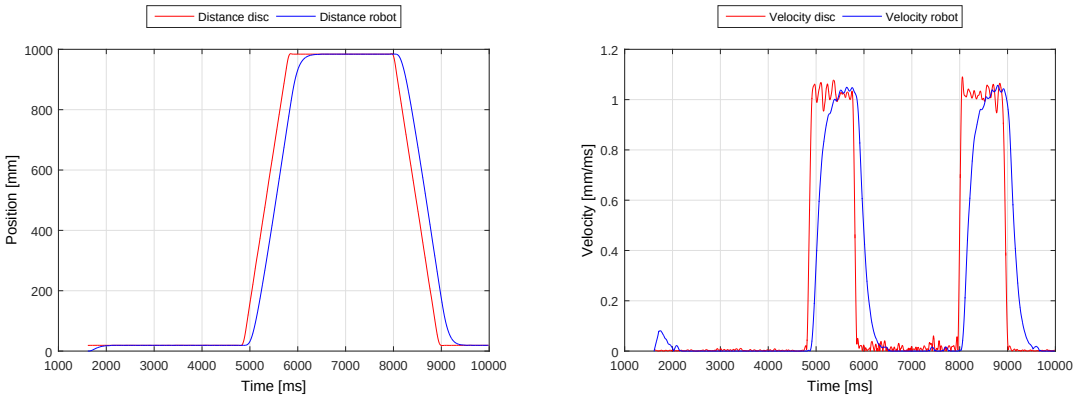


Figure 4.5: Processing time camera, v100.

With disc velocity equal to $1000mm/s$ the robot was approximately $200ms$ behind the disc throughout the motion, except during acceleration and deceleration. This difference was constant. When distance travelled was $300mm$ the robot was $208ms$ behind the disc and at $700mm$ the difference was $207ms$, see Figure 4.6. The cameras processing time was not constant. 344 out of 1246 samples the camera used $9ms$ and the remaining 902 samples it used $10ms$, see Figure 4.7.



(a) Distance. (b) Velocity.

Figure 4.6: Distance and velocity with the disc travelling at $1000mm/s$.

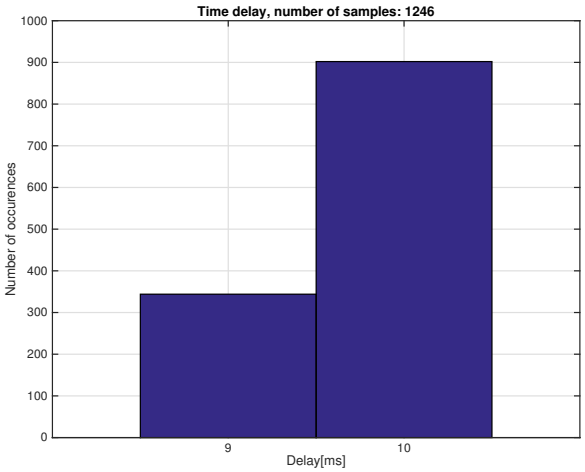
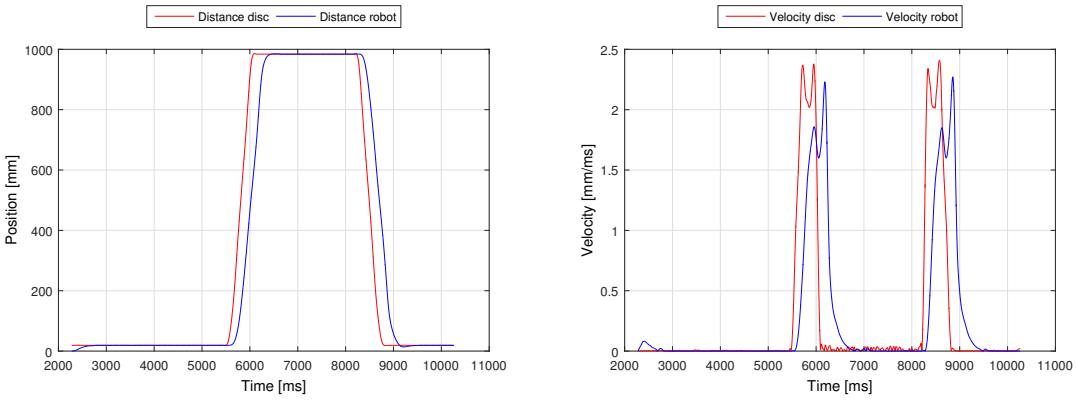


Figure 4.7: Processing time camera, v1000.

With the discs velocity equal to $2500mm/s$, which is the top velocity of the robot [5], the difference from when the disc had travelled a certain distance until the robot had travelled the same distance was not constant. At $300mm$ the robot was $175ms$ behind and at $700mm$ the difference was $233ms$, see Figure 4.8. The processing time of the camera was not constant. 281 out of 1274 samples the camera used $9ms$ and 993 samples $10ms$, see Figure 4.9.



(a) Distance. (b) Velocity.

Figure 4.8: Distance and velocity with the disc travelling at $2500mm/s$.

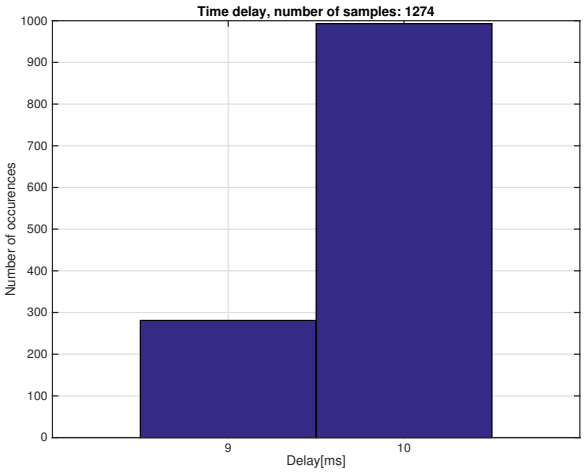


Figure 4.9: Processing time camera, v2500.

Velocity	**300 [mm]**	**700 [mm]**
100 [mm/s]	209 [ms]	210 [ms]
500 [mm/s]	210 [ms]	210 [ms]
1000 [mm/s]	208 [ms]	207 [ms]
1500 [mm/s]	200 [ms]	208 [ms]
2000 [mm/s]	179 [ms]	227 [ms]
2500 [mm/s]	175 [ms]	233 [ms]

Table 4.3: Difference in time from the disc was in a certain point until the robot reached the same point.

Analysis

The time from the disc was in a certain point to the robot reached that same point was dependant of the discs velocity. Higher velocity meant that the robot was further behind the disc compared with lower velocities. There was a lot of noise in the samples of the discs position. This made it impossible to use unfiltered data. There may therefore be some inaccuracy in the numbers presented due to smoothing. The robot was able to follow an object travelling with constant velocity as long at the velocity was under $2000mm/s$. The difference in the discs and robots position was constant through out the motion with velocity $\leq 1500mm/s$.

The cameras processing time was not constant varying with $\pm 1ms$. The cameras processing time was not affected by the discs velocity. Approximately 1/3 of the times the camera used $9ms$ and the remaining times it used $10ms$.

4.2 Different Sample Rate

In RAPID it is possible to set how often the robot should request new position estimates from the PC. This value is set as a parameter in `EgmActPose` and has to be a multiple of 4. To find out this affect the delay in the system, two different values were tested. `Samplerate:=4`, since this is the smallest allowed value and `Samplerate:=12`, since this is the multiple of 4 closest to the cameras processing time.

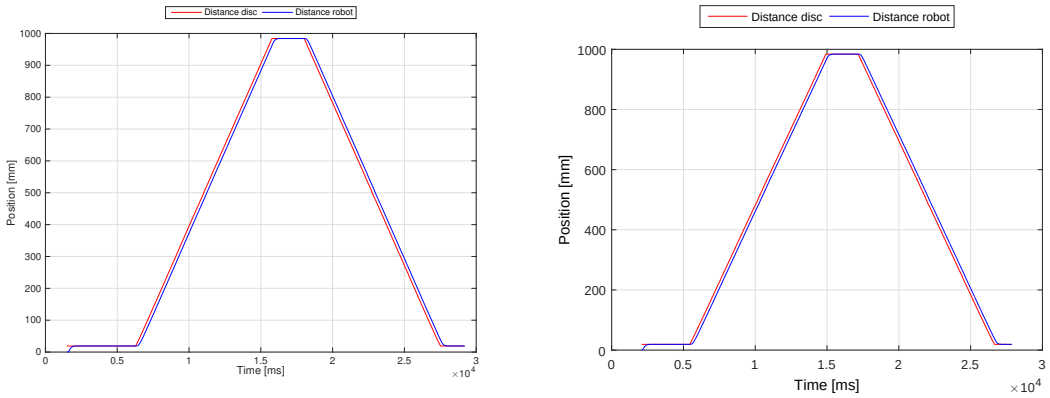
4.2.1 Setup

The setup is identical to the setup in Section 4.1.2, only difference is the value of the parameter `Samplerate` in RAPID.

4.2.2 Results

In Section 4.1.2 `Samplerate:=4` was used. Therefore the numbers in this section was the same. Only results using `Samplerate:=12` and the difference between the two sample rates is presented in this section.

When the discs velocity was $100mm/s$ the robot was $213ms$ behind the disc when distance travelled was $300mm$. When distance travelled was $700mm$ the robot was $210ms$ behind, see Figure 4.10. This means that the difference in using `Samplerate:=4` and `Samplerate:=12` was $4ms$ at $300ms$ and $0ms$ at $700mm$.

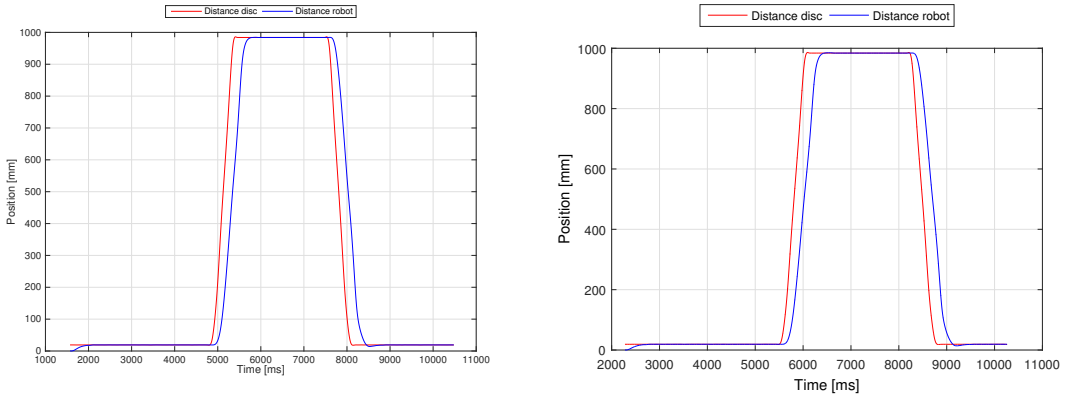


(a) `Samplerate:=12`. (b) `Samplerate:=4`.

Figure 4.10: Distance travelled by robot and disc with two different sample rates, disc velocity was $100mm/s$

With disc velocity equal $2500mm/s$ the robot was $196ms$ behind the disc when distance travelled was $300mm$. When distance travelled was $700mm$ the robot were $238ms$ behind, see Figure 4.11. This means that the difference between `Samplerate:=4` and `Samplerate:=12` with disc velocity equal $2500mm/s$ was $21ms$ at $300mm$ and $5ms$ at $700mm$.

Results from all velocities tested are found in Table 4.4.



(a) `Samplerate:=12` (b) `Samplerate:=4`

Figure 4.11: Distance travelled by robot and disc with two different sample rates, disc velocity was $2500mm/s$.

Velocity	300 [mm]	700 [mm]
100 [mm/s]	4 [ms]	0 [ms]
500 [mm/s]	5 [ms]	4 [ms]
1000 [mm/s]	5 [ms]	6 [ms]
1500 [mm/s]	7 [ms]	6 [ms]
2000 [mm/s]	18 [ms]	4 [ms]
2500 [mm/s]	21 [ms]	5 [ms]

Table 4.4: Difference between `Samplerate:=4` and `Samplerate:=12`. With `Samplerate:=12` the robot is $4 - 21ms$ further behind than with `Samplerate:=4`.

4.2.3 Analysis

The results show that using `Samplerate:=12` gives slower response than using `Samplerate:=4`. Since the sample rate of the robot changed, the rate at which the data was written to file were changed. The data is written every time the robot ask for new positions. This may be a explanation for why there is an difference. Since the cameras processing time is $10ms$, $\pm 1ms$, the rate at which the robot is fed with new position data is not changed using the two sample rates. With `Samplerate:=4` the robot revives new position data every third sample (every 12thms), which is the same as using `Samplerate:=12`. When disc velocity equals $2000mm/s$ and $2500mm/s$, the difference is quite huge at the first reading ($300mm$). This is most likely a coincidence.

4.3 Ramp Experiments

This section presents all the experiments conducted on the ramp. The purpose of these experiments was to see if the robot could follow a disc sliding down a ramp. Different predictor values were tested. These experiments tried to find an answer to the following questions:

- Can the robot follow a disc sliding down a ramp?
- How far ahead does the predictor have to predict?
- Is the distance between disc and robot constant throughout the motion?

4.3.1 Setup

During the ramp experiments the lab was setup like in Figure 4.12. The Cognex camera was looking down at the ramp and most of the ramp were within the range of the robot. Due to the limited work range of the Irb140 the start point of the

disc was moved down the ramp, by about 300 mm. By limiting the sliding surface typical errors like "50449: Mechanical unit close to joint bound" and "50456: Close to singularity while External Guided Interface active" were eliminated.

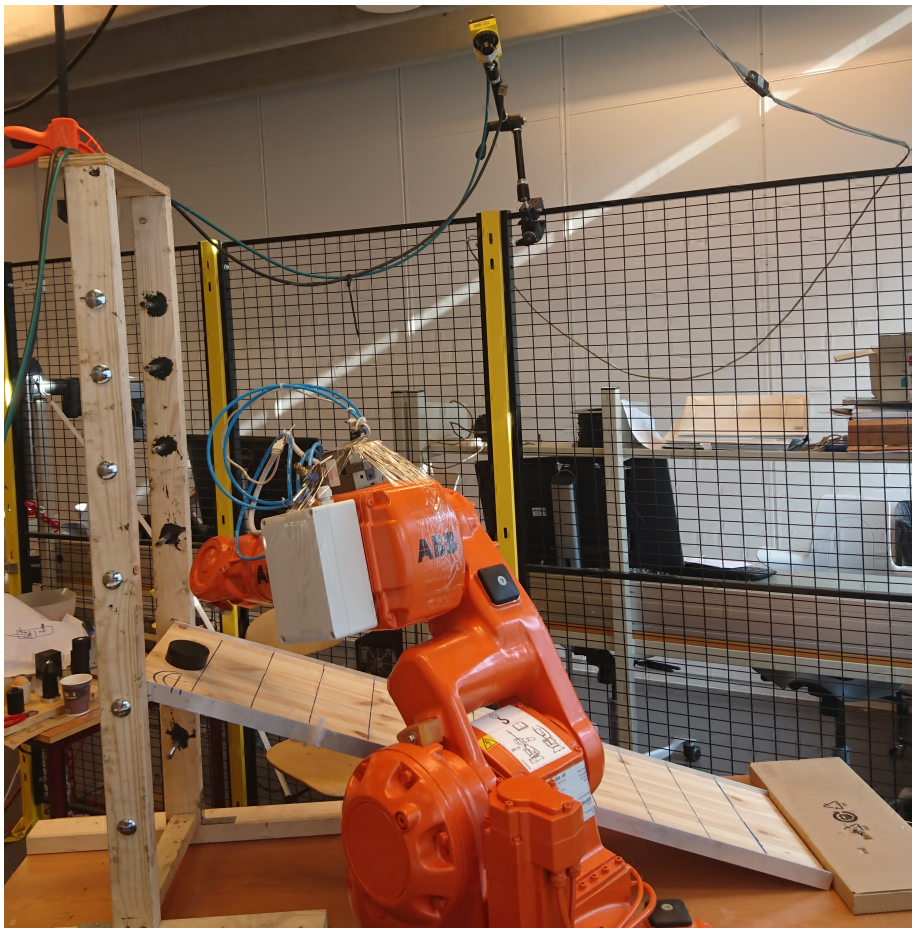


Figure 4.12: Physical setup at the lab. The camera was mounted so the FOV covers the whole ramp, and the ramp was placed within the work area of the robot.

