Eindhoven University of Technology Automated Football Table

R.J.M. Janssen

DCT 2009.045

Master's Thesis

Coaches: ir. J.J.T.H. de Best
dr. ir. M.J.G. van de Molengraft

Supervisor: prof. dr. ir. M. Steinbuch

Committee: ir. J.J.T.H. de Best
dr. ir. M.J.G. van de Molengraft
prof. dr. ir. H. Bruyninckx
prof. dr. ir. M. Steinbuch

Eindhoven University of Technology
Department of Mechanical Engineering
Control Systems Technology Group

Eindhoven, June 17, 2009

Summary

The Control System Technology group at Eindhoven University of Technology actively performs research on vision based control methodologies. One of the current activities is the design of an automated game of table football. By using computer vision as a feedback sensor the goal is to defeat a human opponent in a game of table football.

Currently the table is equipped with one electro-mechanically controllable rod, and one human controllable rod. These rods both hold three puppets that can be used to intercept the ball and kick it towards the opponent goal.

Because the hardware design of the table was already made in 2008, the work in this report focuses on the software design that was made to localize the ball and control the electro-mechanically actuated rod.

The software is designed according to the Unified Modeling Language notation. This notation creates a structural framework for the design of the software, from the general requirement to the basic functions.

The basic functions can be split up in two schemes. The first scheme is a *high level scheme* where the images from the camera are processed and the ball is localized. By using a *Kalman observer* the position and velocity of the ball are estimated. These estimates are used to determine the point of interception where the ball will cross the electro-mechanically controllable rod. By selecting one of the controllable puppets to intercept the ball, the according translation of the rod can be determined. This translation is send to the *low level scheme*. In this scheme the motors are controlled and the translation is executed. When the ball approaches the rod, the high level scheme sends a kick command to the low level scheme which then controls the motors to perform a kicking movement.

The results of the overal performance as well as some individual aspects of the automated table are presented. Although some points can be improved, the table shows that it is already capable of defeating an average skilled human player. When the table is fully optimized it is expected to be a worthy opponent, during a late-night lonely game of table football..

Samenvatting

De laatste jaren wordt er op de Technische Universiteit Eindhoven actief onderzoek gedaan naar het gebruik van computer visie in de regeltechniek. Een van de huidige projecten is een geautomatiseerde voetbaltafel. Het doel van deze tafel is om door het gebruik van computer visie een menselijke tegenstander te verslaan in een potje tafelvoetbal.

Op dit moment is de tafel voorzien van één elektrisch-mechanisch aangedreven stang en één normale stang. Op beide stangen zitten 3 poppetjes, die kunnen worden gebruikt om de bal te onderscheppen en terug te schoppen richting het doel van de tegenstander.

Omdat het mechanisch ontwerp van de tafel reeds in 2008 is gedaan, richt dit verslag zich voornamelijk op de software die is ontwikkeld om de bal te kunnen lokaliseren en om de elektrisch-mechanisch aangedreven stang te kunnen sturen.

Het ontwerp van de software wordt gedaan met behulp van de Unified Modeling Language notatie. Deze notatie maakt een gestructureerd ontwerp van de software mogelijk, waarbij alle aspecten van het ontwerp worden beschouwd.

De geïmplementeerde functies worden opgesplitst in twee schemas. In het eerste schema worden de beelden van de camera verwerkt, en wordt de bal gelokaliseerd. Vervolgens worden met behulp van een *Kalman observer* de positie en snelheid van de bal geschat. Deze schattingen worden gebruikt om te kunnen bepalen waar de bal de geautomatiseerde stang zal snijden. Door een van de drie poppetjes te selecteren om de bal te onderscheppen, kan worden berekend wat de translatie van de stang moet zijn. Deze translatie wordt vervolgens naar het tweede schema wordt gestuurd. Dit schema stuurt de motoren aan, en zorgt dat de translatie wordt uitgevoerd. Wanneer de bal bijna bij de stang is, wordt er vanuit het eerste schema ook een opdracht gegeven om te schieten. Dit commando wordt dan vervolgens weer uitgevoerd door het tweede schema, dat zorgt dat de stang een schoppende beweging maakt.

Op het eind van het verslag worden zowel het algemeen resultaat alsook sommige individuele aspecten van de tafel gepresenteerd. Alhoewel sommige puntjes nog voor verbetering vatbaar zijn is de tafel nu al in staat om een gemiddelde speler te kunnen verslaan. Als de tafel volledig geoptimaliseerd is, is de verwachting dat het een geduchte tegenstander zal zijn gedurende een eenzaam nachtelijk spelletje tafelvoetbal..

Contents

Eindhoven University of Technology Automated Football Table

Chapter 1

Introduction

1.1 Background

For many years vision based control has become a common used control method in industry and academic research areas. One of the first concepts that showed the possibilities of using vision as a way to provide sensory feedback is demonstrated in [1]. In this setup a robotic arm was used to place a square prism block in a square basket, where camera vision was used to determine the relative position of the block to the basket. More modern day applications that use camera vision as a feedback sensor can be found in applications such as conveyor belt control [2], the Da Vinci tele-operation surgical system [3], guidance of small unmanned aircraft [4] and in academic research projects such as the RoboCup robotic soccer domain [5].

In the last few years Eindhoven University of Technology is actively investigating the possibilities, drawbacks and benefits of using computer vision to control an electro-mechanically actuated setup. As a case-study, a football table is acquired to function as a platform on which vision based control methodologies can be tested. The overal goal of this project is to design the table such that it will be capable of combining high speed computer vision with low level control to defeat a human player in a game of table football. Several other project groups have already been working on the development of an automated football table, which all used different methods to track the ball. The University of Adelaide [6] has used a laser grid to detect the ball. Because the grid is mounted exactly below the feet of the puppets, the ball is the only object that can breach it. The Georgia Institute of Technology [7] designed an automated football table that uses an overhead camera to detect the ball. By designating different colors to all the objects in the field, color segmentation could be used to distinguish the ball from the puppets. An automated football table that eventually became commercially available was the KiRo project [8] which was later developed to StarKick [9]. In this project a camera was mounted underneath the table. By replacing the field with a transparant plate, they could track the ball from underneath without having to deal with any occluding objects such as the puppets.

Even though all of these projects worked well and some of them were even able to defeat professional players, the implementations made in these projects also have their drawbacks. The laser grid used in the Adelaide project can only detect the ball when the ball is moving at ground level. When the ball bounces it cannot be localized by the grid. This also goes for the camera mounted underneath the table, as was used in the KiRo and StarKick projects. By mounting an overhead camera and using color segmentation in theory the ball can be constantly found. However, processing 24-bit RGB color images instead of 8-bit monochrome images triples the computational effort that should be performed by the workstation. Due to the highly discontinuous character of the ball's trajectory, tracking of the ball should be based on fast updates of the ball's position. Therefore processing of the images should be performed on a very high sample rate, which can only be obtained by processing 8-bit monochrome images.

1.2 Problem statement and objective

In regard of the issues described above, there are several aspects in this project that need to be investigated. These aspects are,

- segmenting the ball from its environment,
- tracking of the ball in spite of its discontinuous trajectory,
- the performance of the controllable rod,
- the accuracy of intercepting the ball.

To tackle these issues first a proper hardware design has to be made. A software algorithm that detects and tracks the ball has to be developed and the rod that intercepts the ball has to be controlled in a stable and safe manner. Because the biggest part of the hardware design was already dealt with in [10] and the fact that the current setup exists of only one automated rod, the main focus of this report will be on segmenting the ball, tracking the ball and the real-time control of a single rod. Therefore the objective of the work described in this report is

Design and implement a vision based control strategy that detects the ball and controls a single rod in real-time to be able to defeat a human opponent in a game of table football.

To fulfill this objective, first the necessary hardware that is capable of detecting the ball and moving the rod has to be configured and properly installed. A software design has to be developed, that translates the general objective into practical implementations. In

these practical implementations an image processing algorithm has to be developed that is able to use a camera to localize the position of the ball from the acquired images. The trajectory of the ball has to be estimated from which a prediction can be done where the interception point on the rod must be. A low level controller is designed that uses the onboard encoders to stabilize and control the rod such that one of the puppets is able to intercept and return the approaching ball. Finally the table has to be fully tested and presented to the general audience.

1.3 Outline

In this report first a brief overview of the used hardware components and their functionality will be explained. The software design will be discussed in Chapter 3. The basic functions that were used in the software design are discussed in Chapter 4. Chapter 5 will present the results that where obtained when the table was tested. Finally the conclusions and recommendations will be presented in Chapter 6. In the Appendix there will be a startup guide that describes how to use the setup and how to properly startup a game of automated table football.

Chapter 2

Hardware design

Because the hardware design was already discussed in [10], this chapter will only give a brief overview of the setup.

As the table is meant for professional table football play, it was decided that after any alterations the table should still comply to the official USTSA regulations as defined in [11]. To have a platform that complies with these regulations a professional football table is acquired. A steel frame is placed on top of the table, on which a camera is mounted. One side of the table is equipped with electro-mechanically controlled rods. This way the puppets can be moved towards a predetermined position, and they can also lift their legs or perform a kicking movement. A schematic overview of the whole setup is depicted in Fig. 2.1.

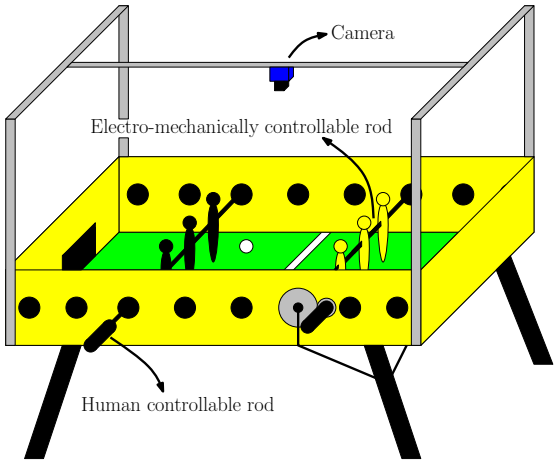


Figure 2.1: Schematic overview of the table.

The high speed ethernet camera is mounted above the field. These images are fed into a dual core 2GHz work station, running a Linux low-latency kernel. At the workstation all

relevant information is extracted from the images, and the workstation also handles the communication with the data acquisition equipment which allows to control the motors. An overview of the connections between the hardware components and their corresponding dataflow is depicted in Fig. 2.2.

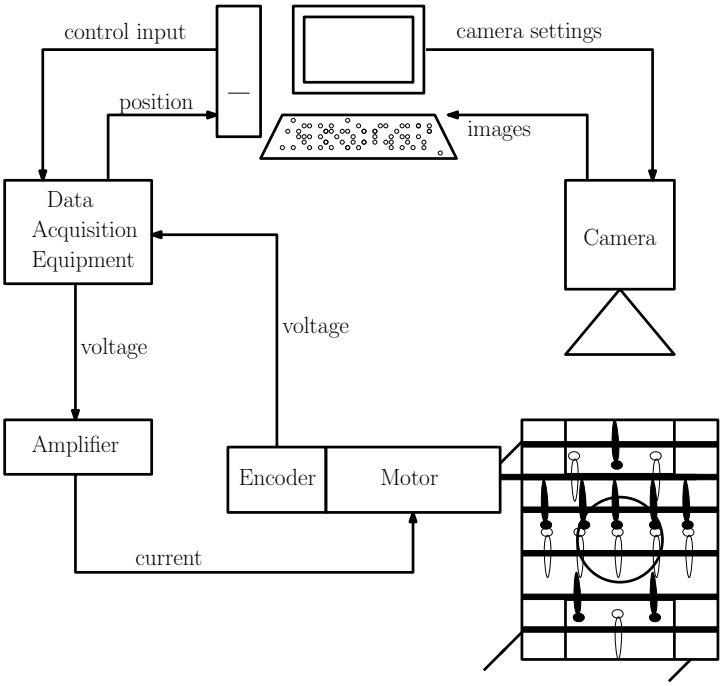


Figure 2.2: Hardware components and dataflow.

Now that the setup has been briefly explained, in the next chapter the software design will be discussed.

Chapter 3

Software design

In accordance with the described goal in Chapter 1, he general objective of the software is to control one rod to intercept the ball and kick it back towards the human goal. To perform this objective, a software design is developed that translates the general objective into basic functions. These basic functions then interact with the hardware components to perform their tasks. The software design is developed by using the UML (Unified Modeling Language) notation [12]. With this notation it is possible to visualize the high level control architecture and the way this architecture interacts with the low level hardware components. Schematically the software design can be split up in three levels. These are the *general requirement*, *analysis* and *implementations* that are made. This is depicted in Fig. 3.1.

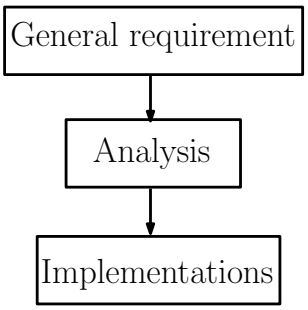


Figure 3.1: Software design levels.

In the next sections the levels will be explained in detail.

3.1 General requirement

First the general requirement of the automated football table has to be defined. This general requirement is to *defeat the human opponent in a game of table football*. How this requirement can be achieved is visualized in a *use-case diagram*, see Fig. 3.2.

