The Design of a Semi-Automated Football Table

Rob Janssen, Jeroen de Best, René van de Molengraft and Maarten Steinbuch

***Abstract*— This paper presents the design of a semi-automated football table. One side of the table is equipped with automated rods, whereas the other side has regular rods. With this setup one or two human players can play against a computer controlled adversary. The paper describes the design and integration of a suitable hardware setup with vision based control algorithms that control the automated rods to intercept and return the ball.**

I. Introduction

The goal of this project is to design an automated football table that will be capable of defeating one or two human players in a game of table football. To achieve this goal several aspects need to be investigated such as the design of a suitable hardware setup, the development of the vision algorithms to track the ball and the low level control of the automated rods. These rods are controlled such that they can intercept the ball and kick it back towards the human goal. To determine the references for the rods, the position and heading of the ball have to be estimated. Therefore one of the most important aspects of the project is how to localize and track the ball on the field. Several other project groups that have been working on the design of an automated table have used different methods to locate the ball. The University of Adelaide [1] mounted a laser grid underneath the feet of the puppets. By doing so, the ball was the only object that could breach the grid. The Georgia Institute of Technology [2] used an overhead camera and different colors for all objects in the field. With this property color segmentation could be used to segment the ball. An automated football table that eventually became commercially available was the KiRo project [3] which was later developed to StarKick [4]. By replacing the field with a transparent plate, the ball could be tracked from underneath the table.

Even though all of these implementations worked well, they also have their drawbacks. The laser grid used in the Adelaide project can only detect the ball when it does not bounce. This also goes for the camera mounted underneath the table. By mounting an overhead camera and using color segmentation the ball can be constantly found. However, processing 24-bit color images instead of 8-bit monochrome images triples the computational load on the workstation. Due to the highly discontinuous character of the ball's trajectory caused by bouncing, tracking should be based on fast updates instead of model based estimations. Processing images on a fast rate is therefore significant to accurately track the ball.

Rob Janssen, Jeroen de Best, René van de Molengraft and Maarten Steinbuch are with the Faculty of Mechanical Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands

In this paper a method is proposed that uses monochrome images for ball segmentation. By creating a mask of the field static objects such as the lines and the markings are removed from the image. By using real-time position information from the automated rods, a perspective projection algorithm is implemented to create a dynamic mask for the moving puppets in real-time. This mask is also removed from the images, such that the ball remains the only visible object.

In this paper first the hardware design is discussed. In Section III the Unified Modeling Language is adopted to create a structural software architecture, after which in Section IV the practical implementations are discussed. Finally the results are presented in Section V.

II. Hardware Design

A professional football table is acquired on which an overhead camera is mounted. The automated rod can translate or perform a kicking movement, see Fig. 1. The Prosilica

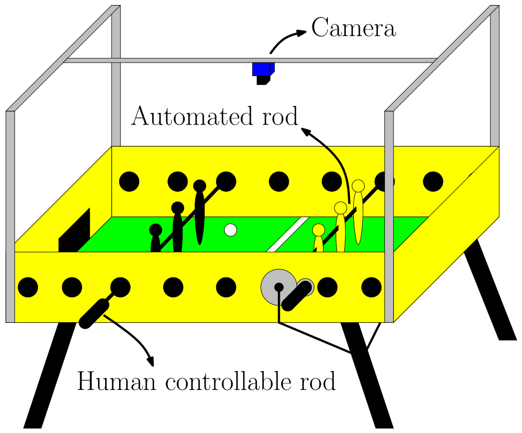

Fig. 1: Schematic overview of the table.

GC640c ethernet camera sends images to a HP xw4600 workstation where the images are processed. The workstation also communicates with the Beckhoff data acquisition equipment that allows to control the motors.