The camera was placed on the opposite side of the ramp in relation to the robot. With a small side view of the ramp to avoid the robot blocking the cameras view of the disc. The camera was also placed high above the ramp to cover the whole sliding surface.[2]

The ramp consisted of wood, with aluminium edges to avoid the disc from sliding out on one of the sides. Each level (the bolts in Figure 4.12) represented a 10 degrees change in the ramps angle. The hight of these levels were calculated using the *Sine Rule*, see Equation 4.4 and Figure 4.13. Due to friction the disc would not slide with an angle of 10 degrees and therefore the application was not tested with a ramp angle lower than 20 degrees.

$$\frac{a}{sin(A)} = \frac{b}{sin(B)} = \frac{c}{sin(C)} \tag{4.4}$$

[2]The lens mounted on the camera had limited wide angle. Other lenses where not tested.

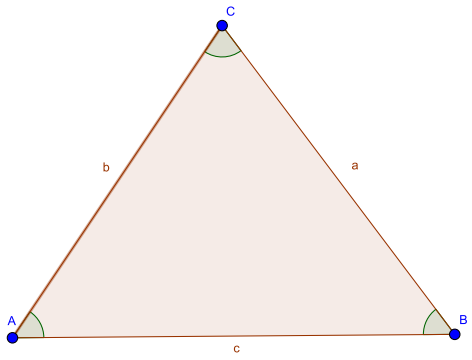


Figure 4.13: Triangle with unknown angles.

The disc was made out of *Polyoxymethylene (POM)* which is a hard types of plastic [15]. Robot used during the tests was an Irb140 ABB industrial robotic arm, with an working range of 810 mm [5].

Two different angles were tested, 20 and 30 degrees. During the experiments the cameras processing time was $15ms$ when ramp angle was 20 degrees and $18ms$ when the angle was 30 degrees. `EGMActPose` and `EGMRunPose` used the parameters in Listing 4.1 and 4.2, unless otherwise stated. Due to the difference in the cameras processing time, `Samplerate:=` depended on the ramps angle. When the ramp angle was 20 degrees, `Samplerate` was equal to 16, and 30 degrees it was set equal to 20.

```
1 EGMActPose egmID1\Tool:=tSuctionCup \WObj:=TopOfRamp,posecorTopOfRamp ,EGM_FRAME_WORLD,
2 posecorTopOfRamp, EGM_FRAME_WORLD \x:=egm_minmax_lin1 \y:=egm_minmax_lin1 \z:=egm_minmax_lin1
3 \rx:=egm_minmax_rot1 \ry:=egm_minmax_rot1 \rz:=egm_minmax_rot1\LpFilter:=3\Samplerate:=16
4 \MaxSpeedDeviation:= 5000;
```

Listing 4.1: `EGMActPose` during ramp test

```
1 EGMRunPose egmID1, EGM_STOP_HOLD\x \y \z\CondTime:=20 \RampInTime:=0.05\RampOutTime:=0.5;
```

Listing 4.2: `EGMRunPose` during ramp test

4.3.2 Noise

In the position data from the camera there were some noise, see Figure 4.14. Samples from the robot were not as noisy as the samples from the camera. The noise showed better when the position was differentiated to produce the velocity and even more when the velocity was differentiated to display the acceleration, see Figure 4.15.

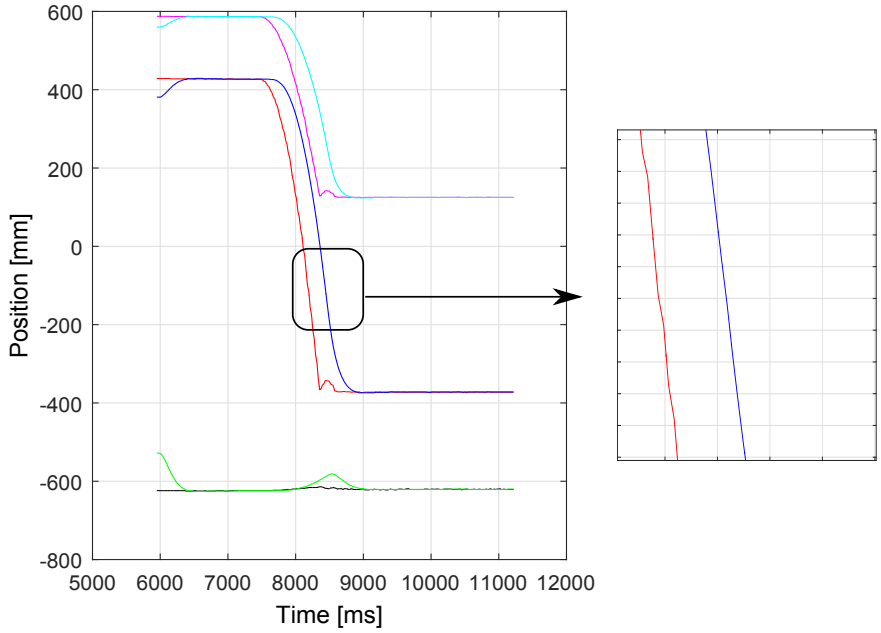
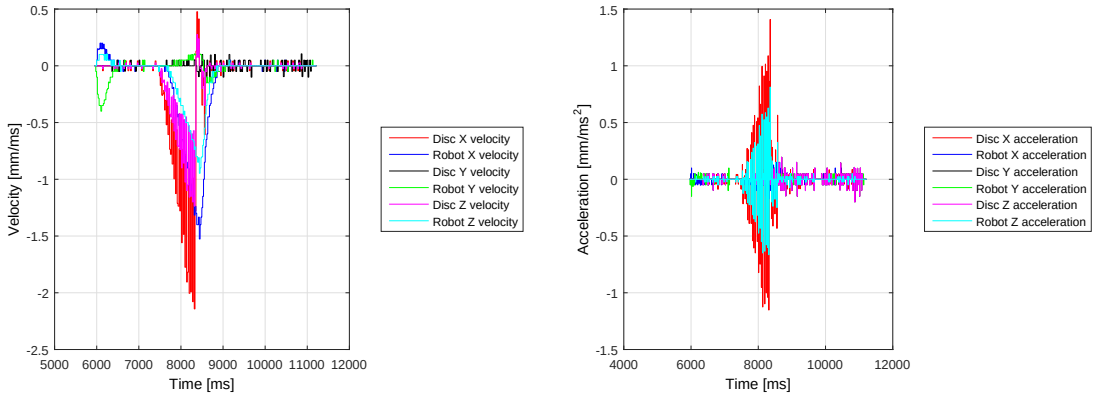


Figure 4.14: Position in x-, y- and z- direction. The zoomed area is to highlight the noise from the camera readings (red curve).



(a) Velocity.

(b) Acceleration.

Figure 4.15: Noise after differentiated the position to get the velocity and acceleration.

Due to the noise, all data presented were smoothed using the `smooth` in Matlab, see Listing 4.3. `'loess'` is local regression using weighted linear least squares and a 2nd degree polynomial [16].

```
for i = 1: 1: 3
  robotXYZi(1:end,i) = smooth(robotXYZi(1:end,i),200,'loess');
end
```

Listing 4.3: Filtering the data using `smooth`

4.3.3 Different Bandwidths on the Low Pass Filter

This experiment tested different bandwidths on the low pass filter which is between the EGM controller and the robots motion controller, see Figure 2.2. This experiment was conducted to see how different bandwidths influenced the movement of the robot.

Setup

This experiment was conducted with the setup described in Section 4.3.1. All tests were run with a 30 degree angle on the ramp.

Results

With bandwidth equal $3Hz$ the robot started its motion $181ms$ after the disc. Midway, when distance travelled was equal to $382mm$, the robot was $253ms$ behind. At the end, from the disc reached its stationary position to the robot stopped its movement the difference was $322ms$, see Figure 4.16.

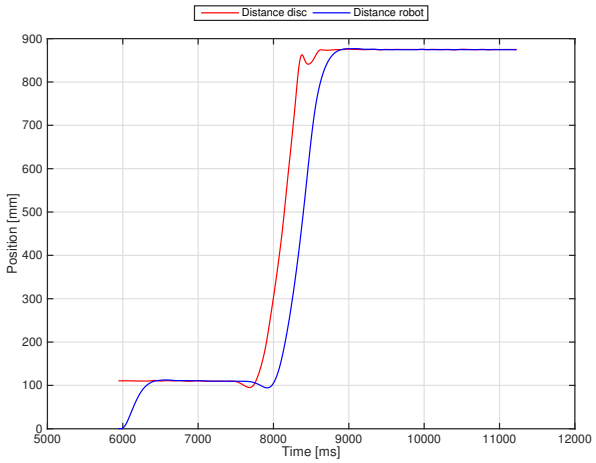


Figure 4.16: Distance travelled with bandwidth low pass filter equal $3Hz$.

When the bandwidth was equal to $5Hz$, the robot started its motion $202ms$ after the disc. Midway the difference was $250ms$ and at the end $444ms$, see Figure 4.17.

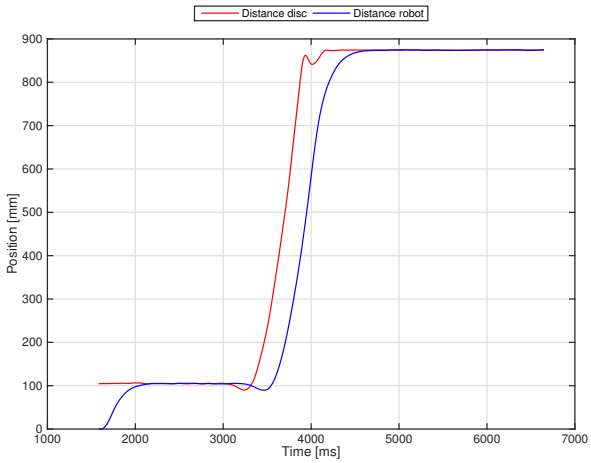


Figure 4.17: Distance travelled with bandwidth low pass filter equal $5Hz$.

When the bandwidth was equal to $10Hz$ the robot started its motion $100ms$ after the disc. Midway the difference was $240ms$. At the end, the robot reached a stationary position $680ms$ after the disc, see Figure 4.18.

A summary of the numbers are presented in Table 4.5.

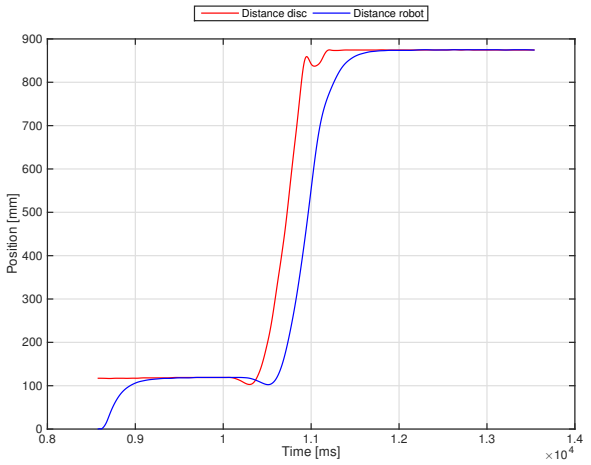


Figure 4.18: Distance travelled with bandwidth low pass filter equal $10Hz$.

Bandwidth equal $15Hz$ was also tested, but triggered a "Dynamic load too high" when the robot moved from the start point to the start point of the disc.

Bandwidth [Hz]	Start [ms]	Midway [ms]	End [ms]
3	181	253	322
5	202	250	444
10	100	240	680

Table 4.5: How long behind the robot is compared to the disc at certain distances using different bandwidths on the low pass filter in the EGM controller.

Analysis

The results show that the bandwidth of the low pass filter affects the deceleration of the robot, higher bandwidth gives longer deceleration. From the numbers it may look like it also affect the start of the motion, with a bandwidth equal to $10Hz$ the robot started its motion faster than with the other bandwidths tested. Since the robot started its motion slower using $5Hz$ compared with $3Hz$, the quicker response with $10Hz$ is most likely a coincident.

4.3.4 Movement in the X-direction

Most of the discs movement were in the work objects x-direction. In this section the results in x-direction are presented. Different time steps in the predictor was tested to find the best fit, see Section 3.2.4. Time step is the time between two consecutive samples, i.e. 15 time steps means the time between two consecutive samples multiplied with 15. Time behind is the time from when the disc had travelled a certain distance until the robot had travelled the same distance or visa versa. Distance travelled gives a good pinpoint on the distance between robot and disc.

20 Degrees Angle on the Ramp

With 20 degrees angle on the ramp the theoretical velocity for the disc after $0.5s$ in the x-direction is equal to $0.29m/s$, while the true velocity was measured to $0.34m/s$, see Figure 4.19.

Results

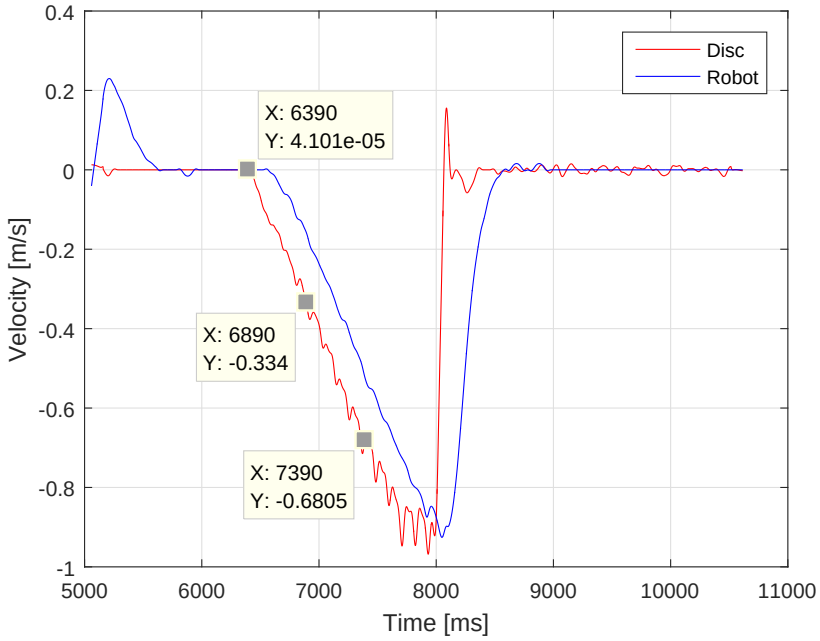


Figure 4.19: Velocity in the x-direction with a 20 degrees angle on the ramp. The cursors are placed at the start of the movement, 0.5 seconds after the movement started and 1 second after the movement started.

The measured velocity was not equal to the theoretical velocity. The difference is presented in Table 4.6.

Time	Theoretical	Measured	Difference
0.5 sec	$0.29[m/s]$	$0.34[m/s]$	$0.05[m/s]$
1.0 sec	$0.59[m/s]$	$0.69[m/s]$	$0.1[m/s]$

Table 4.6: Difference between calculated velocity and measured velocity for the disc in the x-direction. Ramp angle was 20 degrees.

With no prediction the robot is behind the disc throughout the motion. $0.5s$ into the motion the robot was $238ms$ behind the disc, after $1s$ the difference was $241ms$ and $1.25s$ into the motion the difference was $240ms$. The difference in the robots and discs position was approximately constant throughout the motion, except during acceleration and deceleration, see Figure 4.20.

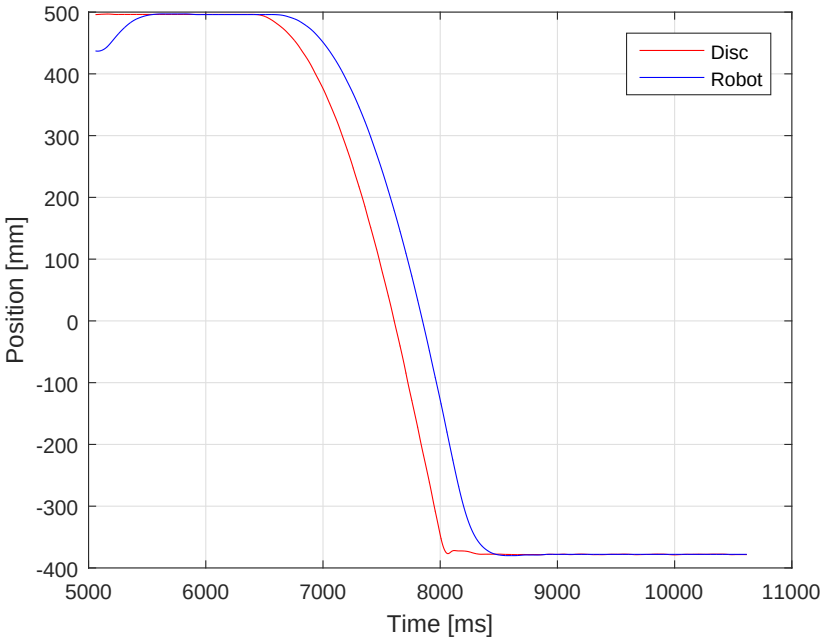


Figure 4.20: No prediction, 20 degrees angle on the ramp.

Predicting 15 time steps ahead the robot was behind the disc throughout the motion. $0.5s$ in the motion the robot was $81ms$ behind the disc. $1s$ the difference was $66ms$ and after $1.25s$ the difference was $60ms$. How far the robot was behind the disc were not constant, see Figure 4.21. There was an overshoot at the end of the robots motion.