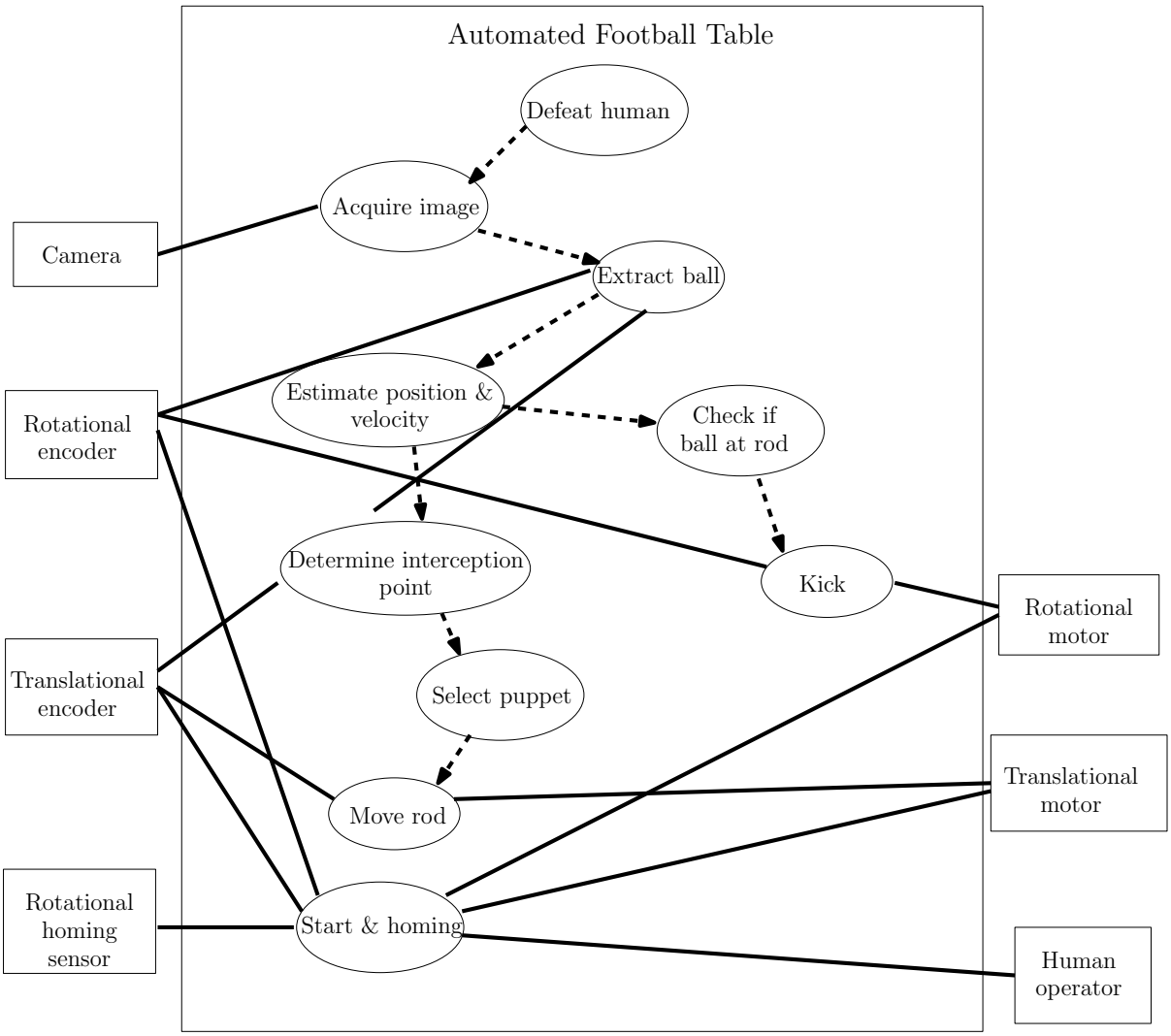


Figure 3.2: Use-case diagram of the football table.

In this diagram the general requirement is stated and how this requirement can be achieved. The requirement is to defeat a human opponent. To achieve this requirement, first an image has to be captured and processed. The ball has to be localized from the image and an estimation has to be made where the interception point must be on the automated rod. One of the puppets on this rod will be selected to intercept the ball, and the according translation for the rod can be determined. The translation is then performed by interacting with the hardware components. According to the position of the ball also has to be checked if it is possible to perform a kick. If a kick can be performed this command has to be executed by interacting with the hardware. When a human operator starts up the system, first the system has to execute a homing operation to properly calibrate its parameters.

3.2 Analysis

How to achieve the general requirement can be analyzed by developing a stepwise approach of actions that have to be performed. These actions can be listed in an *activity diagram*. This diagram is presented in Fig. 3.3.

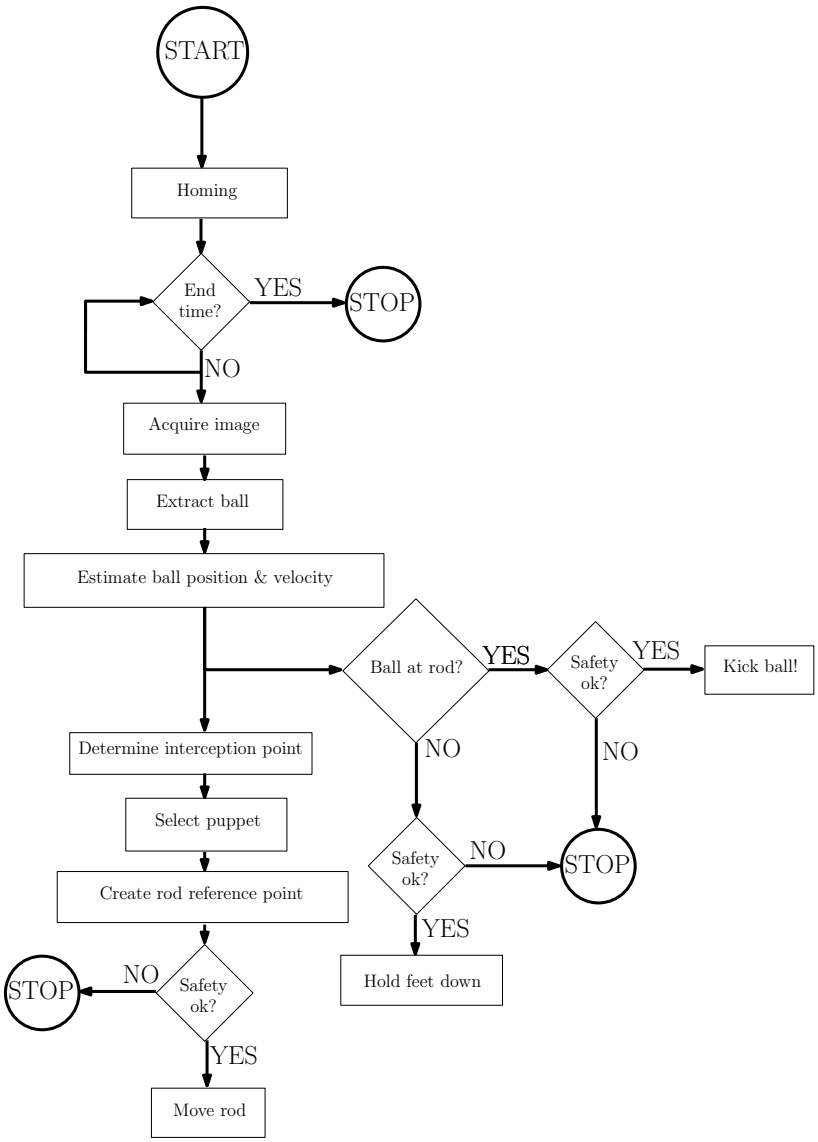


Figure 3.3: Activity diagram of the football table.

This diagram describes how the actions are executed in chronological order. In this diagram an action is depicted by a square block. Decisions are depicted by a square prism block, and a startup or possible termination is depicted by a circle.

3.3 Implementations

The final step is to describe the relationship between the strategy that should be followed, the basic functions and the hardware components. This relationship can be described in a *component diagram*, see Fig. 3.4.

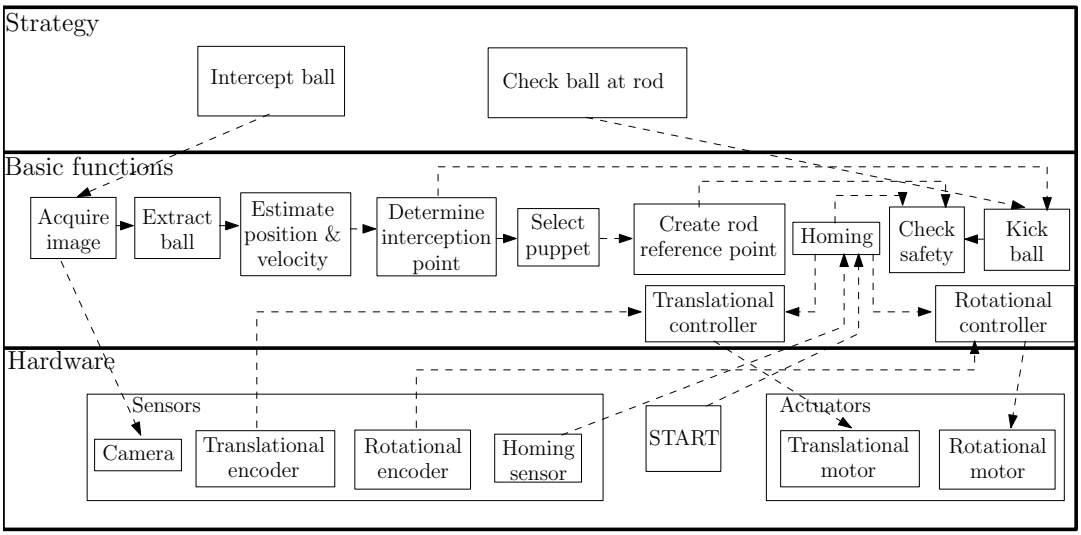


Figure 3.4: Component diagram of the football table.

This diagram describes all the sensors, actuators and software functions that are used. These components have to be mapped to a software layout that is capable of successfully performing each function, while interacting with the hardware.

Mapping of components

In the component diagram presented in Fig. 3.4 several components are directly related to motion control aspects of the setup. These components include the safety measures, the homing functions, the homing sensor, the controller, the encoders and the actuators. These components have to be used at a very high rate (in the order of kHz's [13]) to be able to successfully perform their task. Therefore these components are gathered in one software scheme. Because this scheme contains only low level motion control components, this scheme is called the *low level scheme*.

The remaining components are responsible for the image processing and estimation of the references that need to be performed. These components can also be gathered in one scheme. Because in this scheme the higher level calculations and decisions are made, this scheme is called the *high level scheme*.

The schemes share information through a specific form of Inter Process Communication, called *shared memory* [14].

How the components are divided over the two different schemes is depicted in a *deployment diagram*, see Fig. 3.5.

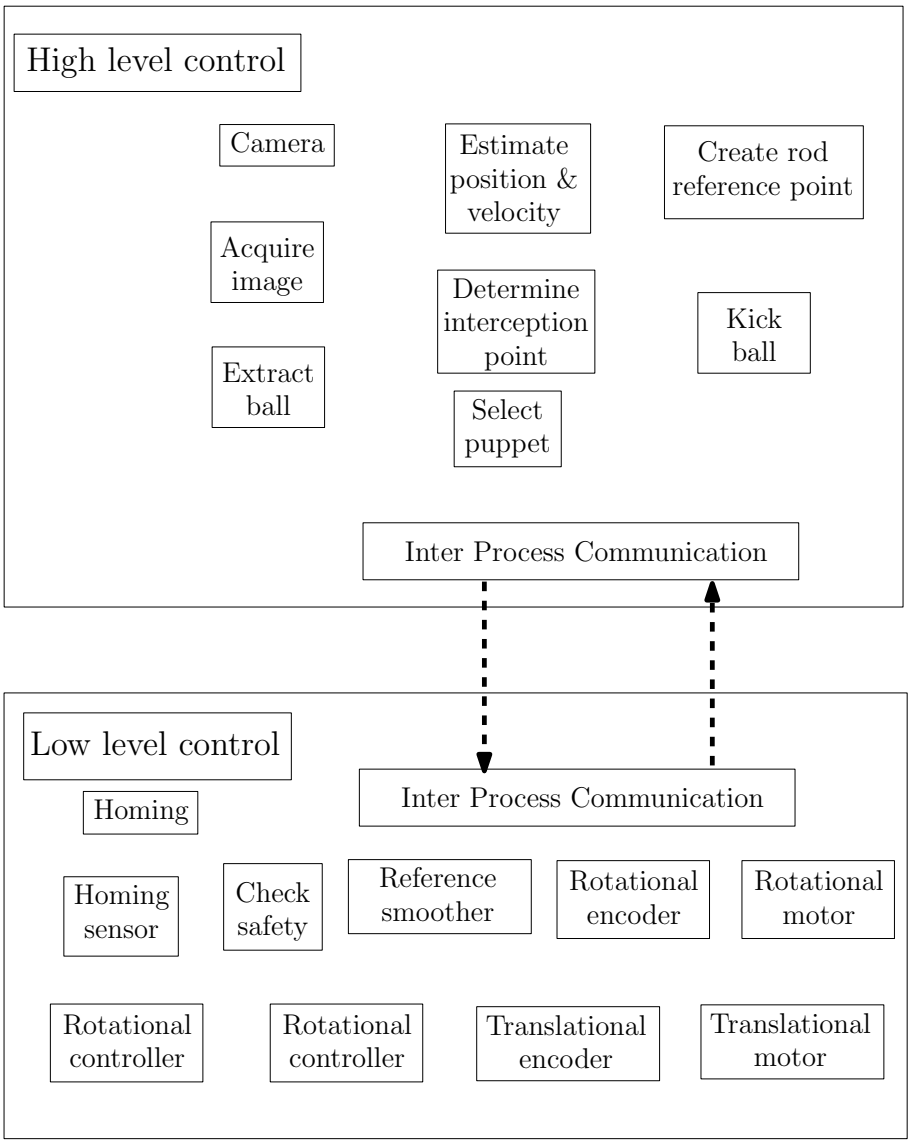


Figure 3.5: Deployment diagram of the football table.

Each of the functions that are used in the deployment diagram are detailly explained in the next chapter.

Chapter 4

Basic functions

The previous chapter explained that the hardware components and the software functions are divided over two different schemes. A *high level scheme* in which the images are processed and the references for the rod are created, and a *low level scheme* in which these references are executed. This chapter is divided in two sections in which the basic functions of both the high level and the low level schemes are discussed.