III. Software Architecture

The initial strategy is to control a single rod to intercept the ball and kick it back towards the human goal. To successfully execute this strategy, an *activity diagram* [5] is developed, see Fig. 2. In the activity diagram several components are directly related to motion control of the rod. These components have to be executed at a very high rate [9]) to successfully perform their task. Therefore these components are gathered in the *low level scheme*. The remaining

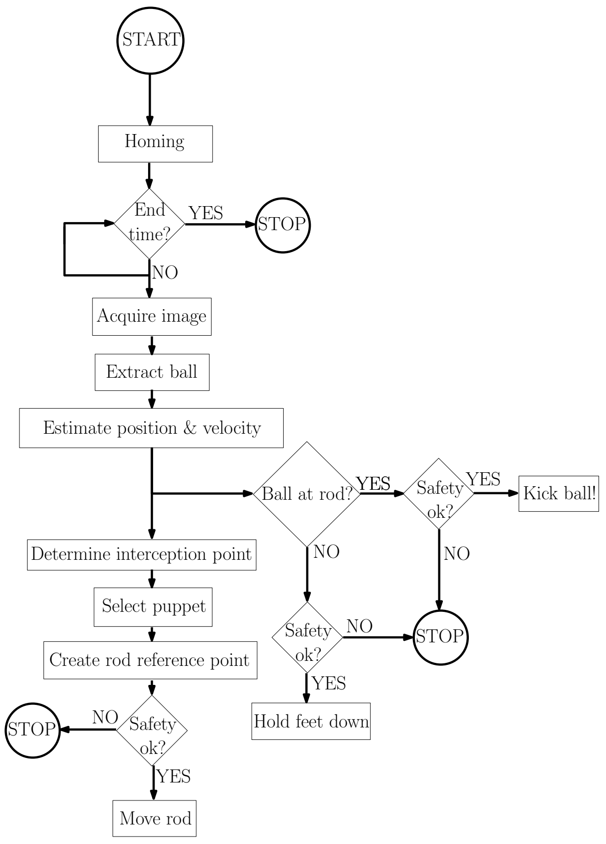

Fig. 2: Activity diagram of the football table.

components are responsible for the image processing and high level calculations of the references, and are gathered in the *high level scheme*.

IV. Implementations

A. High level scheme

The *high level* scheme calculates the translational movement of the rod and when to perform a kick. First an image is acquired from the camera. From this image the ball is segmented and by using an observer the heading of the ball is estimated. This estimation is used to determine the translation for the rod, and when to perform a kick. Below these steps are explained in detail.

Acquire image First an image is acquired from the overhead camera. A typical image is depicted in Fig. 3. In

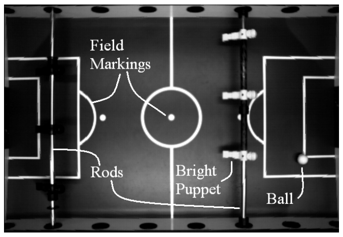

Fig. 3: Picture of the field.

this picture several objects are depicted, from which only the ball has to be segmented.

Segment ball Segmentation of the ball from its environment is stepwise explained in this section. The coordinate frame that is used throughout this section is depicted in Fig. 4.

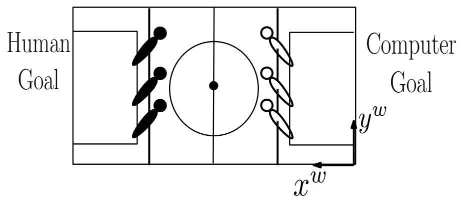

Fig. 4: Coordinate system.

1) Region of interest: To remove redundant image data, first a region of interest (ROI) is defined. The only object that can fully occlude the ball is the upper torso of a puppet, see Fig. 5. Worst case the ball rolls underneath the puppet

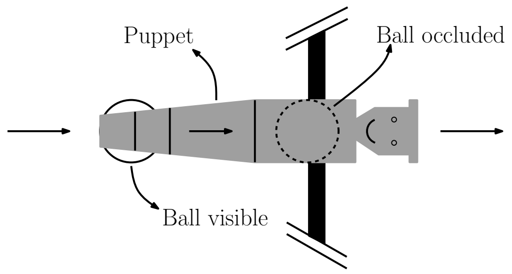

Fig. 5: Horizontal puppet occluding the ball.