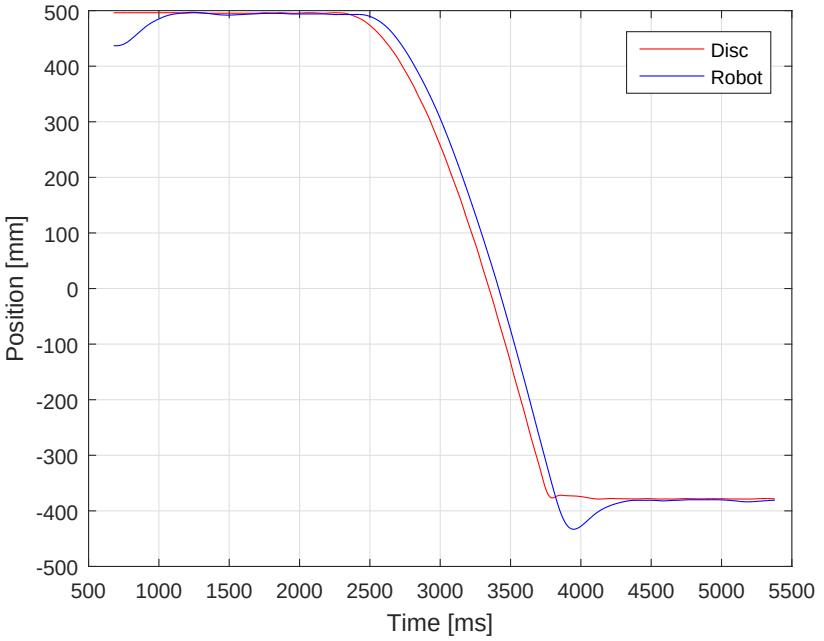


Figure 4.21: Predicting 15 time steps ahead. Ramp angle was 20 degrees.

The robot caught up with the disc and pass it when predicting 20 time steps ahead. $0.5s$ into the motion the robot was $42ms$ behind the disc. After $1s$ the difference was $12ms$ and $1.25s$ into the motion the robot had passed the disc and was $16ms$ in front of the disc, see Figure 4.22. There was an overshoot at the end of the robots motion.

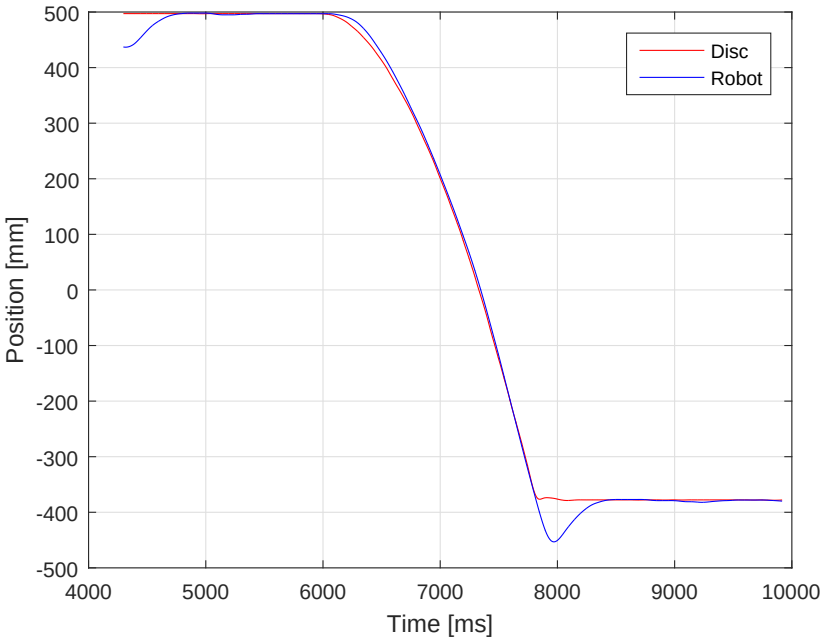


Figure 4.22: Predicting 20 time steps ahead. Ramp angle was 20 degrees.

Predicting 25 time steps ahead the robot caught up with and passed the disc. $0.5s$ into the motion the robot was $10ms$ behind the disc. After $1s$ the robot had passed the disc and was $31ms$ in front of the disc. $1.25s$ into the motion the robot was $46ms$ ahead of the disc, see Figure 4.23. There was an overshoot at the end of the robots motion.

A summary of all numbers presented in this section are found in Table 4.7.

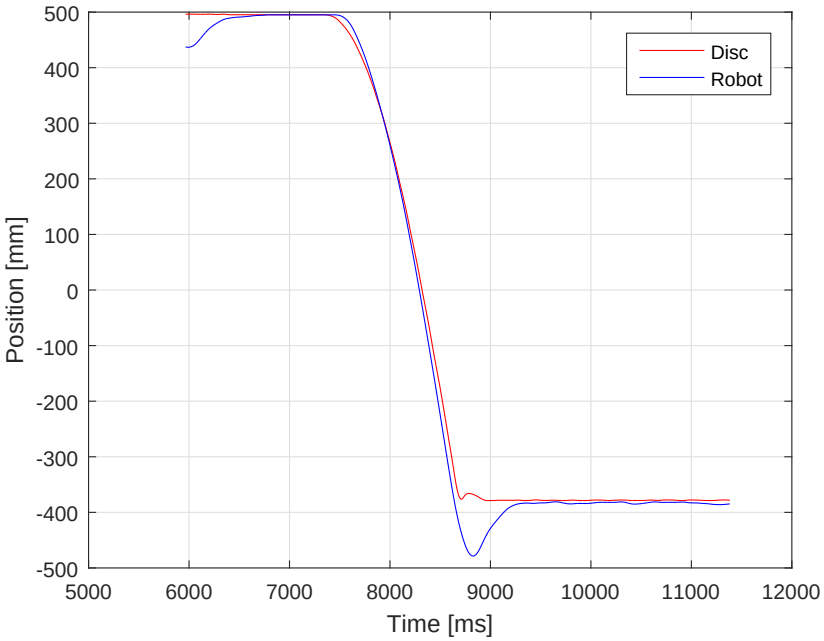


Figure 4.23: Predicting 25 time steps ahead. Ramp angle was 20 degrees.

	Time steps in the predictor			
Time	**0**	**15**	**20**	**25**
0.5 sec	$-238[ms]$	$-81[ms]$	$-42[ms]$	$-10[ms]$
1.0 sec	$-241[ms]$	$-66[ms]$	$-12[ms]$	$31[ms]$
1.25 sec	$-240[ms]$	$-60[ms]$	$16[ms]$	$46[ms]$

Table 4.7: Difference in when the disc and robot reached the same point in the x-direction. Negative sign means the robot was behind and positive means it was in front. Ramp angle was 20 degrees.

30 Degrees Angle on the Ramp

Changing the angle of the ramp to 30 degrees will give an theoretical velocity after $0.5s$ equal to $1.18m/s$ in the x-direction, measured value was equal to $1.06m/s$, see Figure 4.24.

Results

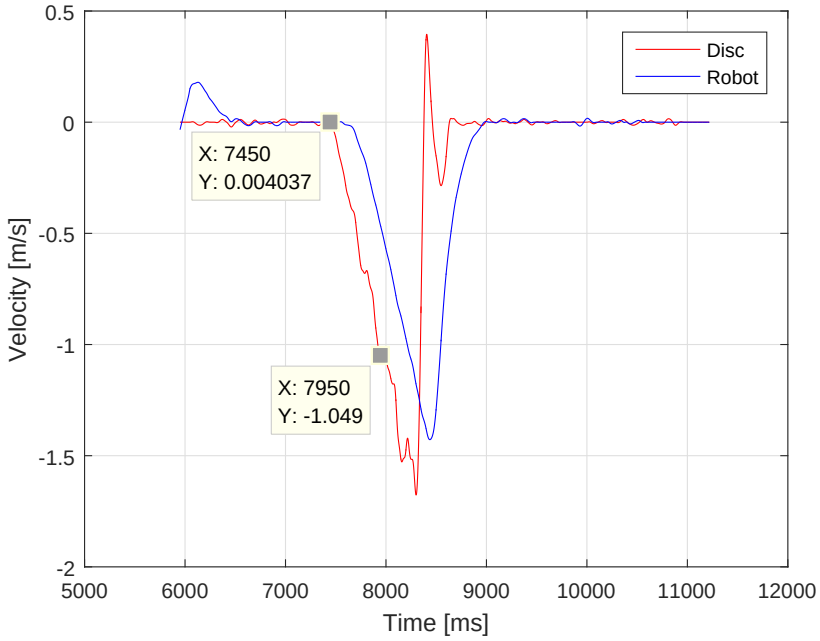


Figure 4.24: Velocity in the x-direction with a 30 degrees angle on the ramp. The cursors are placed at the start of the movement and 0.5 seconds after the movement started

Without prediction the robot was behind the robot throughout the motion. How far behind the robot were was not constant. 250ms into the motion the robot was 234ms behind. 500ms into the motion the difference was 250ms and 750ms after the motion had started the difference was 255ms, see Figure 4.25.

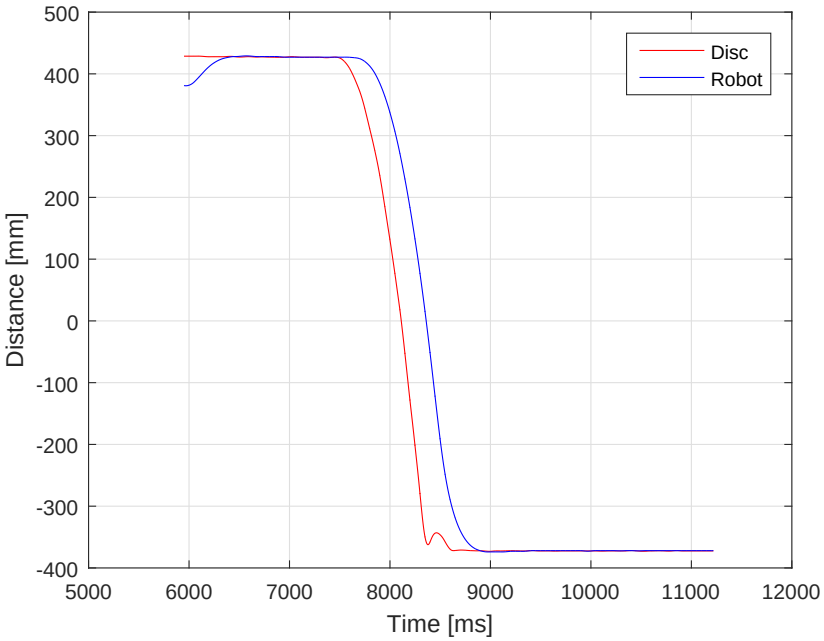


Figure 4.25: No prediction, 30 degrees angle on the ramp.

When predicting 15 time steps ahead the robot was behind the disc throughout the motion. The difference was not constant. After $250ms$ the robot was $105ms$ behind the disc. $500ms$ into the motion the difference was $91ms$ and after $750ms$ the difference was $85ms$, see Figure 4.26. There was an overshoot at the end of the robots motion.

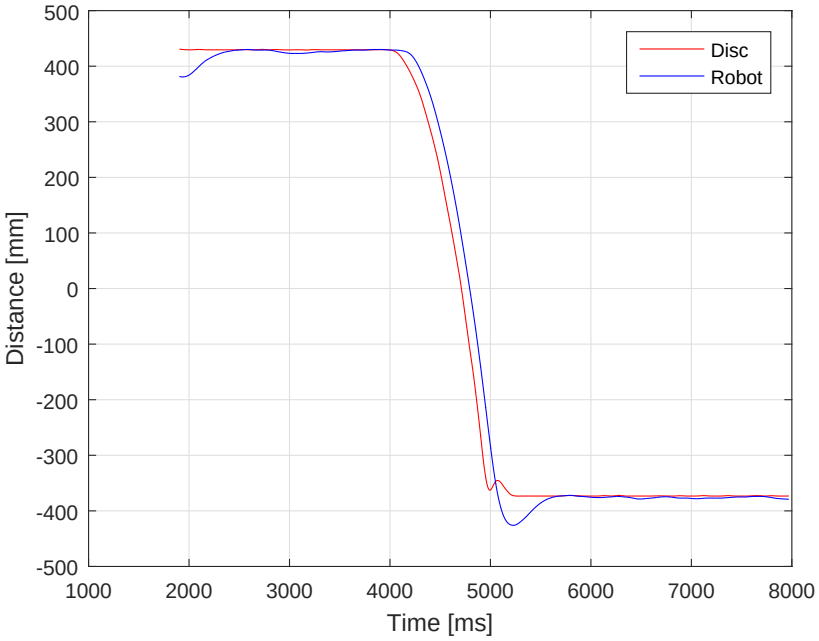


Figure 4.26: Predicting 15 time steps ahead. Ramp angle was 30 degrees

Predicting 20 time steps ahead the robot was behind the disc throughout the motion. The difference was not constant. After 250ms the robot was 82ms behind the disc. 500ms into the motion the robot was 54ms behind and 750ms into the motion the robot was 35ms behind, see Figure 4.27. There was an overshoot at the end of the robots motion.

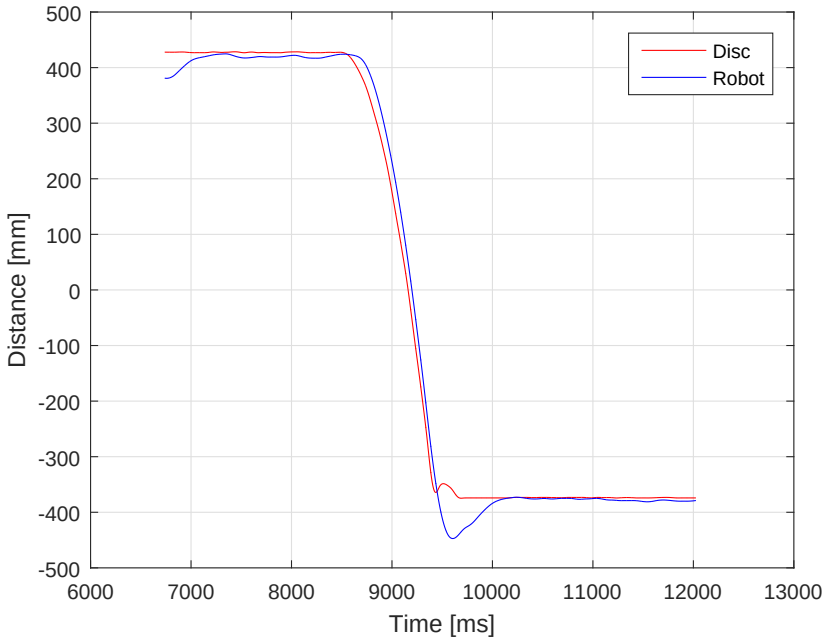


Figure 4.27: Predicting 20 time steps ahead. Ramp angle was 30 degrees.

When predicting 25 time steps ahead in time the robot caught up with and passed the disc. After 250ms the robot was 72ms behind the disc. 500ms after the movement started the difference was 20ms and 750ms into the motion the robot had passed the disc and was 2ms in front of the disc.

All numbers presented in this section are summarized in Table 4.8.