The coordinate frame that is used throughout this chapter is depicted in Fig. 4.1.

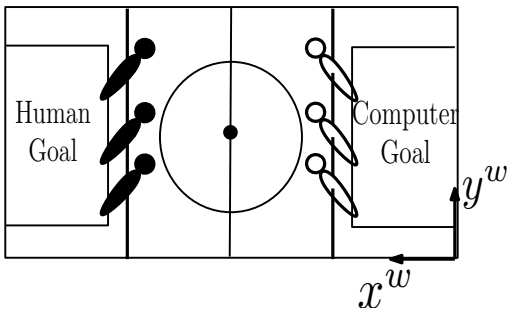


Figure 4.1: Coordinate system.

4.1 High level scheme

In the *deployment diagram* Fig. 3.5 from the previous chapter the high level scheme was visualized. This scheme calculates the translational movement of the rod and determines if the rod should perform a kick. The basic functions described in the high level scheme are depicted in Fig 4.2.

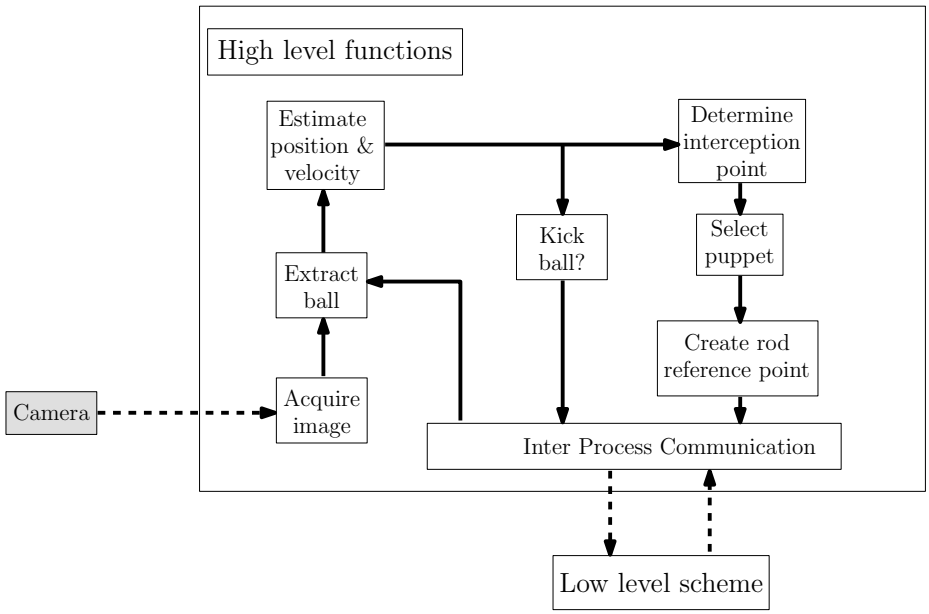


Figure 4.2: High level control scheme.

The functions described in the high level scheme mainly consist of image processing algorithms and the implemented estimations for the position and velocity of the ball. In this section these functions are individually discussed.

4.1.1 Acquire image

To acquire images that are suitable for processing, the camera has to be configured. This configuration depends on some of the research related objectives that were stated in Chapter 1. The referred objectives are

- the table should still comply to the official USTSA regulations [11] and therefore the layout and coloring of the field may not not be changed,
- the camera is mounted above the field,
- the camera captures 8-bit monochrome images.

Furthermore the camera has a resolution of 657×446 [pix]. On this camera a 6 [mm] lens is mounted so that the entire field can be captured. A typical image captured by the camera is depicted in Fig. 4.3.

The camera is set to capture images at its maximum framerate f_c which is 200 [Hz]. The maximum exposure time t_{max} can then be set according to

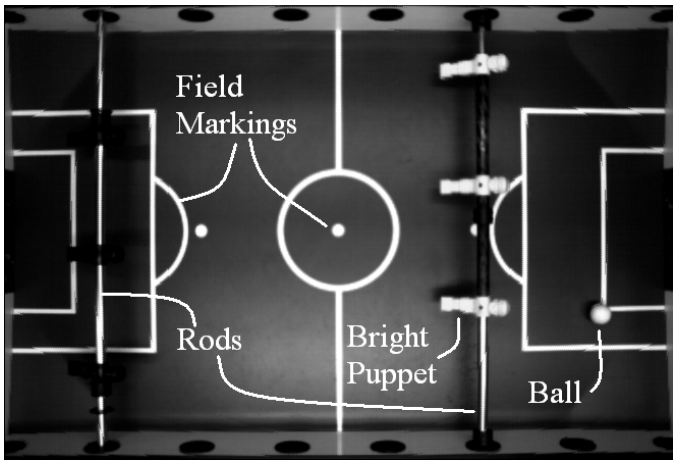


Figure 4.3: Picture of the field.

$$t_{max} = \frac{1}{f_c} \tag{4.1}$$

Therefore the exposure time t_{max} is set at 5000 [μs].

4.1.2 Extract ball

This section describes how the ball is extracted from the acquired images and how the position of the ball is determined.

Region of interest

The first step in the extraction algorithm is to crop the captured images to a region of interest, or ROI, in order to remove redundant image data that should not be processed. Processing a full frame image demands more computation time from the computers CPU, and in order to design a real-time application this computation time should be kept to a minimum.

Localizing the ball in a ROI instead of in a full frame image can however lead to a situation where the ball is located on the field outside the bounds of the ROI. In that case the ball cannot be localized.

When the ball cannot be located in the ROI this can have 4 possible causes

- the beginning of a game. The ball will reappear at a random position,

- a goal was scored. The ball will reappear at a random position,
- the ball went out of field. The ball will reappear at a random position,
- the ball is occluded by a rod or a puppet. The ball will reappear near the position it was lost.

To cope with these 4 possibilities the position of the ROI is programmed to remain static for 5 [s] when the ball is lost. This is approximately the time it takes to either get a new ball from the dispenser or to pick up and return the ball that went out of field. After this period the entire field is scanned for the ball by shifting the ROI over the field.

In case of an occlusion, the position of the ROI initially remains static. Therefore the size of the ROI must be defined such that when the ball moves at its maximum velocity it is still able to reappear in the static ROI, even if it is occluded in one frame. The only object that can fully occlude the ball, is the upper torso of a puppet when the puppet is placed in a horizontal position. This is schematically depicted in Fig. 4.4.

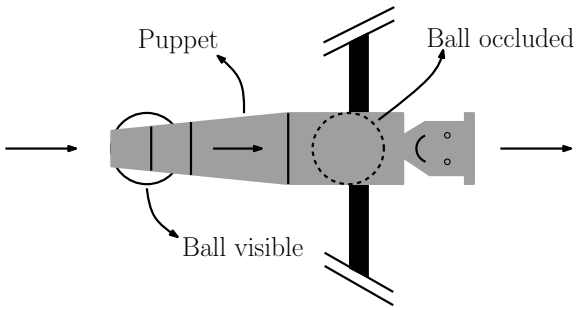


Figure 4.4: Horizontal puppet occluding the ball.

If the occluding puppet is placed in a horizontal position it is assumed that the ball rolls underneath the puppet with a maximum velocity V_{max}. To determine the size for the ROI that allows to locate the fast moving ball without occlusions, first only half the width of the ROI in the x^w direction is considered. The minimum size for half the ROI $r_{min}^{x^w}$ depends on V_{max}, the frame rate f_i and the pixel resolution P_r. If an outer pixel of the ball is present in the ROI, the ball can be located. Therefore half the radius of the ball can be subtracted according to

$$r_{min}^{x^w} = \frac{V_{max}^{x^w} P_r}{f_i} - \frac{r_b}{2} \tag{4.2}$$

However, when an occlusion occurs the ball can be lost. In chronological order this situation can be depicted as follows.

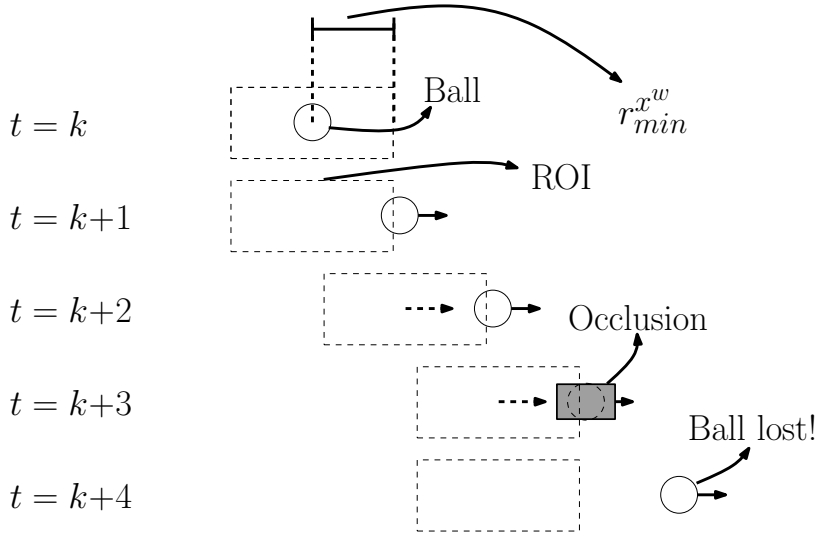


Figure 4.5: Moving of the ball and lagging ROI.

At $t = k$ the ball starts moving. At $t = k+1$ the ROI starts following the ball and because no prediction is used, the position of the ROI always has one sample lag. At $t = k + 3$ an occlusion prevents the ball to be found in the ROI. Therefore the ROI remains static, and the ball is lost. To find the ball in the next frame the width of the ROI is made twice as large, except for the ball radius. This is depicted in Fig. 4.6.

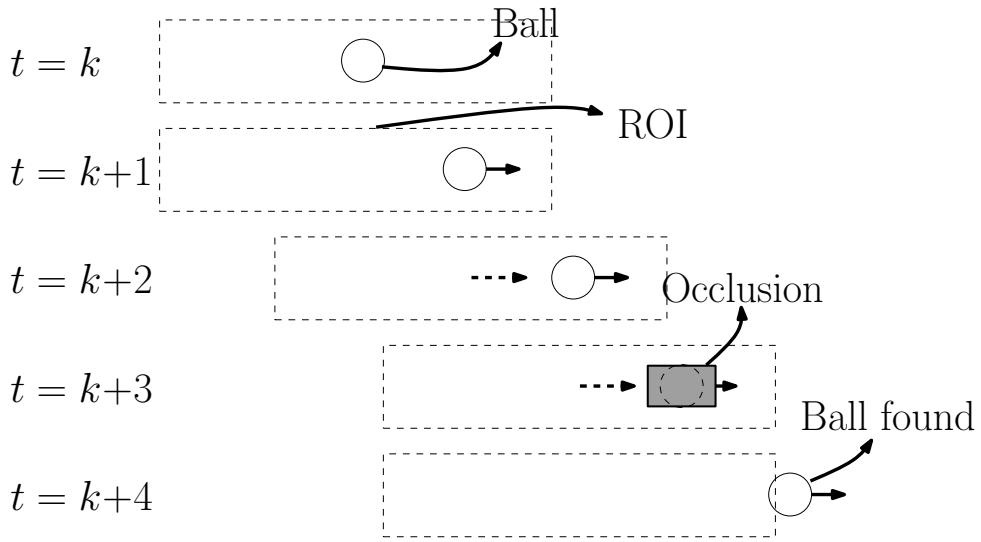


Figure 4.6: ROI with double width.

Because there can be made no distinction if the ball is moving in the positive or in the negative x^w direction the width of the ROI has to be doubled again, this time including the ball radius. Therefore the total width for the minimum ROI in the x^w direction becomes

$$R_{min}^{x^w} = 4\frac{V_{max}^{x^w} P_r}{f_i} - r_b, \tag{4.3}$$

where P_r is the pixel resolution of 571.7 [pix/m] and r_b is the radius of the ball which is 10 [pix]. f_i is the rate at which the image processing algorithm runs and $V_{max}^{x^w}$ is the maximum ball velocity in the x^w direction. Tests have shown that the maximum ball velocity $V_{max}^{x^w}$ can reach speeds up to 5 [m/s] and $V_{max}^{y^w}$ up to 2.5 [m/s]. This implicitly means that the size of the ROI in the x^w direction should be twice the size of the ROI in the y^w direction, because the maximum ball speed is twice as high in this direction.

Therefore

$$R_{min}^{x^w} = 2R_{min}^{y^w}. \tag{4.4}$$

From (4.3) it can be assumed that the maximum allowable ball speed can be increased by increasing the size of the ROI. There is however another factor that limits the size of the ROI. This is the maximum computational load the workstation can handle. This maximum load is determined by the size of the ROI and the image processing rate f_i.

Computational load

With respect to the maximum computational load the size of the ROI limits the maximum allowable image processing rate f_i that can be used before the real-time algorithm starts missing timer interrupts. Missing timer interrupts means that the workstation cannot process the images at the desired rate f_i, because either the rate f_i or the size of the ROI is set too high. A model is assumed to describe this relationship. It is assumed that there is an inverse relationship between the size of the ROI and the maximum allowable image processing framerate f_i. Also a constant factor is assumed, which describes the running processes that are not influenced by the size of the ROI. By experiments this model must be validated. The model that is assumed is

$$R_{min}^{x^w} \times R_{min}^{y^w} = \frac{C_1}{f_i} + C_2. \tag{4.5}$$

Before validating this model first (4.5) is combined with(4.4) to form a model with only two variables. This leads to the following equation.

$$f_i = \frac{C_1}{\frac{1}{2}{R_{min}^{x^w}}^2 - C_2}. \tag{4.6}$$

To validate the model a total of 6 experiments is carried out, where for different values of $R_{min}^{x^w}$ the maximum f_i was set. In all experiments the CPU was fully loaded. The results of these experiments and the model of (4.6) are depicted in Fig. 4.7.