with maximum velocity V_{max}. The minimum width for <u>half</u> the ROI $r_{min}^{x^w}$ depends on V_{max}, the frame rate f_i, the pixel resolution P_r and the ball radius r_b according to

$$r_{min}^{x^w} = \frac{V_{max}^{x^w} P_r}{f_i} - \frac{r_b}{2}. \tag{1}$$

However when an occlusion occurs at this speed the ball is lost, see Fig. 6. At $t = k$ the ball starts moving, where the

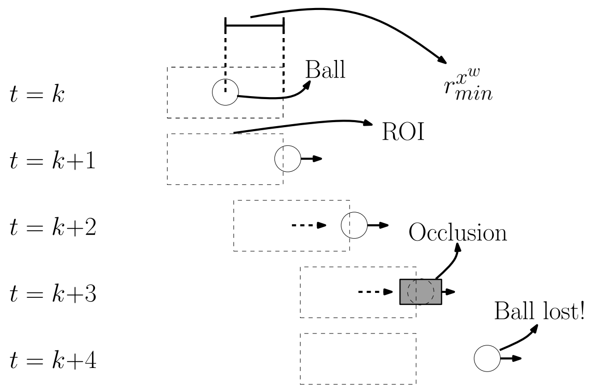

Fig. 6: Moving of the ball and lagging ROI.

following ROI always has one sample lag. At $t = k+3$ an occlusion occurs and the ball is lost. To find the ball in the next frame the width of the ROI is made twice as large. This is depicted in Fig. 7.

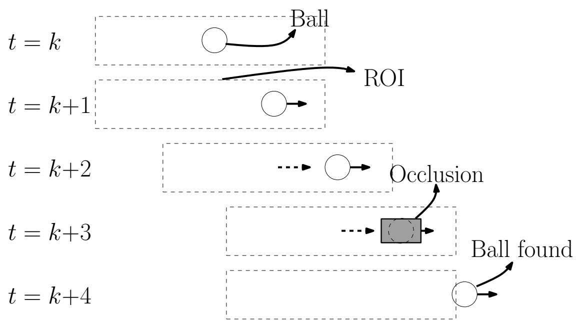

Fig. 7: ROI with double width.

The total width for the minimum ROI in the x^w direction therefore becomes

$$R_{min}^{x^w} = 4\frac{V_{max}^{x^w} P_r}{f_i} - r_b, \tag{2}$$

where P_r is 571.7 [pix/m] and r_b is 10 [pix]. Tests have shown that $V_{max}^{x^w}$ reaches speeds up to 5 [m/s] and $V_{max}^{y^w}$ up to 2.5 [m/s]. Therefore

$$R_{min}^{x^w} = 2R_{min}^{y^w}. \tag{3}$$

Eq. 2 has two unknown variables $R_{min}^{x^w}$ and f_i. A second relationship between these two variables can be found in the maximum allowable CPU load of the workstation. It is assumed that this is an inverse relationship, including a constant factor that describes calculations that are not influenced by the size of the ROI. The assumed model is

$$R_{min}^{x^w} \times R_{min}^{y^w} = \frac{C_1}{f_i} + C_2. \tag{4}$$

First Eq. 4 is combined with Eq. 3 to form a model with only two variables. This leads to

$$f_i = \frac{C_1}{\frac{1}{2}R_{min}^{x^w\,2} - C_2}. \tag{5}$$

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 fitted model of Eq. 5 are depicted in Fig. 8.

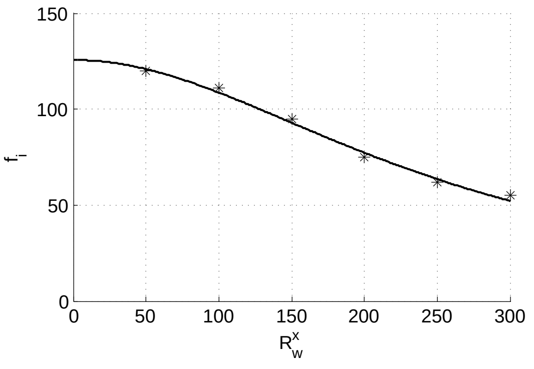

Fig. 8: $R_{min}^{x^w}$ against f_i at maximum computational load.