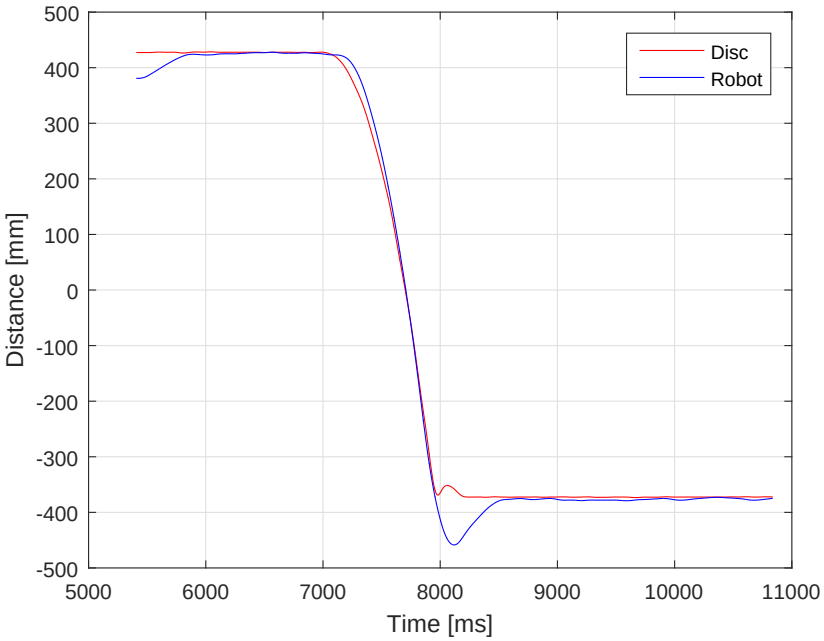


Figure 4.28: Predicting 25 time steps ahead. Ramp angle is 30 degrees

Time	**0**	**15**	**20**	**25**
$250[ms]$	$-234[ms]$	$-105[ms]$	$-82[ms]$	$-72[ms]$
$500[ms]$	$-250[ms]$	$-91[ms]$	$-54[ms]$	$-20[ms]$
$750[ms]$	$-255[ms]$	$-85[ms]$	$-35[ms]$	$2[ms]$

Table 4.8: Difference in time between disc and robot had reached certain points. Negative sign means the robot is behind and positive means it is in front. Ramp angle was 30 degrees

4.3.5 Movement in all Directions

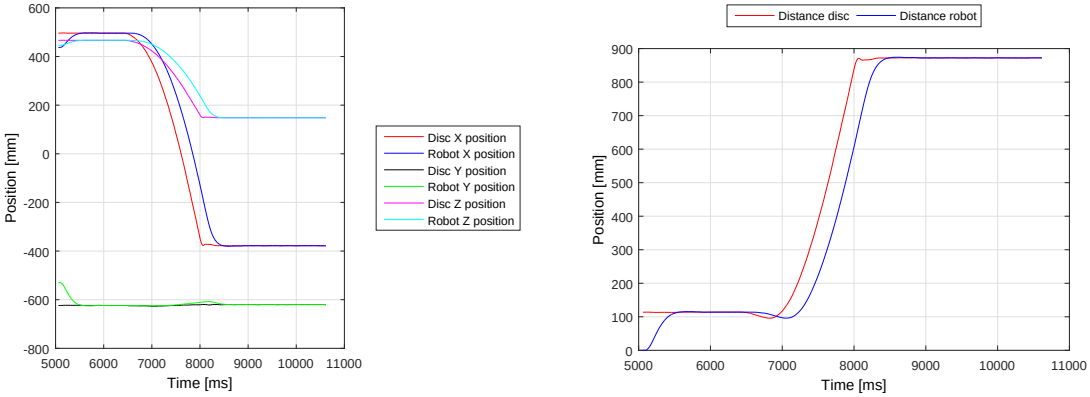
In this section the results from the movement in all three axis are presented. The results come from the same experiments as the results presented in Section 4.3.4. Looking at distance travelled gives a good indication on the difference in position. How far the robot was behind the disc and visa versa. Velocities are also presented.

20 Degrees Angle on the Ramp

This section presents the results and analysis from the movement in all three directions. Midway means that the distance travelled is equal to $380mm$ and at the end of the motion means that distance travelled equals $800mm$. This is right before the robot starts its deceleration.

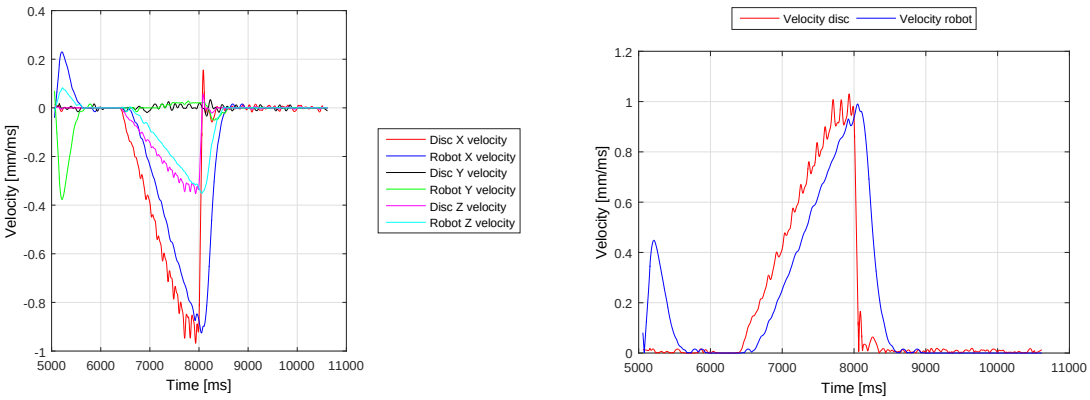
Results

Without prediction the robot was 245ms behind the disc midway in the motion. At the end of the motion the robot was 256ms behind the disc, see Figure 4.29. Distance between disc and robot was not constant throughout the motion. The disc reached a top velocity at 1.03m/s, while the robots top velocity was 0.99m/s, see Figure 4.30.



(a) Position. (b) Distance.

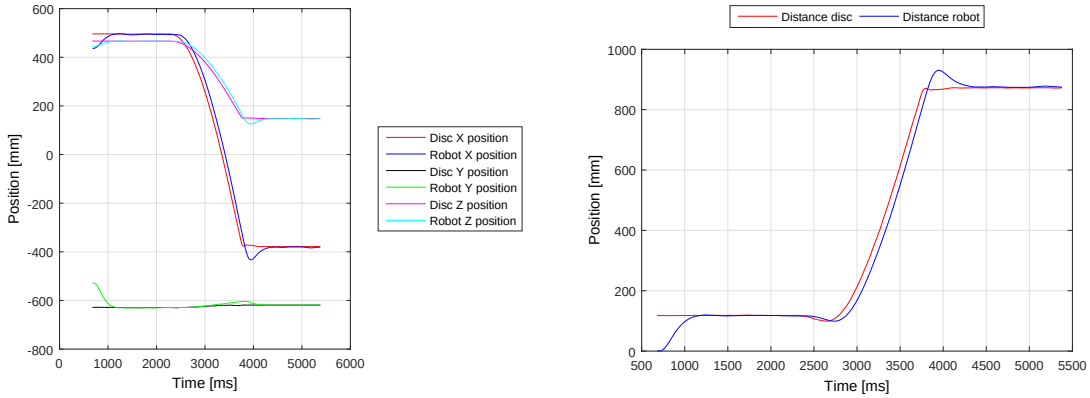
Figure 4.29: Position and distance without prediction with a 20 degree angle on the ramp.



(a) Velocity in all directions. (b) Combined velocity.

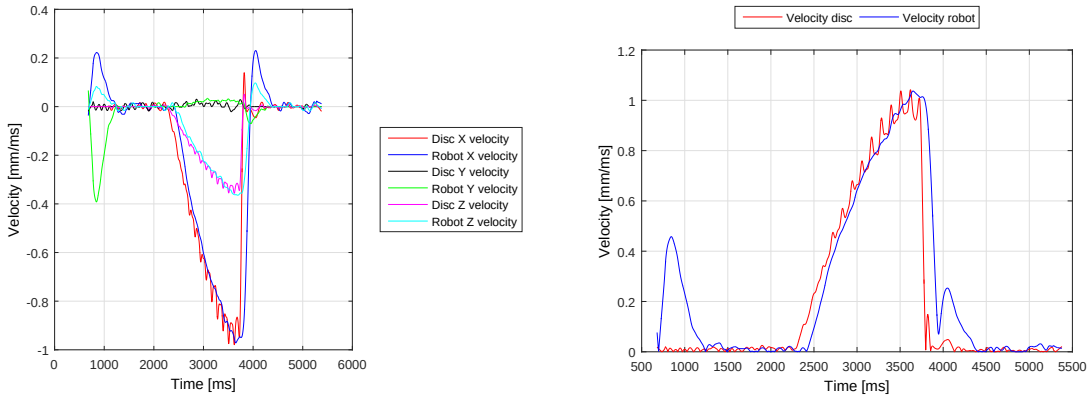
Figure 4.30: Velocities without prediction with a 20 degree angle on the ramp.

Predicting 15 time steps ahead the robot was 71ms behind the disc midway. At the end of the motion the robot was 56ms behind the disc, see Figure 4.31. There was an overshoot equal to 58mm at the end of the robots motion. The robot reached a top velocity equal to 1.04m/s and the discs top velocity was 1.04m/s, see Figure 4.32.



(a) Position. (b) Distance.

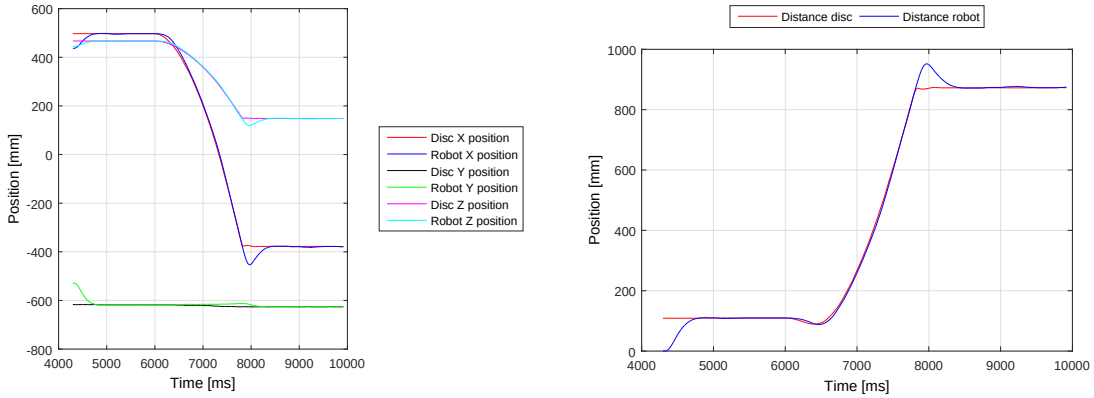
Figure 4.31: Position and distance predicting 15 time steps ahead. Ramp angle was 20 degrees.



(a) Velocity in all directions. (b) Combined velocity.

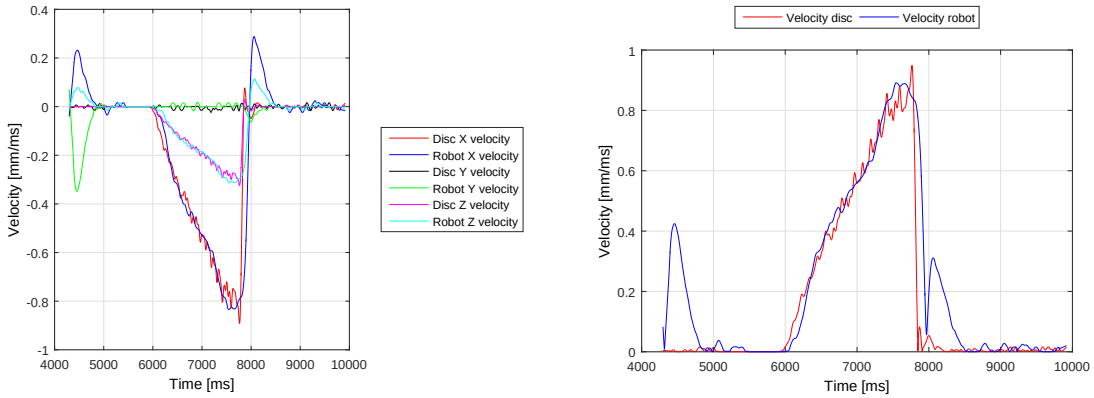
Figure 4.32: Velocities predicting 15 time steps ahead. Ramp angle was 20 degrees.

When predicting 20 time steps ahead the robot caught up with the disc. The robots position was approximately equal to the discs throughout the motion. Midway the robot was $17ms$ behind the disc. At the end of the motion the robot was $5ms$ in front of the disc, see Figure 4.33. There was an overshoot at the end of the robots motion equal to $79mm$. The top velocity of the disc was $0.95m/s$ and the robots was $0.89m/s$, see Figure 4.34.



(a) Position. (b) Distance.

Figure 4.33: Position and distance predicting 20 time steps ahead. Ramp angle was 20 degrees.

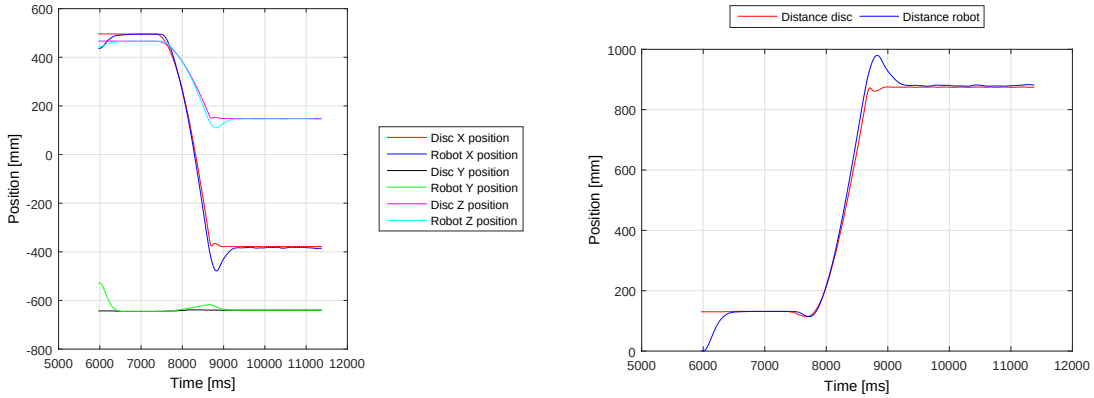


(a) Velocity in all directions. (b) Combined velocity.

Figure 4.34: Velocities predicting 20 time steps ahead. Ramp angle was 20 degrees.

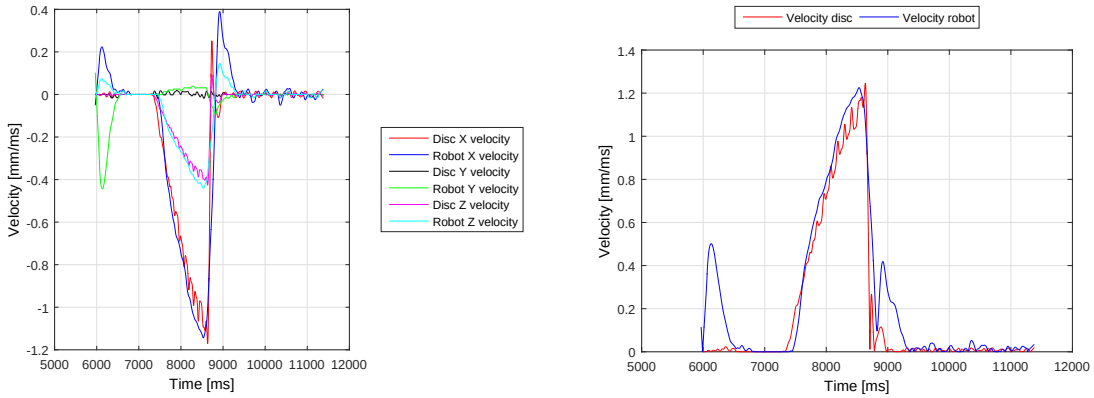
The robot caught up and passed the disc when predicting 25 time steps ahead. Midway the robot was 17ms in front of the disc. At the end of the motion the robot was 47ms in front of the disc, see Figure 4.35. There was an overshoot at the end of the robots motion equal to 105mm. The discs top velocity was 1.25m/s and the robots top velocity was 1.22m/s, see figure 4.36.

A summary of the difference between the robot and disc's position is found in Table 4.9.



(a) Position. (b) Distance.

Figure 4.35: Position and distance predicting 25 time steps ahead. Ramp angle was 20 degrees.



(a) Velocity in all directions. (b) Combined velocity.

Figure 4.36: Velocities predicting 25 time steps ahead Ramp angle was 20 degrees.

	Time steps in the predictor			
Distance	**0**	**15**	**20**	**25**
380 [mm]	$-245[ms]$	$-71[ms]$	$-17[ms]$	$17[ms]$
800 [mm]	$-256[ms]$	$-56[ms]$	$5[ms]$	$47[ms]$

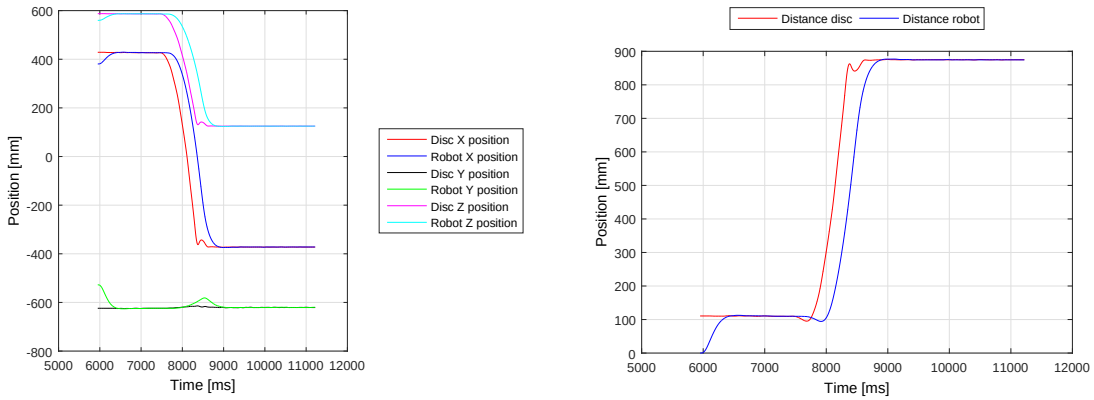
Table 4.9: Difference in when the disc and robot reached the same point. Negative sign means the robot was behind and positive means it was in front. Ramp angle was 20 degrees.