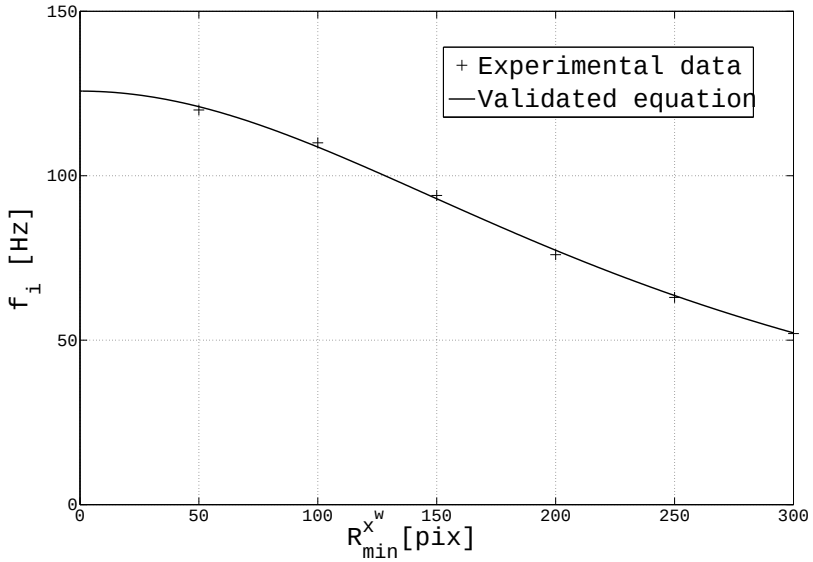


Figure 4.7: $R_{min}^{x^w}$ against f_i at maximum computational load.

In Fig. 4.7 the values that were used for C_1 and C_2 were obtained using an averaging optimization between the measurement points.

$$C_1 = 4.02e6 \text{ [pix}^2\text{/s]}, \qquad C_2 = -3.20e4 \text{ [pix}^2\text{]}, \tag{4.7}$$

By combining the validated model from (4.6) with (4.3) the minimum width for the ROI in the x^w direction can now be made dependent of V_{max} only, according to the quadratic equation

$$\frac{2V_{max}^{x^w}P_r}{C_1}(R_{min}^{x^w})^2 - R_{min}^{x^w} = \frac{4V_{max}^{x^w}P_rC_2}{C_1} + r_b. \tag{4.8}$$

Solving this quadratic equation for a $V_{max}^{x^w}$ of 5 [m/s] gives a $R_{min}^{x^w}$ of 93.3 [pix] and according to (4.4) a $R_{min}^{y^w}$ of 46.6 [pix]. From now on a ROI of 100×50 [pix] will be assumed.
With these dimensions for the ROI, according to (4.6) the rate of f_i can be determined to be 110.6 [Hz]. This rate puts the computational load for the CPU at 100%. Because it should still be possible to move a mouse or open a window on the workstation, the computational load is lowered by setting the rate f_i at 100 [Hz]. This results in an average CPU load of 93%. Because the frame rate f_i is lowered, the maximum allowable velocity for the ball now becomes 4.6 [m/s] for the x^w direction and 2.3 [m/s] for the y^w direction.

Removing static objects by creating a mask

In the football table there are objects present that do not move over time. These static objects include the field lines, the field dots and the rods that hold the puppets (but not the puppets themselves). These objects look similar to the ball and should therefore be disregarded when searching for the ball. This can be done be creating a static mask. This mask can be subtracted from the captured image.

The center of the field is somewhat brighter than the rest of the field. This is due to the fact that the field is not uniformly illuminated. This illumination difference also has to be included in the mask. The mask that contains all the static objects and the illumination differences is depicted in Fig. 4.8.

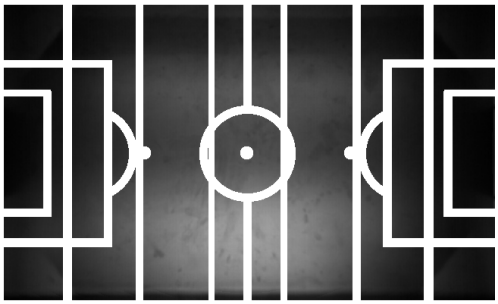
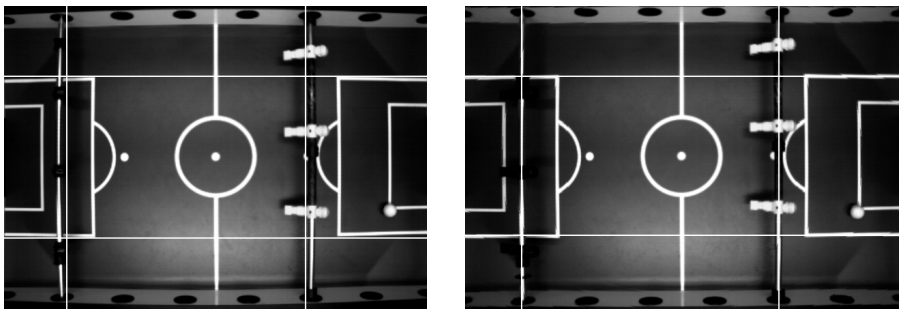


Figure 4.8: Mask that removes static objects.

Undistorting the image

To be able to properly subtract the static mask from the captured image, the captured image first has to be undistorted. To solve for this distortion, a camera calibration is carried out using the Camera Calibration Toolbox for Matlab [15]. The resulting distortion coefficients are then used to restore the images as shown in Fig. 4.9.



(a) Original image. (b) Restored image.

Figure 4.9: Original and restored image. Straight reference lines in white.

Now the mask can be successfully subtracted from the image. Image undistortion is also necessary to find the correct coordinates of the ball on the field, and for the masking algorithm that will be explained in the next section.

Masking yellow puppets

As can be seen in Fig. 4.3 the puppets on the field have two different colors. The bright puppets have to be masked, because the intensity values of these puppets are close to those of the ball. Therefore the bright puppets are chosen to be the electro-mechanically controlled puppets, so that through the onboard encoders their position and orientation will be available for processing. With this data the position and orientation of the puppet relative to the camera can be calculated and through a perspective projection a 2D mask of each of these puppets can be determined. For this a 3D scan of the contour of a puppet is created by using a Magnetic Resonance Imaging, or MRI [1], scanner. To perform the perspective projection the pixel coordinates D_i^p in the 3D scan have to be transformed to 3D pixel coordinates in the camera frame D_i^c. The formula that describes this transformation is:

$$\underline{D}_i^c = \mathbf{R}_s^r(\underline{O}_p^s + \underline{D}_i^p) + \underline{O}_r^c \tag{4.9}$$

A corresponding graphical interpretation is given in Fig. 4.10.

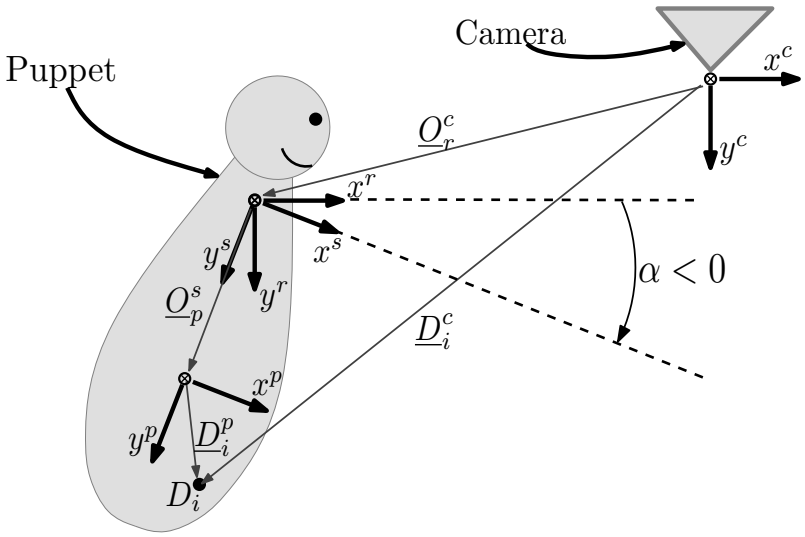


Figure 4.10: Transformation of scan coordinates D_i^p to camera coordinates D_i^c.

The vector $\underline{O}_r^c$ describes the translation from camera frame to rod frame and is given by $[x_r^c\ y_r^c\ z_r^c]^T$. Here x_r^c and y_r^c are constant distances from the camera to the rod, and z_r^c is

[1] the author would like to thank G.J.Strijkers and the University of Maastricht in this matter

the variable translation of the rod that can be controlled by the computer. The rotation matrix $\mathbf{R}_s^r$ holds the other controllable variable α,

$$\mathbf{R}_s^r = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

After the pixel transformation the pixels can be placed into the image plane $[b_x^I, b_y^I]^T$ as depicted in Fig. 4.11

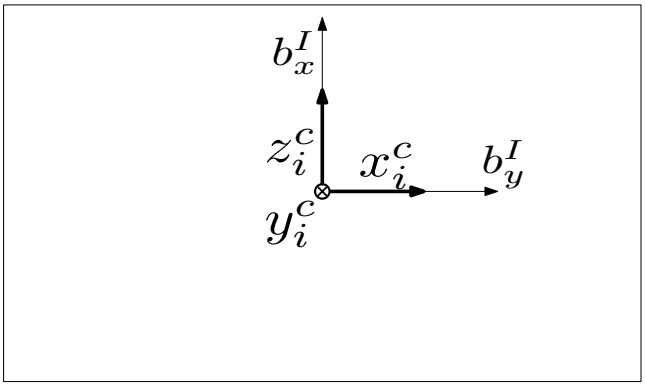


Figure 4.11: 3D camera pixel coordinates to pixels in the image plane.

according to the formulas

$$b_x^I = \frac{f}{P_s}\frac{z_i^c}{y_i^c}, \qquad b_y^I = \frac{f}{P_s}\frac{x_i^c}{y_i^c}, \tag{4.11}$$

where f is the focal length parameter of 6 [mm] and P_s is the camera pixel size and is 9.9 [μm].

Because the resolution of the scan was lower than the resolution of the camera and only the contour of the puppet was evaluated, cavities appear in the 2D projected image. This is illustrated in Fig. 4.12.

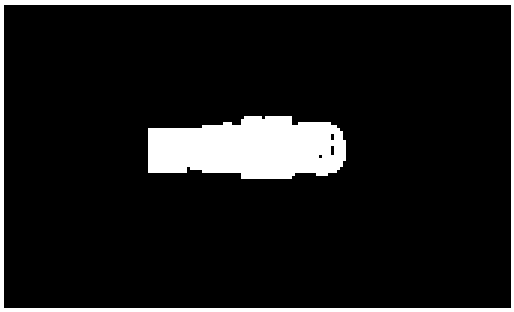


Figure 4.12: 2D projected contour of puppet with cavities.

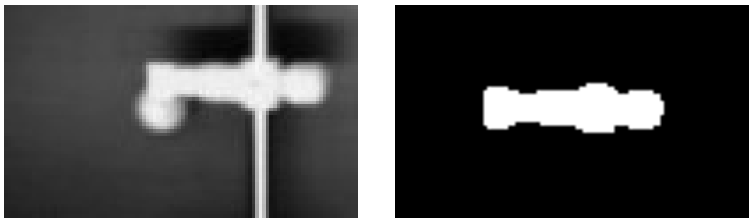
To solve for the cavities, the image $I(x^w, y^w)$ is convoluted according to

$$G(x^w, y^w) = \sum_{s=-2}^{2} \sum_{t=-2}^{2} W(s,t) I(x^w + s, y^w + t) \qquad s = -2,-1,0,1,2 \qquad t = -2,-1,0,1,2 \tag{4.12}$$

with a normalized smoothing kernel $W(s,t)$

$$W(s,t) = \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.13}$$

The result $G(x^w, y^w)$ of the smoothing is shown in Fig. 4.13, next to an originally captured image of the puppet.



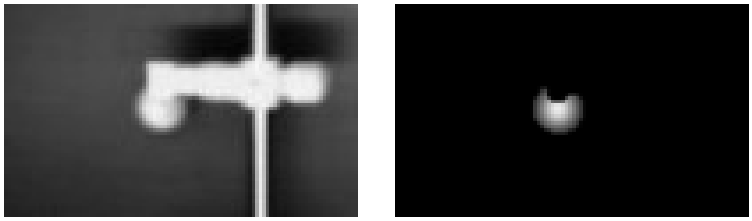
(a) Original image. (b) 2D projected mask.

Figure 4.13: Original image of puppet and 2D projected mask.

This way the mask of a puppet is thus fully constructed from the known geometry of the puppet and the measured position and orientation through the onboard encoders. A video of the projected puppets can be found in [16].

Finding the ball

What is left over in the captured image now can only be the ball itself. The ball will show its full shape, or when occluded by any of the above objects, a part of it. This is depicted in Fig. 4.14.



(a) Original image. (b) The left over.

Figure 4.14: Original image and left over after masking.

Because all the objects with high intensity values are now masked, the only high intensity values present in the ROI can be the ball. Therefore the highest intensity value present is assumed to belong to the ball. Another criterion that must hold, is that the surrounding pixels in a 3×3 window should have a larger intensity value than 100. The minimum and maximum values of the 8-bit monochrome image are 0 and 255 respectively.

This second criterion ensures that no noisy or damaged single pixel can be mistaken for the ball. Also the dark puppets are not mistaken for the ball, because every ball candidate must have an intensity value above the threshold of 100, else it is disregarded.

Now that the ball is extracted from the image and localized on the field the next function in the high level control scheme in discussed, see Fig. 4.2. This function describes how the position and velocity of the ball are estimated.

4.1.3 Estimate position and velocity

After the ball has been localized, the position and velocity have to be estimated. This is necessary to determine the interception point where the moving ball will cross the rod. To estimate the position and velocity of the ball, an *observer* can be used. In regard of the observer two assumptions are made.

- the system is linear and describes a freely rolling ball
- the position of the ball is measured and these measurements include noise

According to these assumptions, a *Kalman observer* [17] can be used. A Kalman observer is a linear filter that recursively estimates the states of a system from a series of incomplete

or noisy measurements.