The values for C_1 and C_2 were obtained using an averaging optimization which gives

$$C_1 = 4.02e6 \text{ [pix}^2\text{/s]}, \qquad C_2 = -3.20e4 \text{ [pix}^2\text{]}. \tag{6}$$

By combining the validated model from Eq. 5 with Eq. 2 the minimum width for the ROI in the x^w direction can now be made dependent of V_{max} only, leading to the quadratic equation

$$\frac{2V_{max}^{x^w} P_r}{C_1}(R_{min}^{x^w})^2 - R_{min}^{x^w} = \frac{4V_{max}^{x^w} P_r C_2}{C_1} + r_b. \tag{7}$$

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 Eq. 3 a $R_{min}^{y^w}$ of 46.6 [pix]. From now on a ROI of 100×50 [pix] will be used. With these dimensions for the ROI, according to Eq. 5 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.

2) Removing static objects by creating a mask: Static objects include the field lines, the field dots and the rods that hold the puppets (but not the puppets themselves). These objects can be masked by using a static mask.

3) Masking yellow puppets: As can be seen in Fig. 3 the bright puppets also have to be masked. Therefore these puppets are chosen to be mechanically controlled, so that through the onboard encoders their position and orientation is available for processing. By combining this information with a 3D scan of the puppet in a perspective projection, a 2D mask of each of these puppets can be determined. For this 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, according to

$$\underline{D}_i^c = \mathbf{R}_s^r(\underline{O}_p^s + \underline{D}_i^p) + \underline{O}_r^c \tag{8}$$

where $\mathbf{R}_s^r$ is the rotation matrix, $\underline{O}_p^s$ is the transformation vector from the rod to the heart of the puppet and $\underline{O}_r^c$ is the transformation vector from the heart of the puppet to pixel coordinates D_i. A corresponding graphical interpretation is given in Fig. 9.

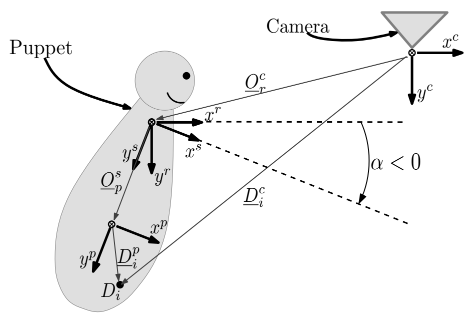

Fig. 9: 3D scan coordinates D_i^p to camera coordinates D_i^c.

The vector $\underline{O}_r^c$ is given by $[x_r^c\ y_r^c\ z_r^c]^T$. In this formula z_r^c is 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,

$$\mathbf{R}_s^r = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

After the pixel transformation the pixels can be placed into the image plane $[b_x^I, b_y^I]^T$ according to

$$b_x^I = \frac{f}{P_s}\frac{z_i^c}{y_i^c}, \qquad b_y^I = \frac{f}{P_s}\frac{x_i^c}{y_i^c}, \tag{10}$$

where f is the focal length parameter of 6 [mm] and P_s is the camera pixel size of 9.9 [μm]. The result is shown in Fig. 10. An animation can be found in [8].

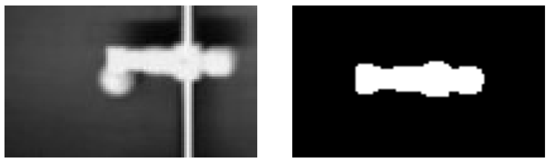
(a) Original image. (b) 2D projected mask.

Fig. 10: Original image of puppet and 2D projected mask.

4) Finding the ball: What is left over in the captured image now can only be the ball itself. This is depicted in Fig. 11.