30 Degrees Angle on the Ramp

This section presented the results in from movement in all three directions with a 30 degree angle on the ramp. Midway means that distance travelled was equal $390mm$ and at the end of the motion, distance travelled was equal $700mm$. This was before the robot started its deceleration.

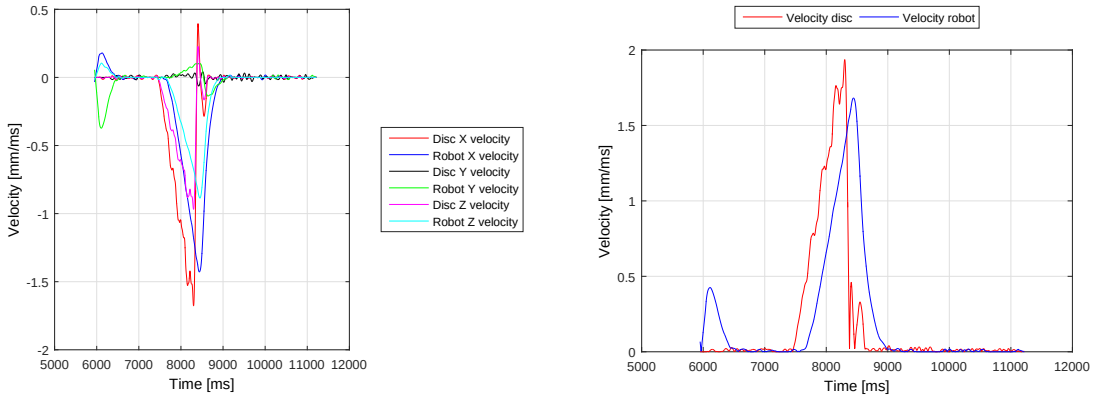
Results

Without prediction the robot laid behind the disc throughout the motion. Midway the robot was $253ms$ behind the disc. At the end of the motion the robot was $262ms$ behind, see Figure 4.37. The discs top velocity was $1.94m/s$ and the robots top velocity was $1.68m/s$, see Figure 4.38.



(a) Position. (b) Distance.

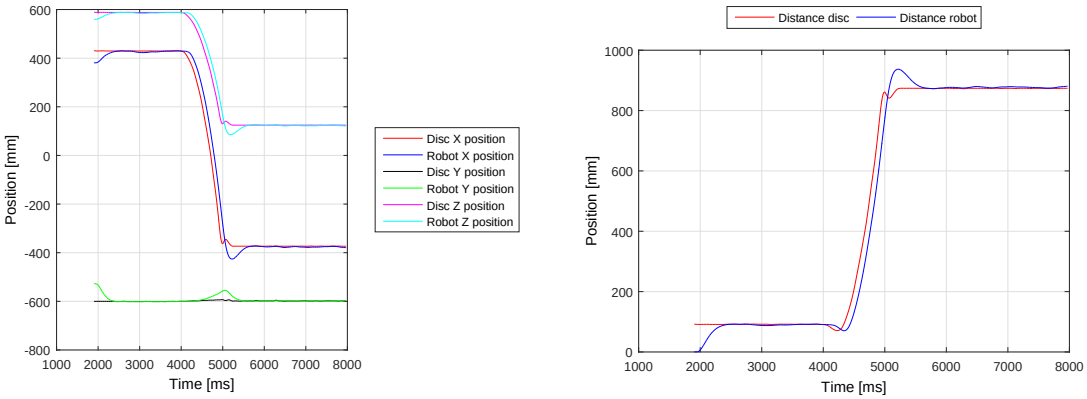
Figure 4.37: Position and distance without prediction with a 30 degree angle on the ramp.



(a) Velocity in all directions. (b) Combined velocity.

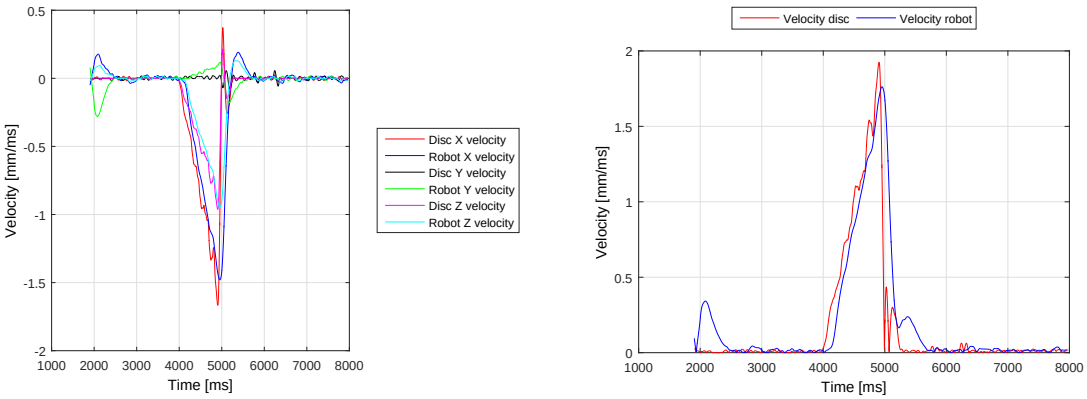
Figure 4.38: Velocities without prediction with a 30 degree angle on the ramp.

Predicting 15 time steps ahead the robot was behind the disc throughout the motion. Midway the robot was 86ms behind. At the end of the motion the robot was 81ms behind. There was an overshoot in the robots motion at the end equal to 63mm , see Figure 4.39. The discs top velocity was 1.93m/s and the robots top velocity was 1.76m/s, see Figure 4.40.



(a) Position. (b) Distance.

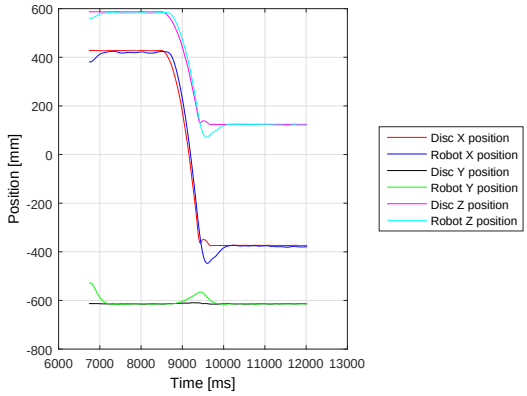
Figure 4.39: Position and distance predicting 15 time steps ahead. Ramp angle was 30 degrees.



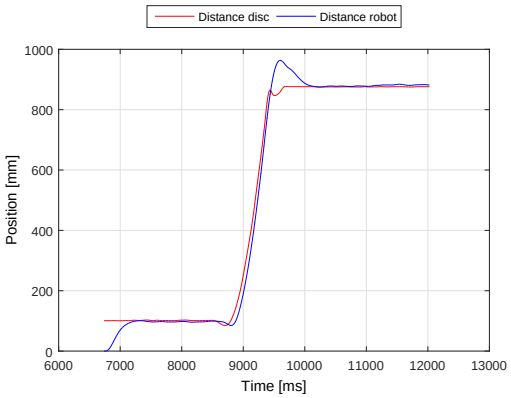
(a) Velocity in all directions. (b) Combined velocity.

Figure 4.40: Velocities predicting 15 time steps ahead. Ramp angle was 30 degrees.

Predicting 20 time steps ahead the robot was behind the disc throughout the motion. Midway the robot was 40ms behind the disc. At the end of the motion the robot was 30ms behind the disc. There was an overshoot in the robots motion at the end equal to 87mm, see Figure 4.41. The discs top velocity was 1.98m/s and the robots top velocity was 1.91m/s, see Figure 4.42.

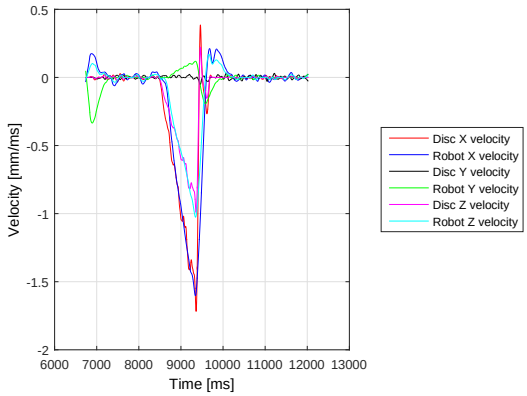


(a) Position.

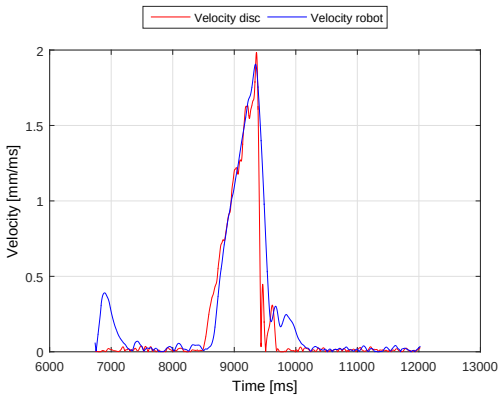


(b) Distance.

Figure 4.41: Position and distance predicting 20 time steps ahead. Ramp angle was 30 degrees.



(a) Velocity in all directions.

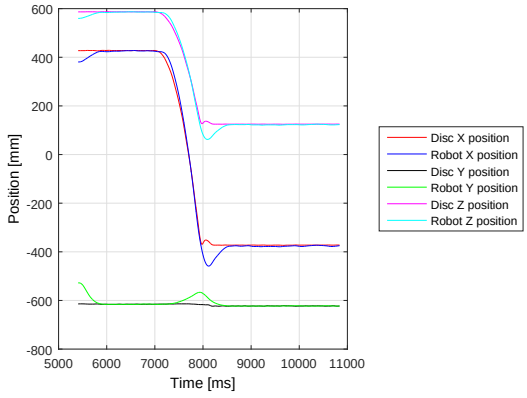


(b) Combined velocity.

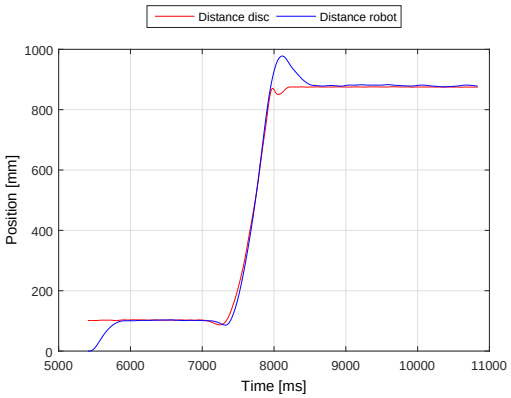
Figure 4.42: Velocities predicting 20 time steps ahead Ramp angle was 30 degrees.

Predicting 25 time steps ahead in time the robot caught up and passed the disc. The difference in time midway in the motion was $14ms$. At the end of the motion the robot was $11ms$ in front of the disc. There was an overshoot at the end of the robots motion equal to $102mm$, see Figure 4.43. The discs top velocity was $1.88m/s$ and the robots top velocity was $1.98m/s$, see Figure 4.44.

A summary of the difference between the robot and disc's position is found in Table 4.10.

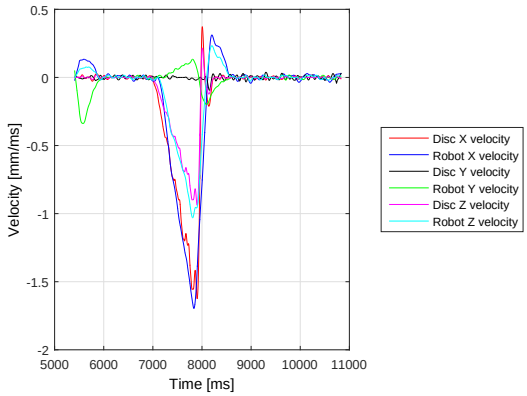


(a) Position.

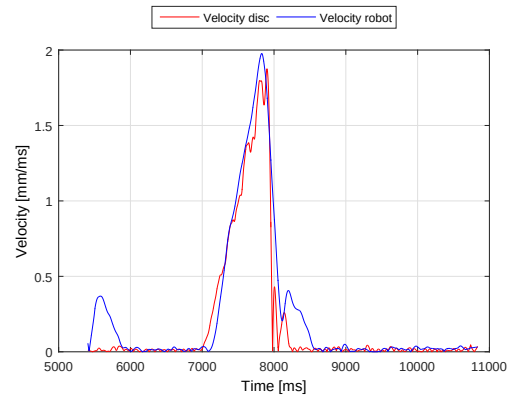


(b) Distance.

Figure 4.43: Position and distance predicting 25 time steps ahead. Ramp angle was 30 degrees.



(a) Velocity in all directions.



(b) Combined velocity.

Figure 4.44: Velocities predicting 25 time steps ahead. Ramp angle was 30 degrees.

	Time steps in the predictor			
Distance	**0**	**15**	**20**	**25**
390 [mm]	$-253[ms]$	$-86[ms]$	$-40[ms]$	$-14[ms]$
700 [mm]	$-262[ms]$	$-81[ms]$	$-30[ms]$	$11[ms]$

Table 4.10: Difference in when the disc and robot reached the same point. Negative sign means the robot was behind and positive means it was in front. Ramp angle was 30 degrees.

4.3.6 Analysis of the Ramp Experiments

This section holds the analysis for the experiments presented in Section 4.3.4 and 4.3.5.

20 Degree Angle

Without prediction the difference in the disc's and robot's position were constant throughout the motion, except during acceleration and deceleration.With prediction the robot was able to follow the disc, but the difference in disc and robot's position were not constant throughout the motion. This indicates that the predictor was not optimal. Further discussion around the predictor is found in Chapter 5. Predicting 20 time steps ahead gave the best result. The overshoots were due to the predictor. At the end of the disc's movement the predictor sent targets to the robot which were beyond the stop point of the disc.

30 Degree Angle

With no prediction the distance between the robot and disc were not constant throughout the motion. The difference was not constant using a predictor. This points to the fact that the predictor was not optimal. The robot is able to follow the disc, even though the difference was not constant. The overshoots were due to the predictor. At the end of the disc's movement the predictor sent targets to the robot which were beyond the stop point of the disc. Predicting 25 time steps ahead gave the best results.

With a 20 degree angle predicting 20 time steps ahead gave the best results and with a 30 degree angle predicting 25 time steps ahead gave the best results. This is another indication that the predictor was not optimal.

5. Discussion

This chapter contains some discussions around the system and the results presented in this thesis.

5.1 Externally Guided Motion

This thesis suggest a solution on how to implement EGM in a system using a camera to give the robot targets. The solution presented gives acceptable results. The robot is able to follow the disc with velocities upwards to max TCP velocity for the robot. This makes the results even more impressive. EGM can potentially be a "game changer" giving more effective production lines. Even though EGM gives good results there are some limitations:

- Lack of interpolation. There is possible to interpolate the sensor data before it reaches the robot, but this may cause jagged movements.
- Without path planning the robot can not plan a path to avoid awkward positions. Positions which may trigger errors and stop the motion e.g. "50449: Mechanical unit close to joint bound at the end of the movement".
- EGM can trigger a lot of the robots safety mechanisms, which stops of the motion. Error messages like "Dynamic load too high" and "Mechanical unit close to joint bound" popped up frequently during this thesis, e.g. if two consecutive targets are far from one another (about $400mm$) this triggered a "Dynamic load too high" because the robot tried to move to fast. There is possible to set a maximum admitted joint speed change, but this will trim the acceleration/deceleration.

5.2 Noise in the Discs Position

The noise in the camera readings may come from the lack of time synchronization. The frequency which the robot asks for a new position estimate and the time the camera uses to take a new picture, find the disc and send the position vary with $\pm 1ms$. The times are not synchronized. The position of the robot and disc is written to file every time the robot request a new position estimate. Lets say that the camera uses $12ms$ to take a new picture, find the disc and send the position over to the PC. The robot requests a new position from the PC with a frequency

of $12ms$. The disc position which is being written may be $11ms$ old. In the next iteration, lets say that the camera uses $12ms$ to send new data, and that the robot asks for a new position after $13ms$. Then the discs position which are written to file would be $0ms$ old. There is also the risk of skipping one position estimate. The fact that there is noise in the data from the simulated disc experiment, presented in Section 4.1, helps to substantiate this theory.

The other more obvious explanation is that there is noise in the readings it self. In In-Sight Explorer it is possible to see the value that represents the position of the disc in mm. When the program ran there where some oscillations in this values, but not by more than $1mm$ at the most. But this was just by visual registration while the program ran, and since this value changed every $10 - 20ms$ it is not possible to conclude if the value varied more than the naked eye registered. A method to register the positions as they came from the camera with timestamps should have been implemented to test if this was the reason for the noise.