The system of a freely rolling ball can be described as

$$\underline{x}_k = A\underline{x}_{k-1} + \underline{w}_{k-1}. \tag{4.14}$$
$$\underline{y}_k = C\underline{x}_k + \underline{v}_k. \tag{4.15}$$

The process noise $\underline{w}_k$ and measurement noise $\underline{v}_k$ are assumed to have a zero mean Gaussian distribution with covariances Q and R respectively. Estimating these covariances will be explained further in this section. A and C are the respective transition and measurement matrix.

$$A = \begin{bmatrix} 1 & \delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.16}$$

where δt is the reciprocal of the rate f_i which is 0.01 [s] and

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.17}$$

The previously estimated state $\underline{\hat{x}}_{k-1}$ can be used to predict the current state $\underline{\bar{x}}_k$ according to

$$\underline{\bar{x}}_k = A\underline{\hat{x}}_{k-1}. \tag{4.18}$$

Also a prediction can be made for the error covariance matrix $\bar{P}_k$, according to

$$\bar{P}_k = A\hat{P}_{k-1}A^T + Q. \tag{4.19}$$

Estimating the error covariance matrix $\bar{P}_k$ is necessary to improve the predicted state $\bar{x}_k$ with the measurement data. For this, the *Kalman gain* has to be calculated according to

$$K_k = \bar{P}_k C^T (C\bar{P}_k C^T + R)^{-1}. \tag{4.20}$$

With this gain and the provided measurement y_k the predicted state can be updated according to

$$\underline{\hat{x}}_k = \underline{\bar{x}}_k + K_k(\underline{y}_k - C\underline{\bar{x}}_k). \tag{4.21}$$

The predicted error covariance matrix can be updated according to

$$\hat{P}_k = (I - K_k C)\bar{P}_k. \tag{4.22}$$

where I is the unity matrix. The estimated error covariance matrix $\hat{P}_k$ can be used in the next step to calculate the Kalman gain K_k again.

Estimating the measurement noise covariance R can be done by looking at the error between a static ball position and the corresponding measurement. In case the ball is almost fully occluded by an object, the actual measurement of the bal will be completely on the outer edge of the ball. Therefore the standard deviation 3σ is determined to be the radius of the ball r_b. The variance σ^2 then becomes $\frac{r_b^2}{9}$. Therefore the covariance matrix R is

$$R = \begin{bmatrix} \frac{{r_b}^2}{9} & 0 \\ 0 & \frac{{r_b}^2}{9} \end{bmatrix} \tag{4.23}$$

where r_b is the radius of the ball and is approximately 10 [pix].

The estimate for the process noise Q can be done by evaluating the error that the point of interception has, which depends on the Kalman estimated states. Therefore in the next section the calculation for the point of interception will be discussed, as well as the tuning of the process noise covariance matrix Q.

4.1.4 Determine interception point

To intercept the ball, the rods have to be controlled towards the point of interception where the freely rolling ball will cross the rod if it follows its estimated trajectory, see Fig. 4.15.

This interception point $[i_k^{x^w} \quad i_k^{y^w}]^T$ can be determined according to

$$i_k^{y^w} = \hat{y}_k^{y^w} + (i_k^{x^w} - \hat{x}_k^{x^w}) \frac{\dot{\hat{y}}_k^{y^w}}{\dot{\hat{x}}_k^{x^w}} \tag{4.24}$$

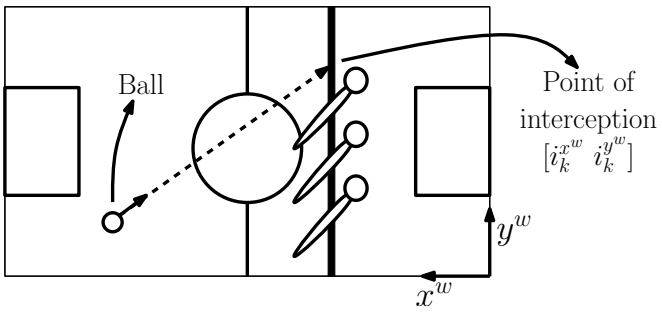


Figure 4.15: Point of interception.

where $\hat{x}_k^{x^w}$, $\hat{\dot{x}}_k^{x^w}$, $\hat{y}_k^{y^w}$, and $\hat{\dot{y}}_k^{y^w}$ are the Kalman estimated position and velocity of the ball and $i_k^{x^w}$ is the static rod position in the x^w direction.

According to (4.24) it is possible that the point of interception is determined to be outside of the table. In this situation the rod must not be moved. Therefore another criterion must be added.

$$i_k^{y^w} = \begin{cases} (4.24), & \text{if } i_k^{y^w} > 0 \text{ and } i_k^{y^w} < \text{width of table}, \\ i_{k-1}^{y^w}, & \text{else.} \end{cases} \tag{4.25}$$

Tuning of the process noise covariance Q
In (4.16) it was assumed that the ball moves with a constant velocity. However, due to bounces this assumption does not hold. Therefore it can be assumed that the process noise covariance will mainly depend on the changing acceleration of the ball. The covariance matrix Q is therefore determined to be

$$\tilde{Q}_k = \begin{bmatrix} \frac{1}{2}\tilde{a}\delta t^2 & 0 & 0 & 0 \\ 0 & \tilde{a}\delta t & 0 & 0 \\ 0 & 0 & \frac{1}{2}\tilde{a}\delta t^2 & 0 \\ 0 & 0 & 0 & \tilde{a}\delta t \end{bmatrix}, \tag{4.26}$$

where $\tilde{a}$ is taken as a tunable parameter. To tune this parameter a criterion must be used that decides when $\tilde{a}$ is properly tuned. The criterion to tune $\tilde{a}$ is that when the human controlled rod is at one side of the table and returns the ball at full speed in the y^w direction towards the opposite side, the electro-mechanically controllable rod with the three puppets should still be able to intercept the ball. This is schematically depicted in Fig. 4.16.

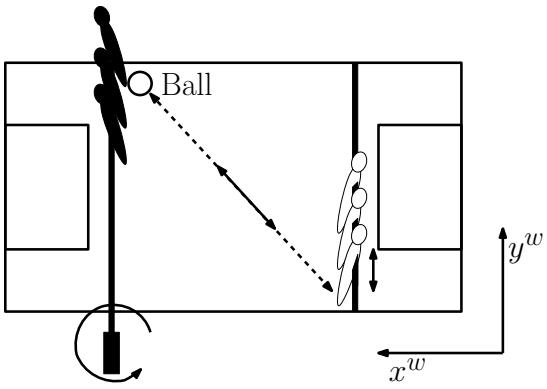
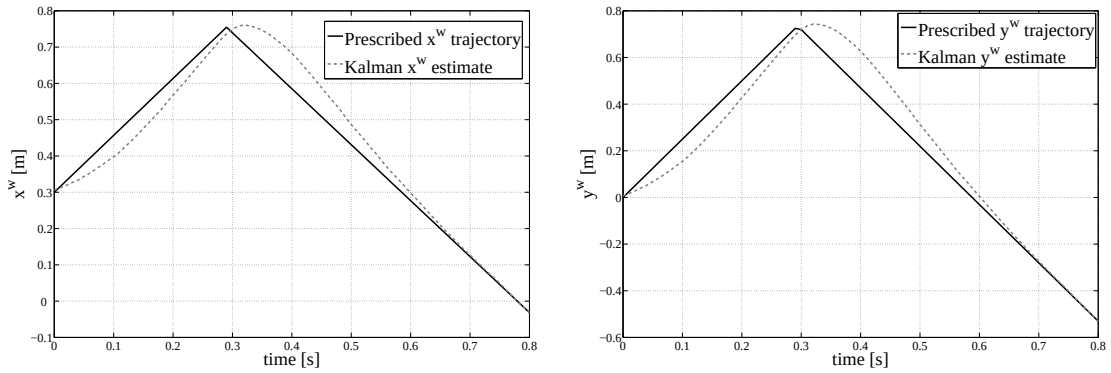


Figure 4.16: Convergence criterion.

What this actually means is that $\tilde{a}$ should be tuned such that the point of interception is accurate enough before the ball arrives back at the electro-mechanically controllable rod. The point of interception is accurate enough if it has an error of less than 0.015 [m], which is half the width of a puppet's feet.

In Fig. 4.17 the prescribed trajectory and the Kalman filtered position estimate are depicted, where $\tilde{a}$ was set at 20.000 [pix/s^2].



(a) Position x^w (b) Position y^w

Figure 4.17: Prescribed trajectories and Kalman filtered position estimates.

Fig. 4.17 shows that at time $t = 0.29$ [s] a bounce occurs. At time $t = 0.58$ [s] the ball has arrived back at the electro-mechanically controllable rod again. This means that the error between the estimated and the actual interception points should be less than 0.015 [m] at time $t = 0.58$ [s]. This is depicted in Fig. 4.18.

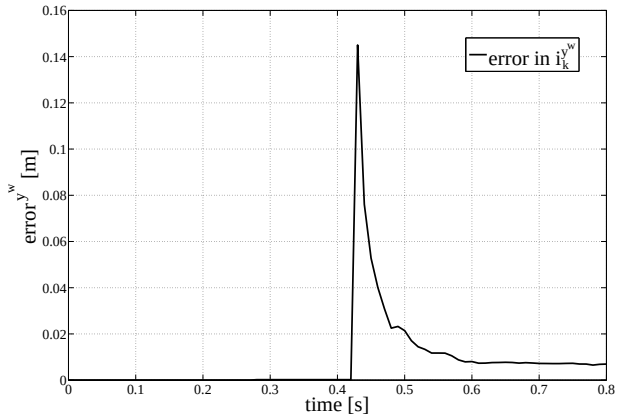


Figure 4.18: Interception error in time.

At time $t = 0.43$ $[s]$ the interception point is calculated to be inside of the table. At time $t = 0.58$ $[s]$ the error is 8 [mm] which is less than half the width of a puppets feet. Therefore a value for the tuning parameter $\tilde{a}$ of 20.000 [pix/s^2] is a proper value.

An animated illustration where the points of interception are determined can be found in [18].

4.1.5 Select puppet

When the interception point is calculated, one of the three puppets can be selected to intercept the ball. How far each of the puppets can reach is depicted in Fig. 4.19.

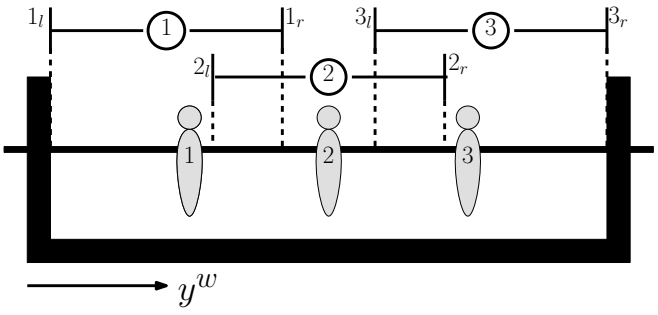


Figure 4.19: Range of each puppet.

Selecting a current puppet T_k depends on the position of the interception point $i_k^{y^w}$ and the previously selected puppet T_{k-1}. Making the selection of the current puppet T_k depend

on the previous puppet T_{k-1} creates hysteresis which prevents constant switching between the puppets.

$$T_k = \begin{cases} 1, & \text{if } i_k^{y^w} < 2_l; \\ 1, & \text{if } i_k^{y^w} < 1_r \text{ and } \mathrm{T}_{k-1}{=}1; \\ 3, & \text{if } i_k^{y^w} > 2_r; \\ 3, & \text{if } i_k^{y^w} > 3_l \text{ and } \mathrm{T}_{k-1}{=}3; \\ 2, & \text{else} \end{cases} \tag{4.27}$$

4.1.6 Create rod reference point

Now that a puppet is selected the reference point for the rod $r_k^{y^w}$ can be calculated. It can be assumed that the rod is actuated in puppet T_1, as depicted in Fig. 4.19. Therefore the equation for the reference point of the rod is

$$r_k^{y^w} = i_k^{y^w} + P_b(1 - T_k) \tag{4.28}$$

where P_b is the equidistant pitch between the puppets.

4.1.7 Kick ball

The final step in the high level scheme is to design a criterion that decides when a kick should be performed. A kick is performed when the estimated position of the ball is between the dashed lines, as depicted in Fig. 4.20.

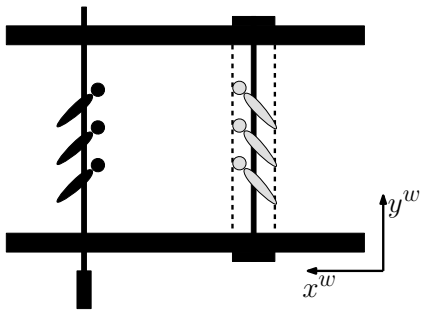


Figure 4.20: Kick ball between dashed lines.

4.1.8 Inter Process Communication

The reference point and the kick command are send to the low level scheme through IPC (Inter Process Communication). The position and orientation of the rod, for the masking of the yellow puppets, are received through IPC. The implementation in this work uses *shared memory* [14] to send and receive the data.

This concludes the functions that are used in the *high level scheme*. In the next section the functions that are used in the *low level scheme* are discussed.

4.2 Low level scheme

In this section the functions that are used in the *low level scheme* are described. The low level scheme is depicted in Fig. 4.21.

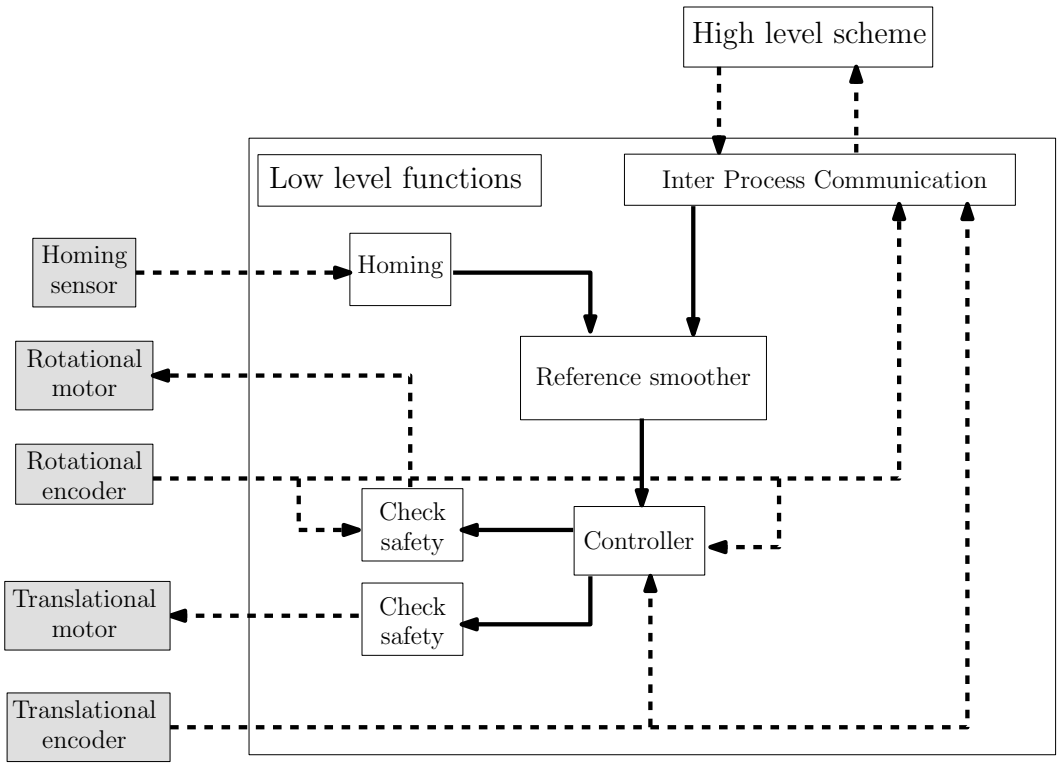


Figure 4.21: Low level control scheme.

4.2.1 Inter Process Communication

In the low level scheme the reference for the rod and the kick command are received through IPC, see Section 4.1.8. Through IPC also the position and orientation of the rod are send to the high level scheme. This data is needed to mask the yellow puppets.

4.2.2 Reference smoother

In the previous section the reference trajectory for the rods and the kick command were determined. The reference trajectory is calculated at a frame rate f_i of 100 [Hz], which is determined in Section 4.1.2. However, to prevent sample delay from causing too much phase lag [13], the low level control scheme should run at a higher rate, which is 2000 [Hz]. Because the reference trajectory is determined at a lower rate than the low level control scheme executes, the reference trajectory will exhibit discontinuous time behavior in the low level control scheme. To have a smooth continuous reference trajectory for the rod, a second order smoothing algorithm is implemented.

The problem addressed is to calculate a smooth trajectory $w(t)$ from a given initial state $w(0), \dot{w}(0)$ and a desired end state $w(t_f), \dot{w}(t_f)$ within the shortest possible time t_f. This trajectory is constrained by a maximum velocity v_{max} and a maximum acceleration a_{max}. The acceleration of the trajectory depends of the control effort $q_w(t)$ according to

$$\ddot{w}(t) = q_w(t) \tag{4.29}$$

with initial conditions $w(0) = 0$, $w(t_f) = w_f$, $\dot{w}(0) = \dot{w}_0$, $\dot{w}(t_f) = 0$ and boundary conditions $|\dot{w}(t)| \leq v_{max}$ and $|q_w(t)| \leq a_{max}$. The minimum time solution to this system can be found at the boundaries of the velocity and acceleration constraints. This means that the system is either constantly accelerating, constantly decelerating or at constant velocity v_{max}. A detailed explanation of this algorithm can be found in [19]. In Fig. 4.22 this smoothing is depicted.

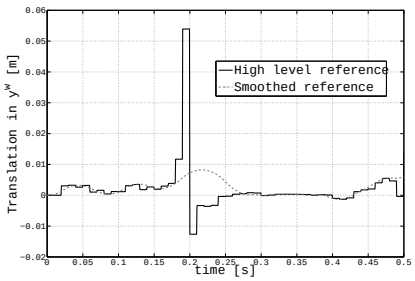


Figure 4.22: Input trajectory and smoothed trajectory.

What can be derived from this picture, is that the high level reference is a discontinuous signal changing at a rate of 100 [Hz]. Smoothing this signal creates a continuous signal that is more suitable to be used as the reference signal for the rods.

4.2.3 Controller

As explained in [10], the rod is actuated by two motors. One motor provides the rotational movement and one motor provides the translational movement of the rod, see Fig. 4.23.

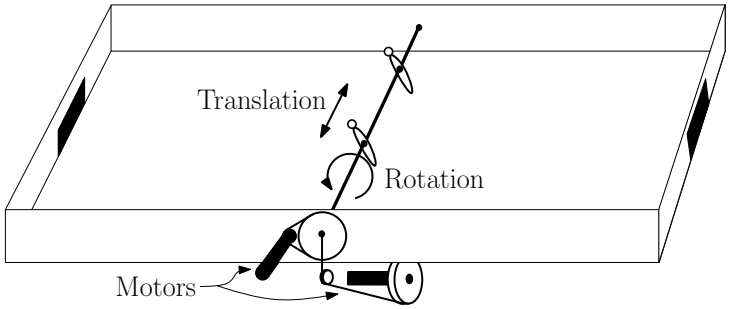


Figure 4.23: Two controllable degrees of freedom.

Since there is no coupling between the two degrees of freedom, independent SISO (single-input single-output) controllers can be designed. To design a suitable controller first the frequency response functions are measured of the system. The frequency response functions of both the rotational as the translational DOF are depicted in Fig. 4.24.

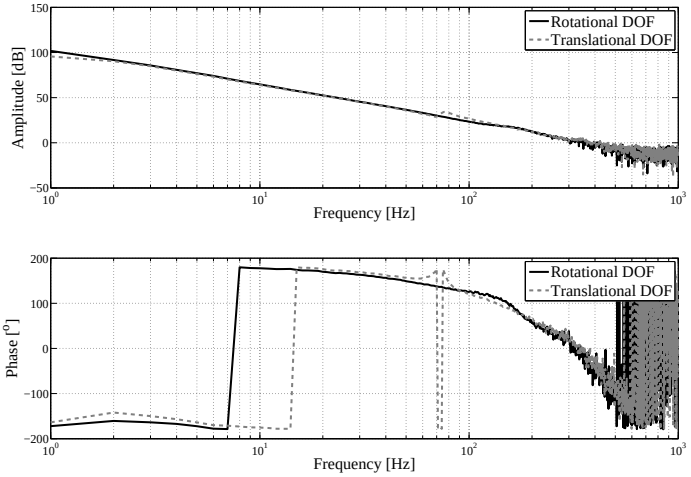


Figure 4.24: Rotational and translational system FRF.

Because the system was designed to have little friction and very stiff connections [10], it is expected that the system will respond as a pure mass with a pure delay and high frequent noise. A lead filter should then be sufficient [20] to create a stable closed loop system. To suppress the high frequent noise a low-pass filter is added.

The values used in the lead- and the first order low-pass filter for each DOF of one rod are given below, as well as the obtained bandwidth, the modulus margin, the phase margin and the gain margin.

	Rotational DOF	Translational DOF
gain [-]	5.1e-4	9.9e-4
lead zero [Hz]	5.3	7.3
lead pole [Hz]	48	66
low-pass pole [Hz]	100	130
bandwidth [Hz]	16.0	22.0
modulus margin [dB]	6.0	6.0
phase margin [deg]	36.9	36.5
gain margin [dB]	10.1	10.1

For both controllers the FRF's (frequency response functions) are depicted in Fig. 4.25.

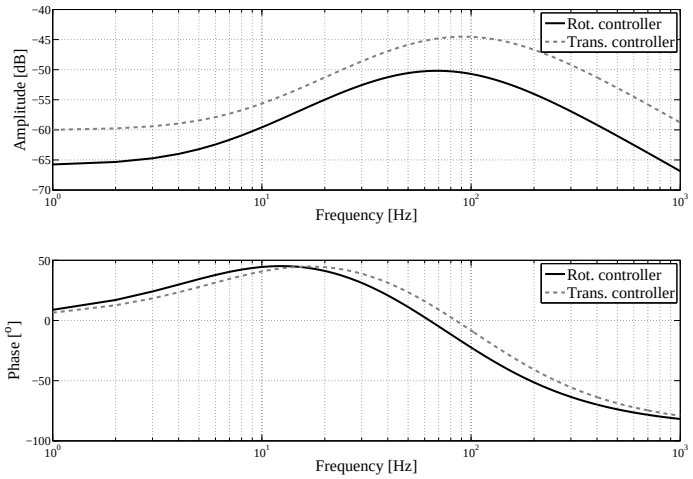


Figure 4.25: Rotational and translational controller FRF.

With these values the discrete time controllers become

$$C_r = \frac{0.001162z - 0.001142}{z^2 - 1.59z + 0.6282}, \qquad C_t = \frac{0.002726z - 0.002663}{z^2 - 1.477z + 0.5402}. \tag{4.30}$$

The open loop responses of the rotational and translational are depicted in Fig. 4.26.

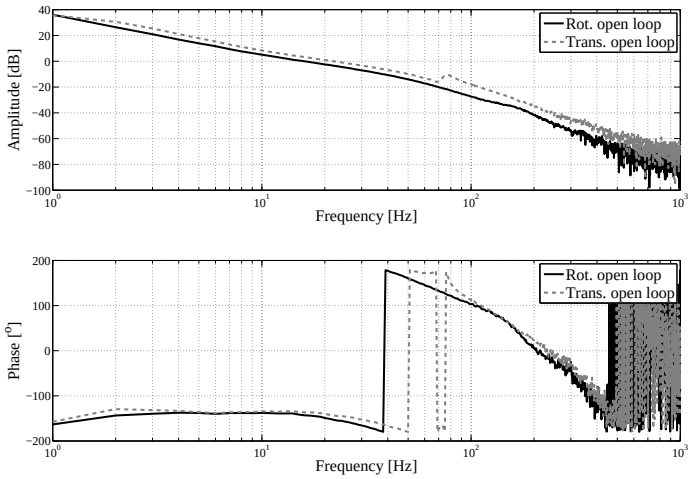


Figure 4.26: Rotational and translational open loop FRF.

4.2.4 Check safety

The signals that come from the controller have to be checked before they are send to the motors. There are two properties of these signals that are checked.

- the values are bounded, which means that no unintentional high controller values are allowed
- the **rotational** signal is bounded by an absolute value of 2π

When one of these properties is violated, all the controller outputs are permanently set to zero.

The functions used in the low level scheme are now described. In the next chapter the results of the project are discussed.

Chapter 5

Results

This chapter describes the results of this project obtained during the testing phase with the automated football table. The setup consists out of one electro-mechanically controlled rod versus one human controlled rod. Both rods have three puppets attached, which are capable of intercepting and returning the ball, see Fig. 5.1.

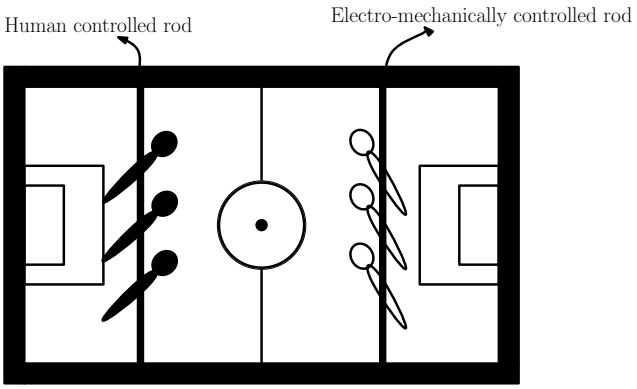


Figure 5.1: Current setup.

In this chapter there are 3 topics that will be discussed.

- the ball tracking performance,
- the accuracy of the interceptions,
- the results of an average practice game.

5.1 Ball tracking performance

Tracking of the ball depends on how much of the time the ball can be found in the ROI. Although the ROI was theoretically defined such that the image processing algorithm can not loose the ball, it is still possible that the ball is lost. Occurrences when the ball is lost can depend on 3 events.

- the automated puppets move too fast and by causing *motion blur* they are mistakenly seen for the ball,
- the velocities of the ball are too high and the ROI can not keep up,
- the human rod or the automated rod occlude the ball and together with the ball move out of the ROI.

All 3 of these events have to be examined.

5.1.1 Fast movement of automated puppets

An issue that makes it possible to see the puppets for the ball, is so called *motion blur*. Motion blur causes the projection of a fast moving puppet to 'stretch out' in the captured image. This is depicted in Fig. 5.2.

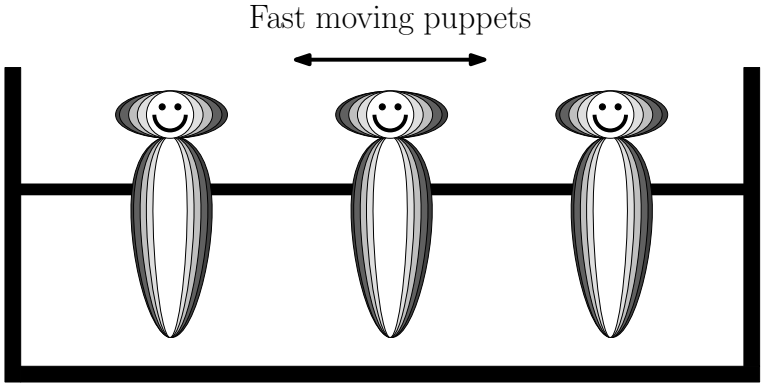


Figure 5.2: Stretched out puppet due to motion blur.

The mask that was developed for the puppets, was implemented somewhat larger due to the smoothing convolution that was described in Section 4.1.2. Therefore it is possible that the mask is sufficiently large to prevent motion blur from mistaking one of the puppets for the ball.