5.3 Friction Coefficient

Friction coefficient chosen in Section 3.2.4 is most likely a bit off, since the difference between disc and robot is not constant in the results presented in Section 4.3. The coefficient is not off by much, but small adjustments in the value have a big affect on the acceleration, e.g. with an angle equal to 20 degrees, the acceleration will be equal to $0.59m/s^2$ with 0.3 and $0.68m/s^2$ if the friction coefficient is equal to 0.29. Wrong friction coefficient is most likely one of the reasons that the predictor is not optimal.

A better way might have been to calculate the acceleration from the second derivative of the position. To be able to do this the noise has to be minimized or filtered.

6. Conclusion and Future Work

EGM have been introduced and presented theoretical and experimentally. The use of EGM in sensor guided paths minimizes the time from new sensor data is sent, to the robot initiates the movement. A suggested solution have been presented. This solution enables the robot to follow a disc sliding down a ramp. With prediction the robot was able to follow the disc with velocities upwards to $2m/s$ and with a sliding surface equal to $800mm$. Without prediction the robot was able to follow, but was $200 - 300ms$ behind the disc throughout the motion. With more effective code and better tuned EGM parameters it should be possible to bisect this difference. EGM gives a quick enough response to new sensor data, enabling the robot to follow an object moving with high, non-constant velocity in a 2D-space. Using a better predictor will produce even better results. Implementing a predictor in the y-direction can enable the disc to bounced in the edges of the ramp and give an more complex motion down the ramp.

6.1 Future Work

- **Improve the predictor:** The predictor implemented is not optimal. The difference between the robot and discs position is not constant throughout the motion. A better predictor will improve the results.
- **Implement a predictor on the y-axis:** Predicting the discs path in the y-axis can enable more complex motions down the ramp.
- **Pick up the disc:** This is the next natural step of this application. A way to do this might be to have an vacuum tool turned on from the start of the movement. When the difference between the discs - and robots - position is lower than a certain threshold, lower the robots height over the ramp until it is at the same height as the disc. After a certain time, when the disc is attached to the tool[1], command the robot on another path which leads out of the ramp.
- **Compare with Robot Reference Interface (RRI):** In this thesis it has been proven that EGM gives fast response to external sensor data, but how fast is it compare to other modules? It might be useful to develop develop two equal applications, one which uses RRI, which is the predecessor to EGM and one which use EGM. Compare the responsiveness of the two applications to see if EGM gives better response, and if so, how much better response.

[1] How long this takes have to be found by experimenting

- **Implement EGM in an existing application using external sensors:** To test if the system can perform its task's faster and if it affects the quality.

Bibliography

[1] ABB, *Application manual - controller software IRC5. RobotWare 6.01*, revision a. ed., 2015. Document ID: 3HAC050798-001.

[2] D. Deng, W. Sam, P. Tao, and T. Lubecki1, "Sensor guided robot path generation for repair of buoyancy module," in *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1613 – 1618, IEEE, 2015.

[3] H. Koch, A.König, A.Weigl-Seitz, K.Kleinmann, and J.Suchý, "Multisensor contour following with vision, force, and acceleration sensors for an industrial robot," in *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, vol. 62, pp. 268 – 280, IEEE, 2013.

[4] Z. Luoa, J. Daia, C.Wangc, F.Wangd, Y.Tiana, and M.Zhaoc, "Predictive seam tracking with iteratively learned feedforward compensation for high-precision robotic laser welding," in *Advances in Manufacturing Machines and Systems – Selected Papers from the 9th International Conference of Frontiers of Design and Manufacturing*, vol. 31, p. 2–7, Elsevier Ltd, 2012.

[5] ABB., "Irb 140. small, powerful and fast 6-axes robot.." http://new.abb.com/products/robotics/industrial-robots/irb-140. Accessed January 24. 2016.

[6] ABB., "Latest robot controller software from abb combines flexibility and productivity for developers.." http://www.abb.com/cawp/seitp202/2a21e772c71016d1c1257daf004046bc.aspx. Accessed January 24. 2016.

[7] Cognex., "In-sight 5000 datasheet.." http://www.cognex.com/downloads/LiteratureDelivery.aspx?id=12966. IS5DS-1309. Visited March 04. 2016.

[8] Cognex., "In-sight standard vision system specifications.." http://www.cognex.com/support/downloads/ns/1/11/35/51xx_54xx_ss.pdf. Accessed March 04. 2016.

[9] T. Morris., *Computer vison and image processing.* Palgrave Macmillan., 2004. ISBN : 0-333-99451-5.

[10] M. Sonka, V. Hlavac, and R. Boyle., *Image Processing, Analysis, and Machine Vision.* Brooks/Cole Publishing Company., 1999. ISBN : 0-534-95393-X.

[11] j. John W. Jewett and R. A.Serway, *Physics for scientists and engineers*, vol. 1. Brooks/Cole Cengage learing, 8th. international ed., 2010. ISBN10: 1-4390 -4876-2.

[12] M. W. Spong, Hutchinson, and Vidyasagar., *Robot Modeling and Control.* John wiley and Sons, INC., 2006. ISBN : 978-0-471-64990-8.

[13] J. Skeet., "protobuf-csharp-port.." https://github.com/jskeet/protobuf-csharp-port. IS5DS-1309. Visited May 16 2016.

[14] Unknown., "Coefficient of friction, rolling resistance and aerodynamics.." http://www.tribology-abc.com/abc/cof.htm. Accessed May 29 2016.

[15] S. Ore and A. Stori, "Plast.." https://snl.no/plast. Accessed May 26 2016.

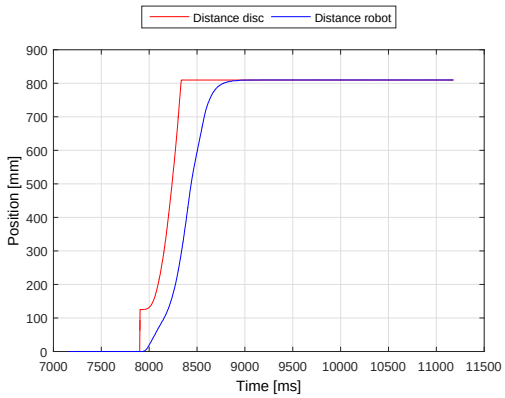
[16] MathWorks., "Curve fitting toolbox - smooth.." http://se.mathworks.com/help/curvefit/smooth.html. Accessed June 08. 2016.

A. Experiment With Constant Acceleration

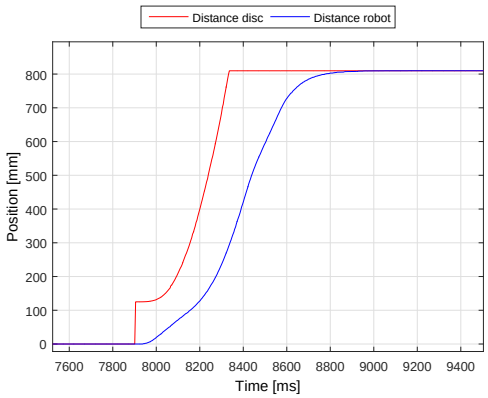
In this appendix plots showing all the results from the experiment in Chapter 4.1.1 is presented. There is also given an example which shows the difference between smoothed and not smoothed plots.

A.1 Without Smoothing

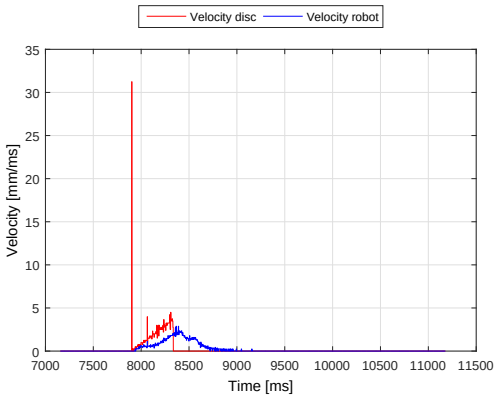
Angle of the simulated ramp was 60 degrees. This section displays the noise present in the readings.



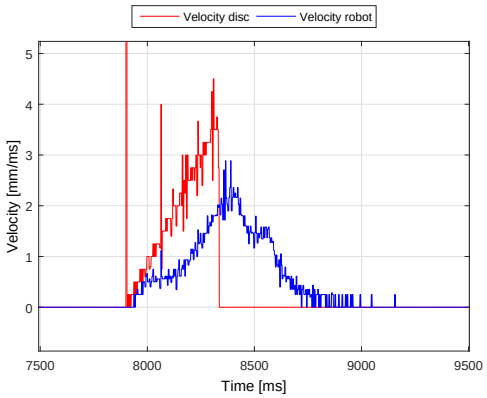
(a) Distance without smoothing



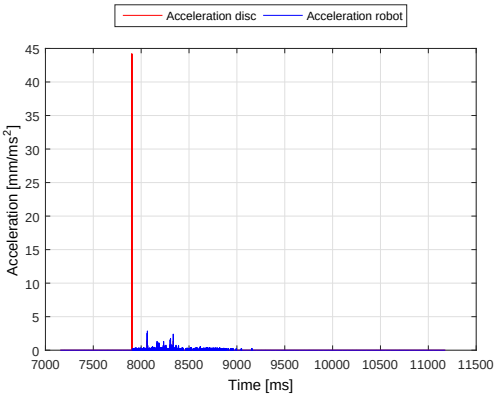
(b) Distance without smoothing, zoomed in



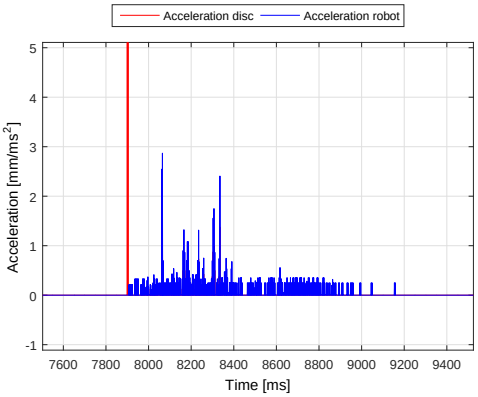
(a) Velocity without smoothing



(b) Velocity without smoothing, zoomed in



(a) Acceleration without smoothing



(b) Acceleration without smoothing, zoomed in

A.2 20 Degrees

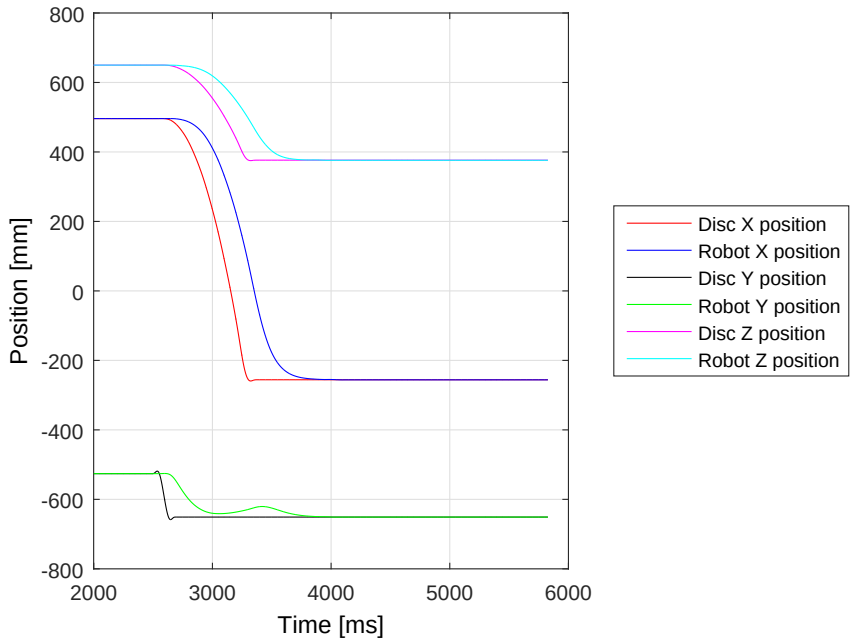


Figure A.4: Positions in all three directions

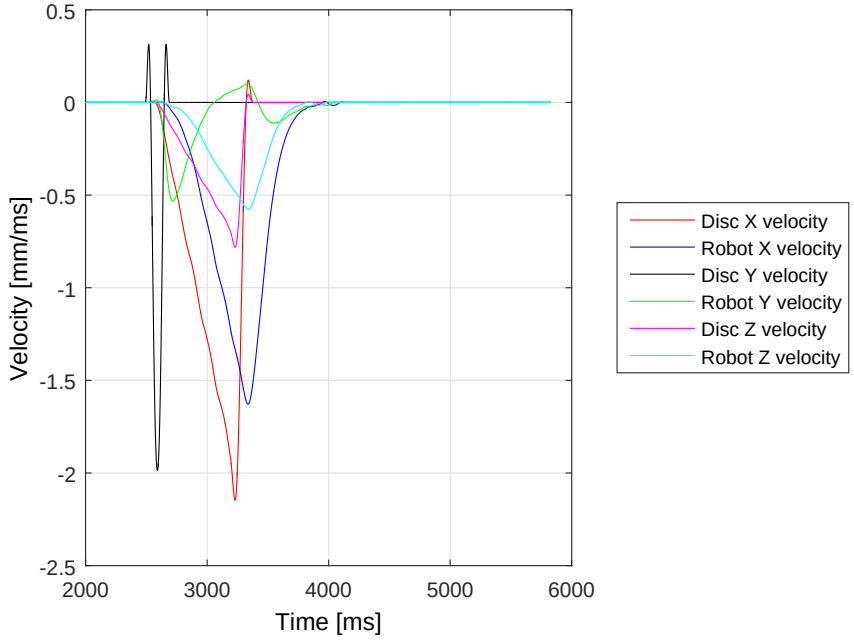


Figure A.5: Velocity in all three directions

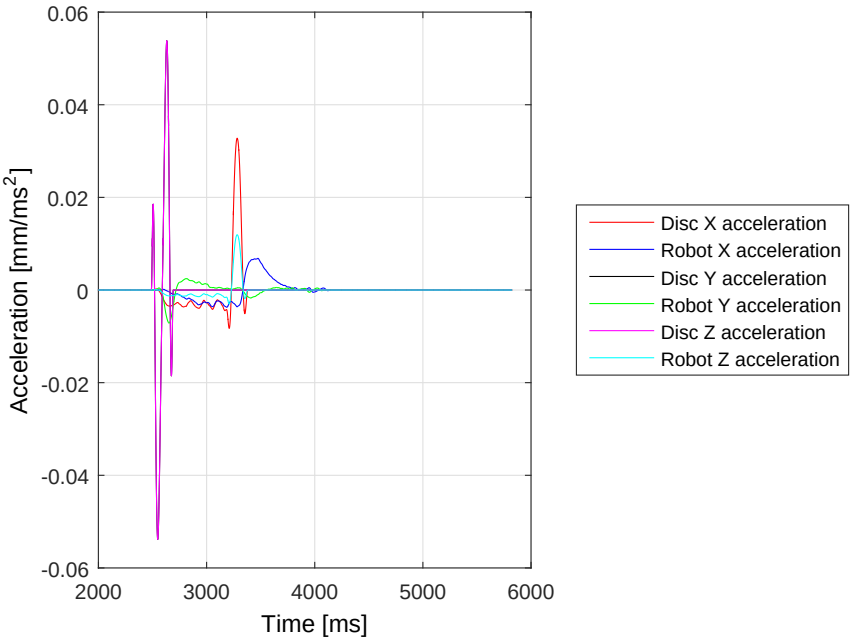


Figure A.6: Accelerations in all three directions

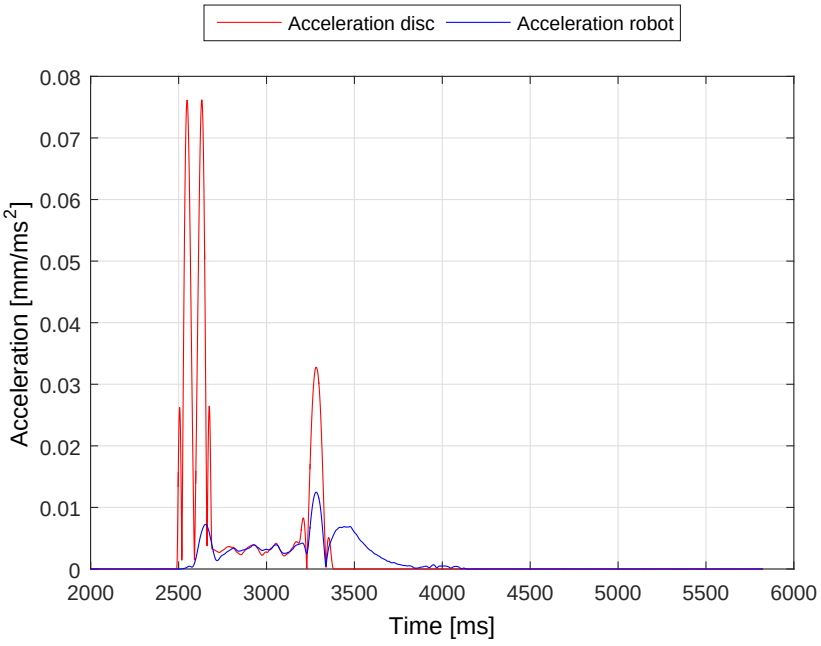
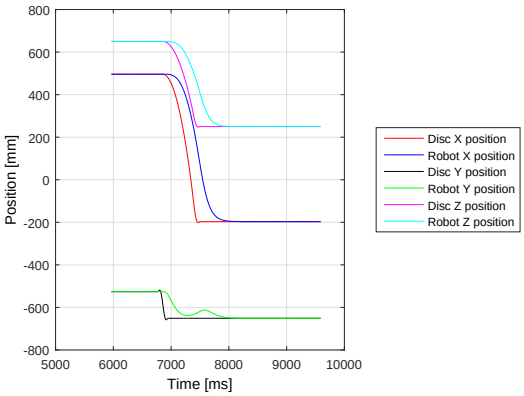
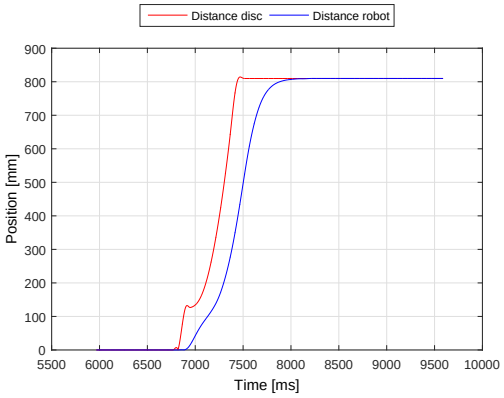


Figure A.7: Acceleration combined

A.3 30 Degrees

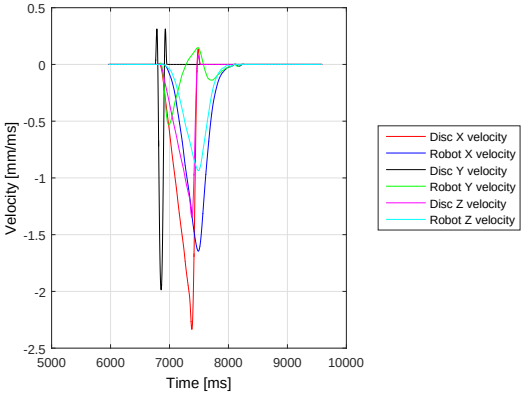


(a) Positions

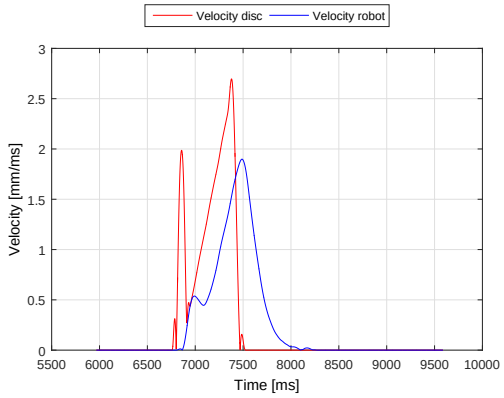


(b) Distance

Figure A.8: The positions and distance with 30 degree angle on the ramp

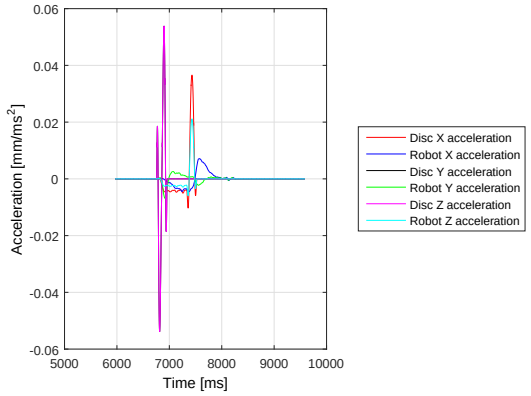


(a) Velocity in all three directions

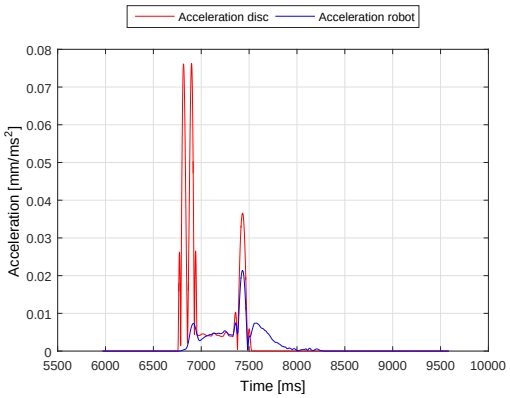


(b) Combined velocity.

Figure A.9: Velocities with 30 degree angle on the ramp



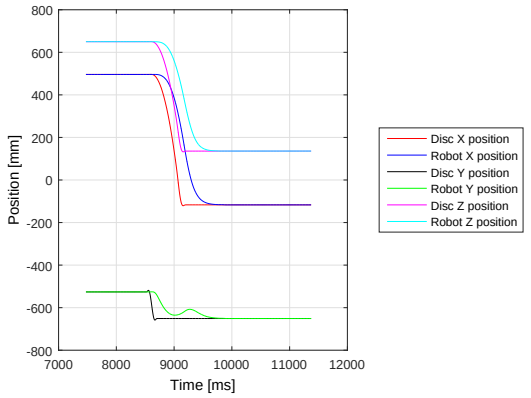
(a) Accelerations in all three directions



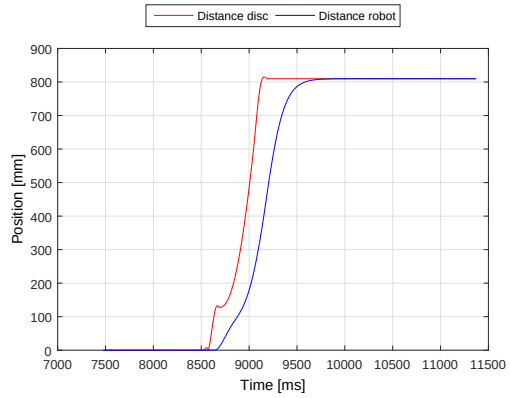
(b) Acceleration combined

Figure A.10: Accelerations with 30 degree angle on the ramp

A.4 40 Degrees

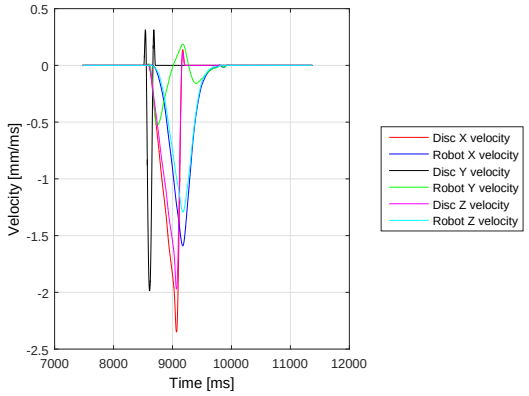


(a) Positions

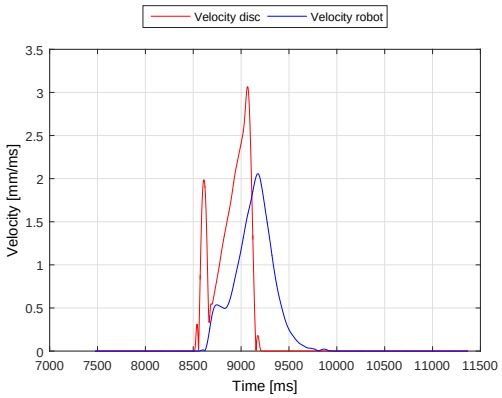


(b) Distance

Figure A.11: The positions and distance with 40 degree angle on the ramp

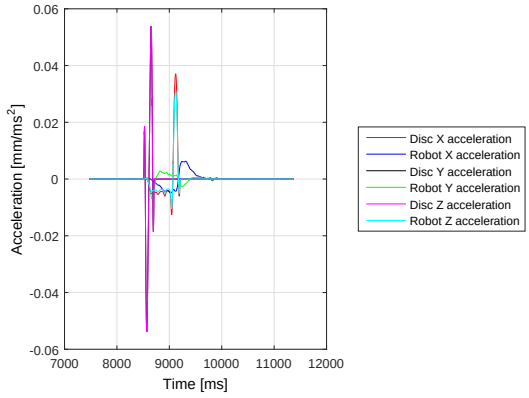


(a) Velocity in all three directions

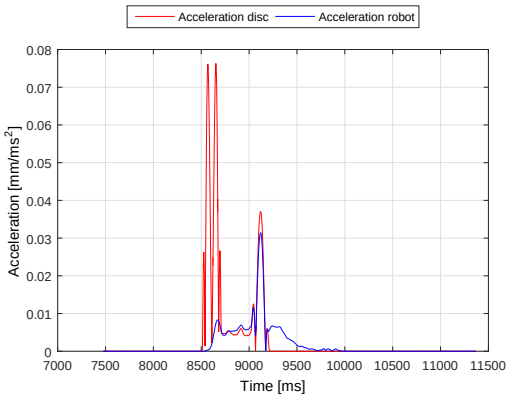


(b) Velocities combined

Figure A.12: Velocities with 40 degree angle on the ramp



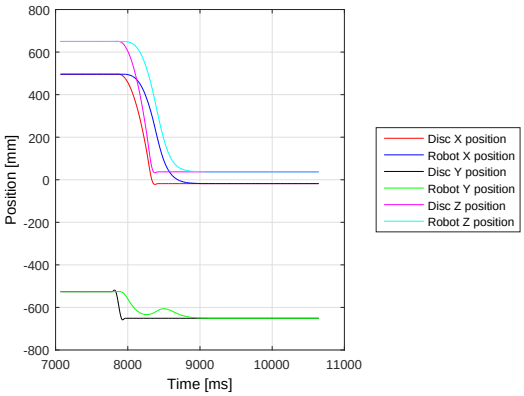
(a) Accelerations in all three directions



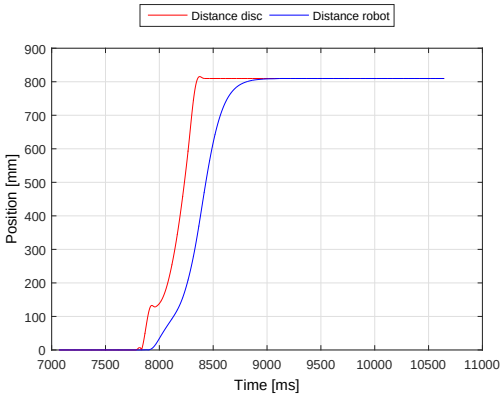
(b) Acceleration combined

Figure A.13: Accelerations with 40 degree angle on the ramp

A.5 50 Degrees

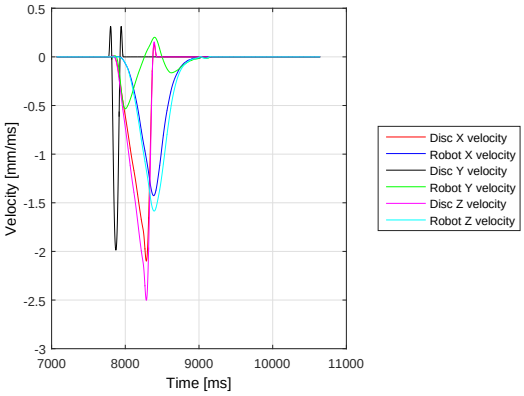


(a) Positions

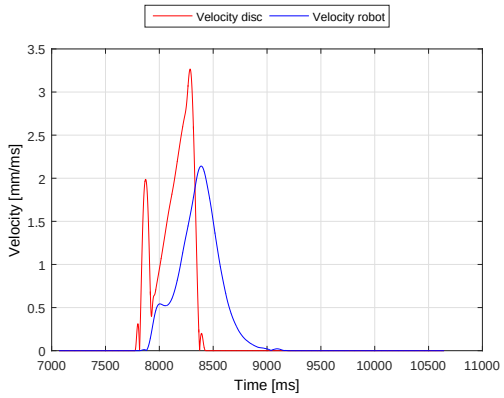


(b) Distance

Figure A.14: The positions and distance with 50 degree angle on the ramp

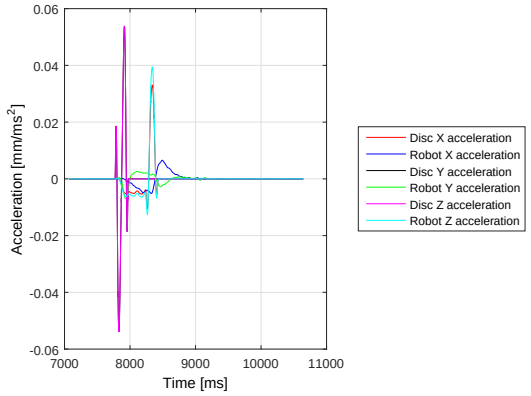


(a) Velocity in all three directions

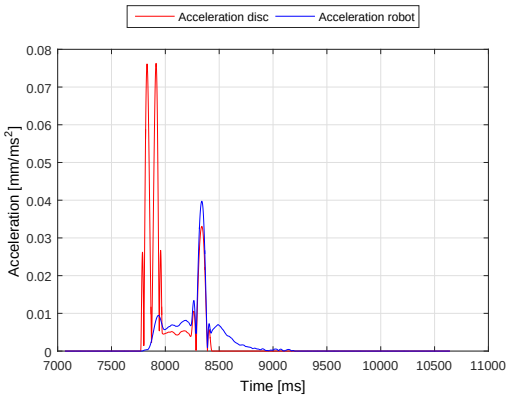


(b) Velocities combined

Figure A.15: Velocities with 50 degree angle on the ramp



(a) Accelerations in all three directions



(b) Acceleration combined

Figure A.16: Accelerations with 50 degree angle on the ramp

A.6 60 Degrees

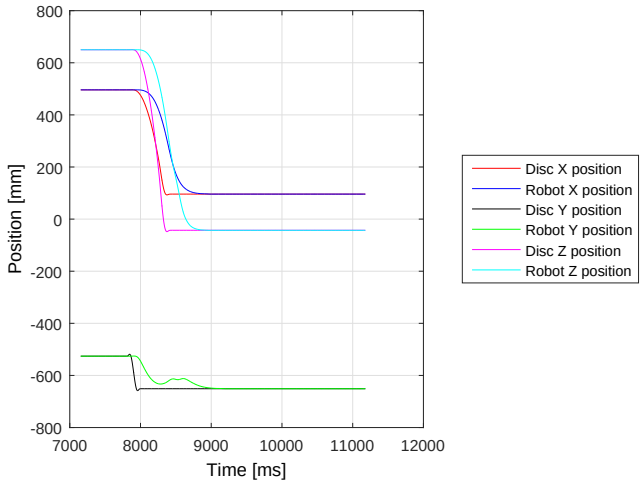


Figure A.17: Positions in all three directions

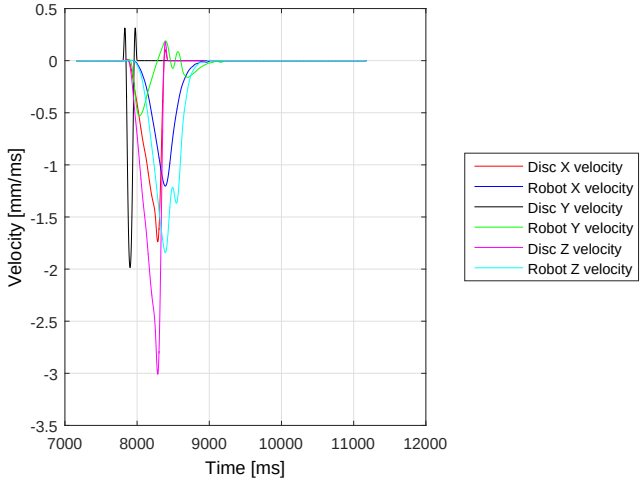


Figure A.18: Velocity in all three directions

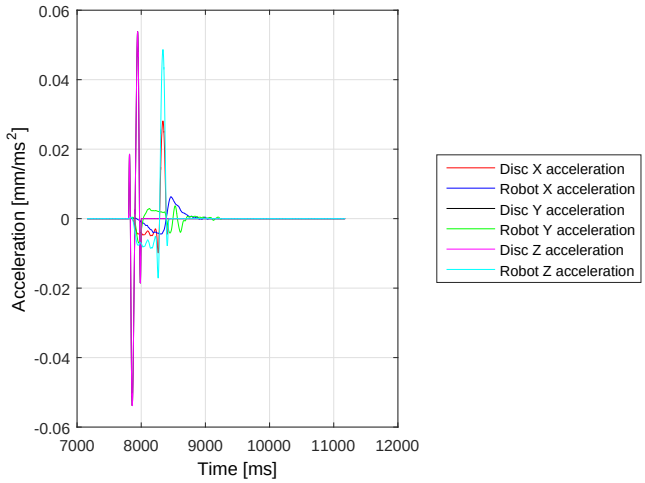


Figure A.19: Accelerations in all three directions

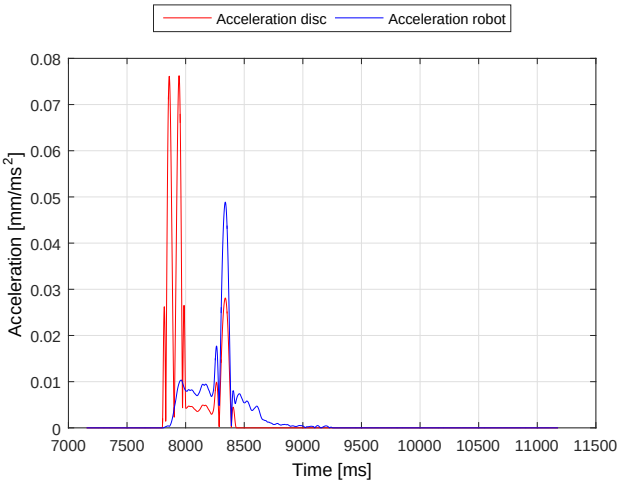


Figure A.20: Acceleration

This page is intentionally left blank.