A test has shown that when the rods were following a random trajectory with maximum velocities and the ball was not inserted in the field, the ball was never found. Therefore it can be concluded that motion blur does not cause the image processing to mistakenly see a puppet for the ball.

5.1.2 Ball velocities

If the velocities for the ball are too high for the ROI to keep up, the ROI can loose the ball. In Fig. 5.3 a plot is made of the velocity of the ball against the occurrences that the ball was lost. The thick lines indicate that the ball was lost.

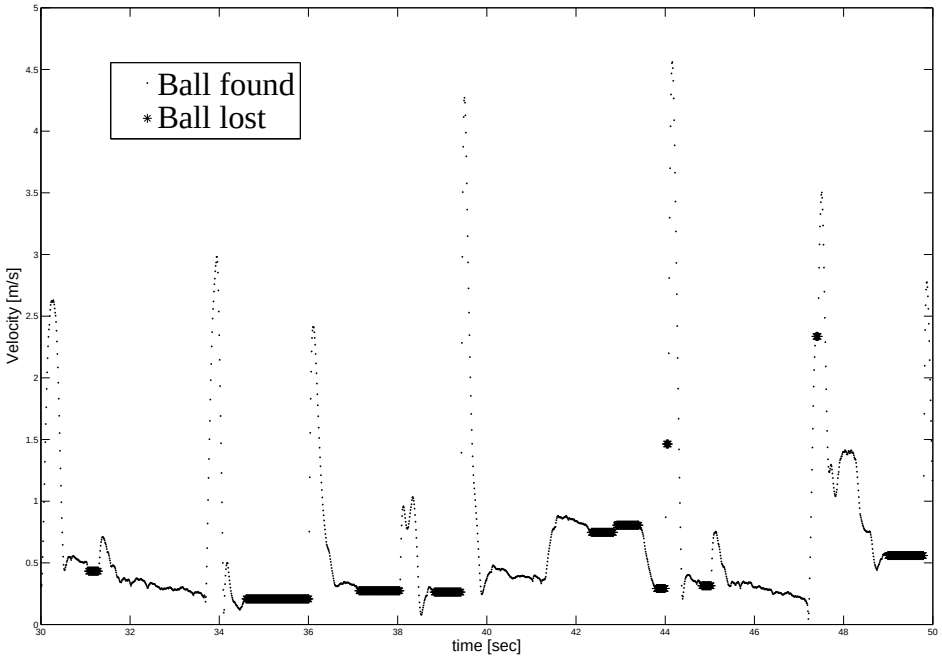


Figure 5.3: Ball found versus ball speed.

What can be seen from this picture is that the ball is never lost when the velocity is at its maximum, 4.5 [m/s].

5.1.3 Occluding rod movements and ball velocities

From Fig. 5.3 can be seen that the ball is sometimes lost. However this does not seem to be caused by the fact that the velocity of the ball, which is 4.5 [m/s] maximum, is too

high or because the puppets are mistakenly seen for the ball. The ball seems to be lost sometimes when it experiences low velocities. Low velocities typically occur when the ball is at a rod, and controlled by a puppet. These movements can occlude the ball such that the ball rolls, while shaded by one of the puppets, outside of the ROI. Therefore it can be concluded that when the ball is lost, this is caused by an occluding movement of the rod. This goes for the human controlled rod as well as for the electro-mechanically controllable rod.

5.2 Accuracy of interceptions

The most important aspect of the performance of the automated football table depends on how accurate the system is able to intercept and return the ball. Determining where the ball should be intercepted and timing of the forward kicking movement are crucial factors in this. In theory the ball should not be able to pass the puppets. However when playing an average game of table football it becomes clear that there are many occurrences that it does happen that the intercept fails. This is depicted in Fig. 5.4. Whenever the ball position becomes smaller than the indicating line at 0.4 [m] in the x^w direction, the ball went past the rod and the intercept failed.

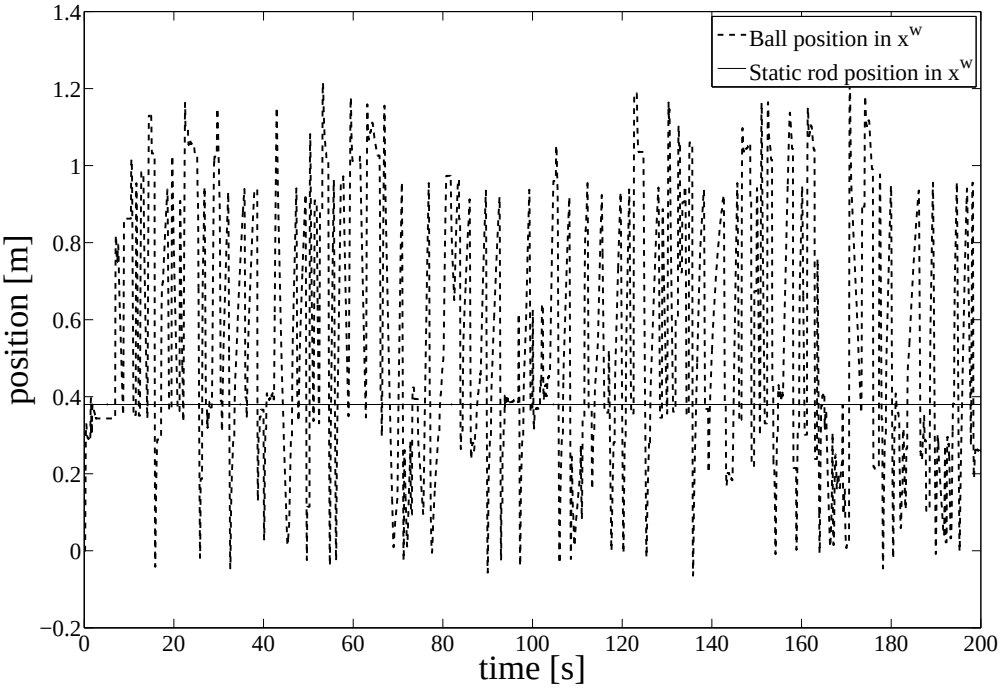


Figure 5.4: Ball x^w direction over time.

Determining the timing of the forward kicking movement and performing a successful intercept depends on several factors.

- the accuracy of the point of interception determined in the high level scheme.
- controlling the rod towards the point of interception in the low level scheme.
- the total amount of time delay between capturing an image and the moment the motors are actuated.

All these steps have to be evaluated in order to pinpoint why an intercept was unsuccessful.

5.2.1 The accuracy of determining the point of interception

When a bounce occurred while the ball was moving a maximum velocity the error of the point of interception can be determined, see Section 4.1.4. This error is again depicted in Fig. 5.5.

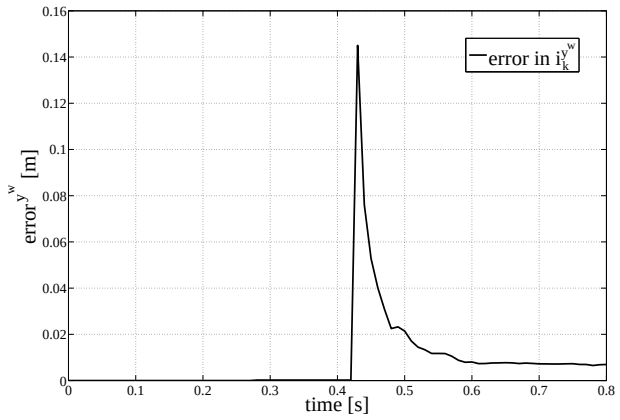
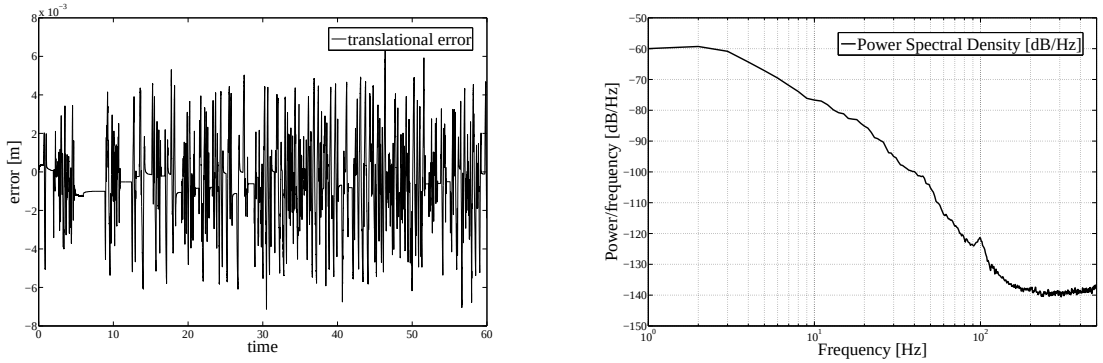


Figure 5.5: Interception point error after full speed bounce.

In Section 4.1.4 it was stated that the bounce occurred at time $t = 0.3$ $[s]$ and that the ball arrived back at the electro-mechanically controllable rod at time $t = 0.6$ $[s]$. In this scenario, that is created specifically for the current setup with only one automated rod, the error was 8 [mm].

5.2.2 Controlling the rod

To intercept a fast moving ball, the puppets on the rod have to be directed to the point of interception with the ball. Therefore the point of interception is calculated in the high level control scheme discussed in Section 4.1, and this reference is used to control the rods. The error of the translation DOF in the low level control loop discussed in Section 4.2 can be investigated, to see if this error has an influence in the missed interceptions. The error of an average game of table football is logged, and presented in Fig. 5.6.



(a) Error in time.

(b) Power Spectral Density

Figure 5.6: Translational error.

In this figure it can be seen that the maximum error is 7 [mm]. In the Power Spectral Density plot it can also be seen that there is a large peak at 100 [Hz]. This peak can be explained by the fact that the *high level scheme* updates the translation for the rod at 100 [Hz].

5.2.3 Time delay

Another issue that should be addressed is time delay. This delay contains the total amount of time it takes to capture, process and actuate the rod after a ball was positioned somewhere on the field. An estimate of the worst case delay can be done by evaluating all the individual processing steps that were performed.

- Exposure time: because the camera captures images at a rate of 200 [Hz] and the exposure time is set at its maximum this delay is always half the exposure time, which is 2.5 [ms]
- Buffering image: according to several tests, this value is estimated to be 15 [ms].
- High level reading: the maximum time delay that can occur while reading from the camera buffer, 5 [ms].

- High level processing: the maximum amount of time it takes to process the image, which is 10 [ms].
- Low level reading: reading out the value from the high level scheme can lag one sample behind, so this maximum amount can also be 10 [ms].
- Low level processing: this is done at 2000 [Hz]. The maximum delay can therefore be 0.5 [ms].
- Low level output: Zero order hold effect. Time delay caused by discretizing the output. This is 0.25 [ms].

The maximum amount of time delay therefore is 43.25 [ms]. The maximum velocity of the ball in the y^w direction in which the rod translates, is assumed to be 2.5 [m/s]. Therefore the maximum translational error that can be caused by time delay is 108 [mm].

5.2.4 Conclusion on intercept accuracy

With respect to the accuracy of the intercept there were 3 different aspects examined. The errors that each of these 3 aspects created, are listed in the table below.

	Maximum possible error [mm]
interception point accuracy	8
control of the rod	7
time delay	108
Total error	**123**

The total maximum error is therefore 123 [mm], where half the width of a puppets feet was 15 [mm]. What can be stated is the fact that the error mainly depends on time delay. If there was no time delay between an actual ball position and the moment of actuating the motors, the maximum error would be 15 [mm] which is exactly the allowable margin. However, at this point it is not possible to predict the exact amount of time delay and therefore it is not possible to correct for it.

5.3 Average practice game

Obviously the ultimate way to test the table is just by playing a game of table football. Because each side has only one rod with three puppets attached, there are no real tactics possible for the human opponent to perform. Still, an amusing game can be played. When an average game is played with speeds up to 4 [m/s] the automated puppets are definitely

a fearsome opponent. The strength of the automated rods lies in the fact that they immediately return the ball, with a very high velocity. It is very difficult for a human opponent to respond to this reactive action. Another benefit the automated puppets have is that because the image processing algorithm only considers the position of the ball and its velocity, the automated rods cannot be fooled by diversions which are usually the strength of a good human player.

When I played a game against the automated puppets, I actually got beaten pretty hard. Reacting on the high velocity kick the puppets deliver is absolutely impossible. However, when a more professional player accepts the challenge and plays with higher velocities the automated puppets will probably lose. More research is therefore needed to optimize the automated table and push its capabilities to the limit.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

The goal in this report was to develop the software that enables the autonomous football table to defeat a human opponent. The exact goal that described this project was

Design and implement a vision based control strategy that detects the ball and controls a single rod in real-time to be able to defeat a human opponent in a game of table football

This goal was partially achieved. An average skilled player that plays with speeds up to 5 [m/s] could be defeated. At this point however, there is no test data available that can indicate the performance of the table beyond this velocity. To achieve the described goal several steps were taken.

1. the hardware was configured and installed,
2. the software design was developed using the UML notation,
3. two communicating software schemes were created: a *high level scheme* and a *low level scheme*,
4. the *high level scheme* processes the images and creates a reference for the rod,
5. the *low level scheme* executes this reference in a stable and safe manner.

Fundamental to the overal result of the table was the ability to keep the ROI located on the position of the ball. When this failed, several possible causes where examined. High velocities and motion blur were not responsible for losing the ball. When the velocities of the ball were examined it became clear that losing the ball mostly occurred at low velocities, when the ball was located at a rod. Therefore the main cause for losing the ball

was the fact that the ball rolls outside the ROI while it is constantly being occluded by one of the puppets.

Another important aspect of the overal result was determining the exact point of interception. Wrong intercepts did occur, and there were several possible causes examined. The implemented Kalman observer was tuned and converged fast enough after a bounce. This however only goes for the current setup with one automated rod. As soon as the setup is expanded to more automated rods, the Kalman observer has to be tuned again, or replaced completely by another form of observer. One of the possibilities for this is to implement a non-linear Kalman observer [21], or to apply some form of particle filtering [22].