B. Experiment with Constant Velocity

B.1 v100

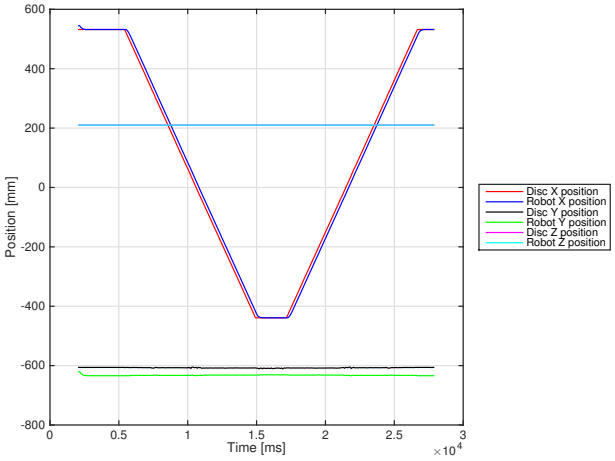


Figure B.1: Position in all directions, v100.

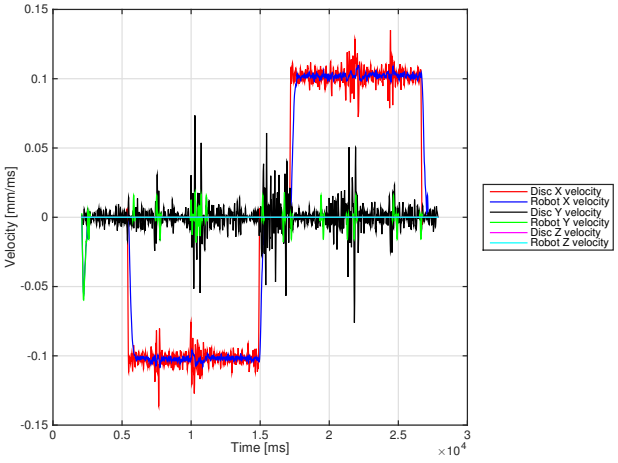
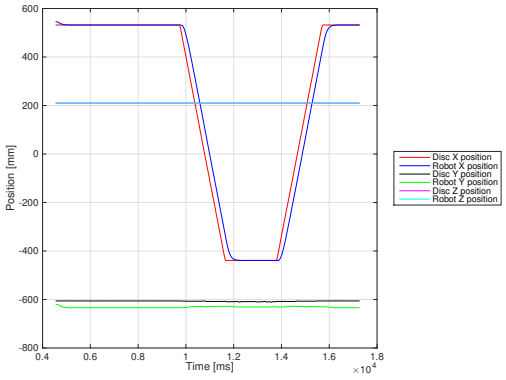
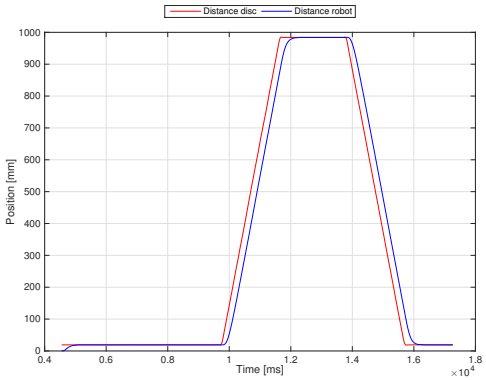


Figure B.2: Velocity in all directions, v100.

B.2 v500

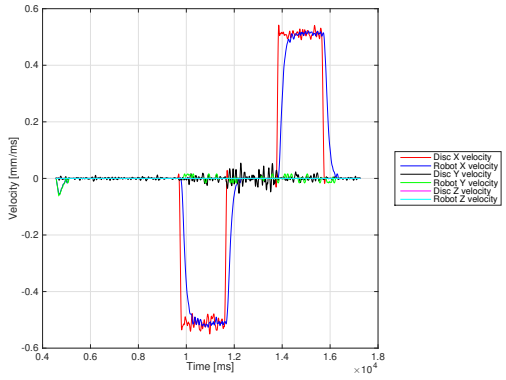


(a) Position in all three directions, v500.

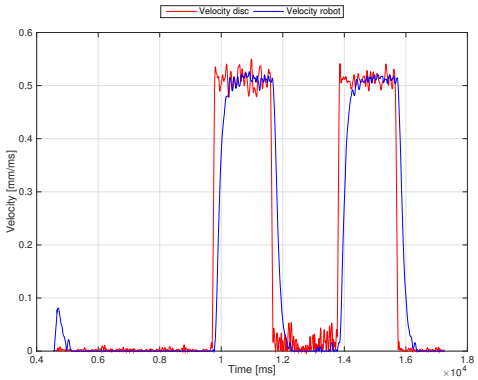


(b) Distance, v500.

Figure B.3: Position and distance for robot and disc, v500.



(a) Velocities in all three directions, v500.



(b) Combined velocity, v500.

Figure B.4: Velocities for robot and disc, v500.

B.3 v1000

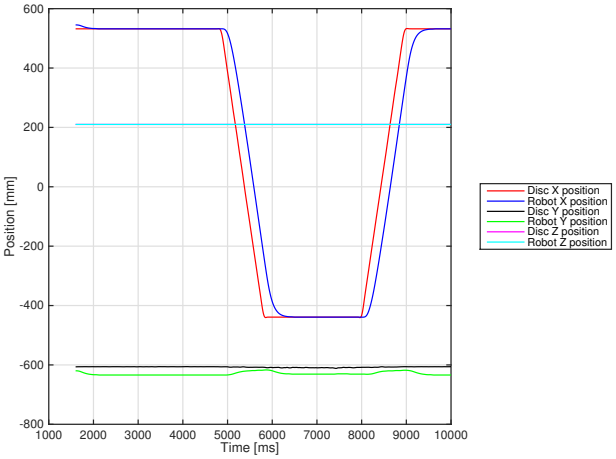


Figure B.5: Position in all directions, v1000.

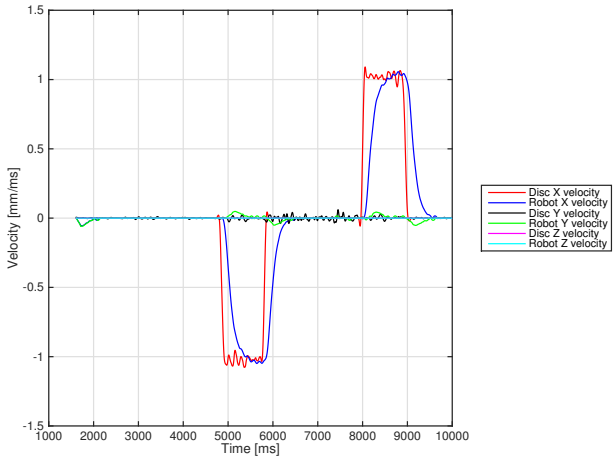
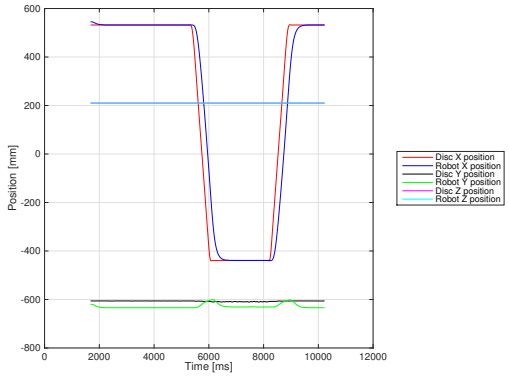
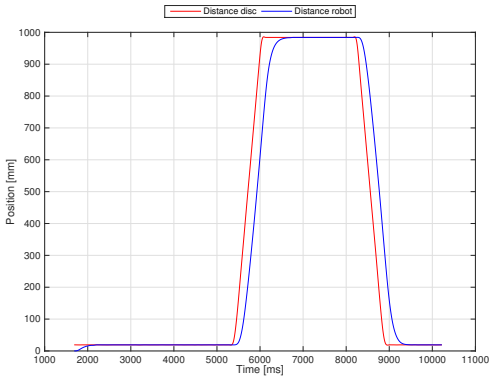


Figure B.6: Velocity in all directions, v1000.

B.4 v1500

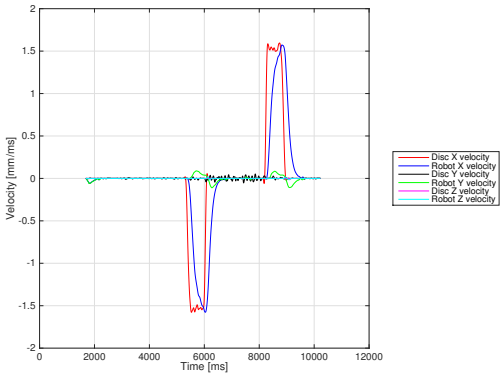


(a) Position in all three directions, v1500.

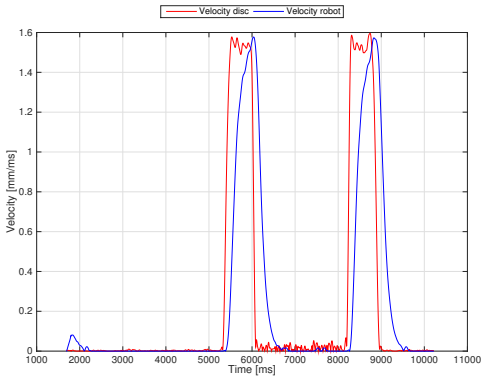


(b) Distance, v1500.

Figure B.7: Position and distance for robot and disc, v1500.



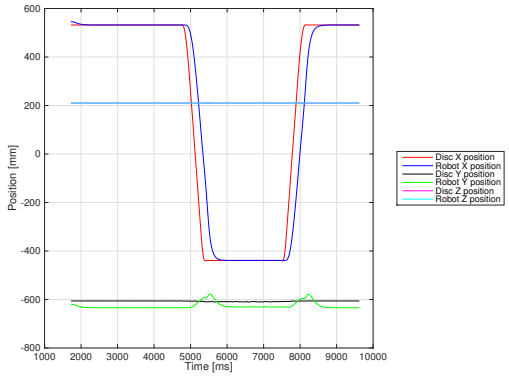
(a) Velocities in all three directions, v1500.



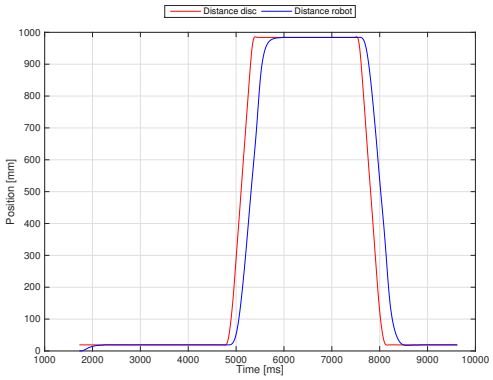
(b) Combined velocity, v1500.

Figure B.8: Velocities for robot and disc, v1500.

B.5 v2000

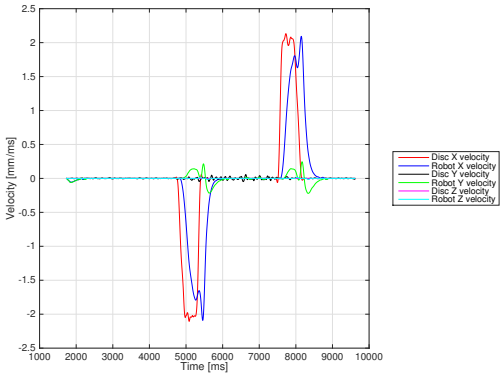


(a) Position in all three directions, v2000.

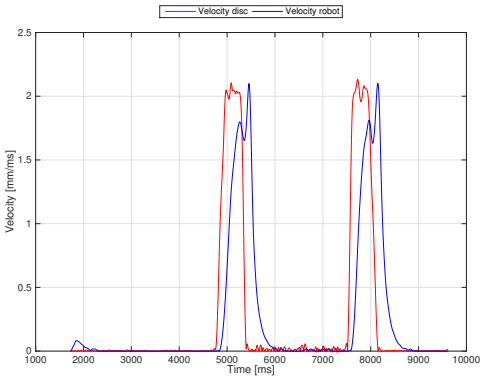


(b) Distance, v2000.

Figure B.9: Position and distance for robot and disc, v2000.



(a) Velocities in all three directions, v2000.



(b) Combined velocity, v2000.

Figure B.10: Velocities for robot and disc, v2000.

B.6 v2500

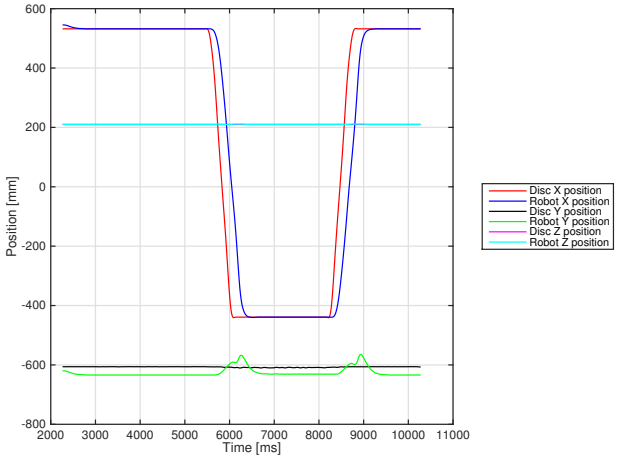


Figure B.11: Position in all directions, v2500.

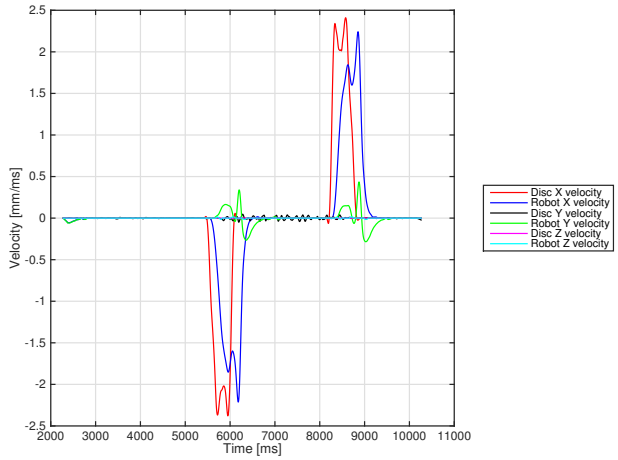


Figure B.12: Velocity in all directions, v2500.

C. Code

All the code written during this thesis are embedded in the document in the file "ExternallyGuidedMotion.7z". All code is place under the folder *src*.

- Opening the `ExternalGuidedMotion.sln` will open Visual Studio and display all the classes written in C#.
- The Matlab scripts are placed in the *Matlab* folder. `RotateTranslate.m` is the script which is used to find distance travelled, velocity and acceleration for the disc and robot. `DelayDistribution.m` plots the processing time in histograms.
- In the *Robotstudio* folder there is an "PackAndGo" file holding the program written in Robotstudio.
- The program for the camera is found in the *Cognex* folder.

All code is available on Github:
https://github.com/orjanmehre/ExternallyGuidedMotion.git