By far the most important cause for a wrong intercept was time delay. This delay occurs between capturing an image of the ball, processing the image and actuating the rod towards a certain point. For a maximum possible error of 115 [mm], time delay could be held responsible for 101 [mm].

Because at this point no concrete implementation is made to estimate this delay, the possibility remains that the error becomes 115 [mm]. Creating an online estimator for the delay could improve the accuracy of the interceptions. If the delay is accurately estimated, implementing a proper prediction algorithm can solve this problem.

6.2 Recommendations

Concrete the recommendations are as follow:

- improve the ball found ratio, by implementing an algorithm that tracks the occluding puppet instead of the ball, in case the ball is occluded at a puppet,
- redesign the observer so that it can deal with the frequent bounces that occur
- investigate how time delay can be estimated
- implement a prediction algorithm that lowers the error in the estimated ball position caused by this time delay

Bibliography

[1] Y. Shirai and H. Inoue. Guiding a robot by visual feedback in assembling tasks. *Pattern Recognition*, 5:99–108, 1973.

[2] N. Shaikh and V. Prabhu. Vision system for model based control of cryogenic tunnel freezers. *Computers In Industry*, 56:777–786, December 2005.

[3] G. Ballantyne and F. Moll. The da vinci telerobotic surgical system: The virtual operative field and telepresence surgery. *Surgical Clinics of North America*, 83(6):1293–1304, December 2003. Robotic Surgery.

[4] J. Kehoe, J. Watkins, A. Causey, and R. Lind. State estimation using optical flow from parallax-weighted feature tracking. Conference on Guidance, Navigation and Control, Keystone, Colorado, August 2006. AIAA.

[5] D. Bruijnen, W. Aangenent, and J. van Helvoort. From vision to realtime motion control for the robocup domain. In *Control Applications*, pages 1297–1302, Singapore, 2007. IEEE.

[6] T. Chau, J.Then, M. Turnbull, S. Wan, and S. Cheng. *Robotic Foosball Table*. University of Adelaide, Adelaide, Australia, October 2007.

[7] M. Aeberhard, S. Connelly, E. Tarr, and N. Walker. *Foosball Robot*. Georgia Institute of Technology, Georgia, USA, August 2007. `http://www.eskibars.com/projects/foosball_robot/`.

[8] T. Weigel and B. Nebel. Kiro - an autonomous table soccer player. *Robocup 2002: Robot Soccer World Cup VI*, pages 384–392, 2002.

[9] Gauselmann GmbH. Starkick, 2005. `http://www.merkur-starkick.de/`.

[10] G. Heldens, J. de Best, and M. Steinbuch. Mechanical design of an automated foosball table. Internal Report DCT 2008.083, Eindhoven University of Technology, 2008.

[11] USTSA and Tornado Table Soccer, Fort Worth, Texas. *USTSA Foosball Rules of Play*. `http://www.foosball.com/learn/rules/ustsa/`.

[12] Object Management Group, Pasadena, California. *Unified Modeling Language*, January 1997. `http://www.uml.org`.

[13] G. Franklin, D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison Wesley, 3rd edition, 1995.

[14] Dave Marshall. *IPC:Shared Memory*. Cardiff School of Computer Science, May 1999. `http://www.cs.cf.ac.uk/Dave/C/node27.html`.

[15] J. Bouguet. *Camera Calibration Toolbox for Matlab.* California Institute of Technology, Pasadena, California, June 2008. `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[16] R. Janssen. Perspective projection of the puppets. Youtube. `http://www.youtube.com/watch?v=FhK2ua1hpo0`.

[17] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82:35–45, March 1960.

[18] R. Janssen. Interception points. Youtube. `http://www.youtube.com/watch?v=d7xgL5mEyeY`.

[19] O. Purwin and R. D'Andrea. Trajectory generation for four wheeled omnidirectional vehicles. *IEEE Proc. of American Control Conference*, 7(0743-1619):4979–4984, June 2005. Portland, Oregon.

[20] Y. Oded and M. Nagurka. Automatic loop shaping of low order qft controllers. In *Electrical and Electronics Engineers in Israel*, IEEE, pages 29–32, 2004.

[21] D. Simon. *Optimal State Estimation: Kalman, H infinity, and Nonlinear Approaches*, volume 1. John Wiley & Sons, July 2006.

[22] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2001.

Appendix A

EUTAFT startup guide v1.0

This appendix describes a stepwise approach how to startup the EUTAFT (**E**indhoven **U**niversity of **T**echnology **A**utomated **F**ootball **T**able) application. Practical issues about connecting the workstation to the hardware are described, as well as a description of how the software is implemented and functions. The steps described apply to the setup of the automated football table equipped with one rod.

A.1 Installing hardware

A.1.1 Necessary components

To startup the table the following components are needed

- Intel dual-core 2Ghz workstation equipped with
  - a Linux low-latency kernel
  - the Matlab/Simulink R2007b environment
  - 2 Gigabit ethernet ports that support jumbo frames
- Beckhoff data acquisition module consisting of
  - 1 EK1100 master module
  - 1 EL1008 digital input
  - 1 EL2008 digital output
  - 1 EL4132 analog output
  - 2 EL5101 incremental encoder interface
- Prosilica GC640c/GC640 gigabit ethernet camera

- Regulated power supply 24V/5A
- Heavy duty power supply 24V/20A
- 2 ELMO Violin power amplifiers 25V/60A
- TECBALL football table equipped with one controllable rod
- CD-rom with EUTAFT v1.0 software

A.1.2 Connecting the components

Connecting the Prosilica camera

The Prosilica camera has to be connected to the workstation through one of the ethernet ports. This port should support jumbo frames. On the workstation startup the NetWorkmanager via a terminal. In the NetworkManager select the appropriate port, and set the IPv4 settings to 'manual'. In the address window fill in the following IP-settings:

Address	Netmask	Gateway
169.254.1.1	255.255.0.0	(leave empty)

The camera is now physically connected to the workstation. The framerate and exposure time settings are described in [1].

Connecting the Beckhoff data acquisition module

The Beckhoff EK1100 master module should be connected to the workstation through the other ethernet port. This port does not necessarily need to support jumbo frames. The (24V) and the (0V) inputs should be connected to the respective (+) and (-) of the regulated power supply. The (-) output port of the EK1100 should also be connected to the (-) of the heavy duty power supply, to create a common ground. The EK1100 master module is now physically connected to the workstation, and the remaining slave modules can be attached one after another to the master.
The EL1008 is a digital input which is needed for rotational homing. The black wire from the rotational homing sensor should be inserted to the (1) input to the EL1008. The blue wire from the homing sensor should be connected to the (+) of the heavy duty power supply and the brown wire should be connected to the (-) of the heavy duty power supply.
The EL2008 is a digital output which is needed to enable the ELMO amplifiers. Place a 10 [kOhm] resistor in the (1) output of this slave, and connect the other end of the resistor to the (EN+) port of both the ELMO amplifiers.
The EL4132 analog output is the control output and must also be connected to the ELMO amplifiers. The (1) and (3) outputs should be connected to the respective (CREF+) and

(CREF-) inputs of the first ELMO amplifier and the (5) and (7) outputs should be connected to the respective (CREF+) and (CREF-) inputs of the second ELMO amplifier. The EL5101 encoder interfaces should be connected to the first and second encoders that are attached to the motors. For this connection can be referred to [2] and [3].

Connecting the ELMO amplifiers

The (VP+) and (PR) inputs of both the amplifiers should be connected to the respective (+) and (-) of the heavy duty power supply. For each amplifier the (M2) and (M1) outputs should be connected to the respective (+) and (-) of each motor. The (CMRET) input should be cross-linked to the (CREF-) input port and the (PR) input should be cross-linked to the (EN-) input. Make sure that some of the remaining ports are connected as described in the Beckhoff EL2008 and EL4132 connection instructions above.

A.2 Installing software

Installing a real-time Linux low latency kernel
To properly install a real-time Linux kernel together with the Matlab/Simulink environment, follow the instructions in [4].
Installing the Prosilica API
To be able to use the Prosilica camera, the Prosilica API (Application Programming Interface) has to be downloaded from [5]. This API contains all the functions and libraries that are needed to operate the camera. Follow the instructions in the included manual to install this API.
Installing the SOEM ethercat library
To be able to use the Beckhoff ethercat modules, the SOEM ethercat library has to be installed. This library makes it possible to use the ethercat slaves. The SOEM ethercat library also holds the specific blockset that can be used in the Matlab/Simulink environment. To install the library follow the instructions in [6]

A.3 Creating the executables

The executables that control the automated football table are developed in the Matlab/Simulink environment. This environment allows the user to work in a modular framework for code development using the real-time workshop, and it automatically generates the code from which the executables are compiled. There are two executables created. One executable (100 [Hz]) holds the high level image processing algorithm, and the other executable (2000 [Hz]) holds the low level control of the motors. The developments of the high level and the low level executables are explained separately.

A.3.1 High level executable

This executable is named the 'VISION' executable, because this executable mainly holds image processing algorithms. To ensure that this executable runs at a rate of 100 [Hz] first a so-called 'target' has to be specified. This target ensures a consistent execution rate, regardless of other processes that run on the workstation. The target for the VISION executable is located on the CD-rom and can be installed according to the included README.txt file. In the Simulink VISION.mdl scheme this target can be selected in the configuration parameters. The VISION.mdl should be configured to run at 100 [Hz] with a discrete fixed-step solver. The scheme contains the following modules.

Acquisition
In this module the images coming from the camera are acquired. The inputs to this module are the position of the region of interest (ROI) and the masks that were developed to mask certain areas and objects in the field. The outputs are the size of the ROI, an optional viewer output, a measured ball position and a ball found indicator.

Scanning
Here the decision is made when to start scanning for the ball if it is lost.

Kalman filter
In this module the position measurements of the ball are converted to an estimated position and velocity. The input to the enable port is the ball found indicator. If the ball is not found, the filter holds its current output.

Rod interception
Here the point of interception of the ball with the rod and the kick flag are determined. In this module it is determined which of the three puppets should intercept the ball. The output is the according translational reference of the rod.

Sf shm vision
The kick flag and the translation of the rod are send through shared memory to the low level control scheme. The output of this module is the measured position and oriëntation of the rod.

Puppet mask
The mask for the electro-mechanically controlled puppets is determined in this module. Inputs are the measured measured position and oriëntation of the rod, and the data file that holds the 3D scan of a puppet.

A.3.2 Low level executable

This executable is named the 'MOTION'executable, because in this executable there are only motion control algorithms implemented. The target for this scheme is included on the CD-rom and can be installed according to the included README.txt file. MOTION.mdl

should be configured to run at a rate of 2000 [Hz] with an ode1 (Euler) fixed step solver.The scheme contains the following modules.

Enabling amplifiers
With this module the ELMO amplifiers can be turned on or off. This selection can be made to run the executable without having the ELMO amplifiers enabled.

Rotation homing
The inputs to this block are the kick flag coming from the high level control scheme and the measured rod oriëntation. This module ensures that the rod is properly homed in the rotational DOF before it starts following the rotational reference.

Translation homing
In this module the translational DOF is homed before it starts following the translational reference coming from shared memory.

Rotational and translational smoothers
Here the translational and rotational references are smoothed so that the reference input to the control loop does not exhibit discontinuous time behavior.

Controllers
These modules hold the controllers that stabilize the rotational and translational DOF. The implemented values in these controllers are presented in [1].

Rotation safety
In this module criterions are implemented that can put the control input immediately to 0. The first criterion is that the control input should be bounded by a maximum and a minimum value of +24 and -24 respectively. Another criterion is that the rotation is bounded by -2π and +2π. If the controller input exceeds these values, the output is permanently put to 0.

Translation safety
The safety criterion used in this module is that the controller input should remain between a minimum of -24 and a maximum of +24.

Terminate to zeros
When the executable finishes, this module ensures that the controller output is terminated to 0.

Sf shm motion
This module sends the measured orientation and translation to the VISION scheme to be processed. Its outputs are the kick flag and the translation reference determined in VISION.

EL4132 and EL5101 modules
The EL4132 and EL5101 modules come from the SOEM library. These modules interface with the data acquisition equipment and are the respective analog output and incremental encoder input modules for both the rotational as the translational DOF.

A.3.3 General build script

To create both of the VISION and MOTION executables, a general build script is made (make.m). If this script is run in Matlab, both executables are created and placed in the current folder. The script also sets some of the parameters in the Matlab/Simulink schemes, in order for them to be successfully compiled. Before running this script, first clean the workstation cache from previously installed executables and libraries by running the cleaning script (make_clean.m).

A.4 Start game

Before starting a game, make sure that all the cables are properly connected and that the surrounding of any moving part is cleared from obstacles. A game can be started by opening two terminals in Linux. Make sure that the user is running these terminals as a 'super-user'. Then both the VISION and MOTION executables can be run by the respective commands ./VISION and ./MOTION.

Bibliography

[1] R. Janssen, J. de Best, and R. van de Molengraft. Eindhoven university of technology automated football table. Master's thesis, Eindhoven University of Technology, June 2009. DCT.nr.2009-045.

[2] Beckhoff. *EL5101 Incremental Encoder Interface*. Haarlem, the Netherlands. `http://www.beckhoff.com/english/ethercat/el5101.htm`.

[3] Avago. *HEDL-5540 A11 Incremental Encoder*. Regensburg, Germany. `http://www.avagotech.com/pages/en/motion_control_solutions/incremental_encoders/housed_mid_size/hedl-5540a11/`.

[4] J. de Best and R. Merry. Real-time linux for dummies. internal report, Eindhoven University of Technology, 2008. DCT.nr.2008-103.

[5] Prosilica. Linux sdk for gige vision, april 2009. `http://www.prosilica.com/products/linuxsdk.html`.

[6] SourceForge. *Simple Open EtherCAT Master*. `http://developer.berlios.de/projects/soem/`.