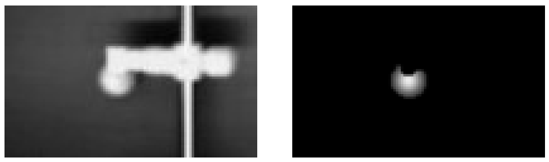
(a) Original image. (b) The left over.

Fig. 11: Original image and left over after masking.

The highest pixel value is assumed to be at the center of the ball.

Estimate position and velocity To intercept the ball the position and velocity have to be estimated . For this a *linear Kalman observer* is used. The standard deviation 3σ of the measurement noise covariance R is determined to be the radius of the ball r_b. The variance σ^2 then becomes $\frac{r_b^2}{9}$.

$$R = \begin{bmatrix} \frac{r_b{}^2}{9} & 0 \\ 0 & \frac{r_b{}^2}{9} \end{bmatrix} \tag{11}$$

where r_b is 10 [pix]. The estimate for the process noise Q can be done by evaluating the interception point error. This will be presented in the next section. **Determine interception point** The rods have to be controlled towards the point of interception where the rolling ball will cross the rod. This interception point $[i_k^{x^w}\ \ i_k^{y^w}]^T$ can be determined according to

$$i_k^{y^w} = \hat{y}_k^{y^w} + (i_k^{x^w} - \hat{x}_k^{x^w})\frac{\hat{\dot{y}}_k^{y^w}}{\hat{\dot{x}}_k^{x^w}} \tag{12}$$

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 estimates of the ball and $i_k^{x^w}$ is the static rod position in the x^w direction. To prevent the interception point to be determined outside the table another criterion must be added.

$$i_k^{y^w} = \begin{cases} \text{Eq. 12}, & \text{if } i_k^{y^w} > 0 \text{ and } i_k^{y^w} < \text{width of table}, \\ i_{k-1}^{y^w}, & \text{else.} \end{cases} \tag{13}$$

Due to bounces, the process noise covariance will mainly depend on the changing acceleration of the ball. The covariance matrix Q therefore is

$$\tilde{Q}_k = \begin{bmatrix} \frac{1}{2}\tilde{a}\Delta t^2 & \tilde{a}\Delta t & 0 & 0 \\ 0 & \tilde{a}\Delta t & 0 & 0 \\ 0 & 0 & \frac{1}{2}\tilde{a}\Delta t^2 & \tilde{a}\Delta t \\ 0 & 0 & 0 & \tilde{a}\Delta t \end{bmatrix}, \tag{14}$$

where the value of $\tilde{a}$ is estimated by manual tuning. The criterion for estimating $\tilde{a}$ is that when the human controlled rod returns the ball at full speed, the electro-mechanically controllable rod should still be able to intercept the ball. The interception point is accurate enough if it has an error of less than 0.015 [m], which is half the width of a puppet's feet. This holds for an $\tilde{a}$ of 20.000 [pix/s^2], see Fig. 12.

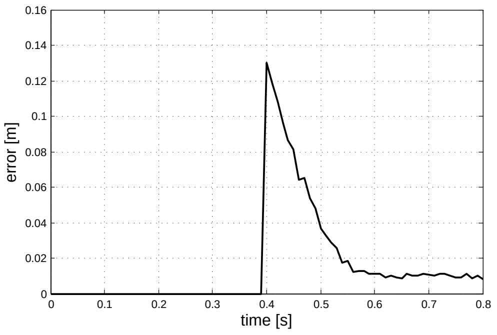

Fig. 12: Interception error in time.

Select puppet The reach of the puppets is depicted in Fig. 13.

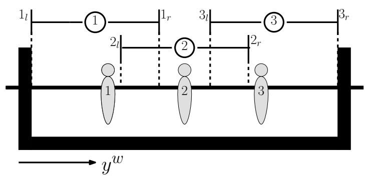

Fig. 13: Reach of the puppets.

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}. The latter criterium causes hysteresis which prevents constant switching between the selected puppet.

$$T_k = \begin{cases} 1, & \text{if } i_k^{y^w} < 2_l; \\ 1, & \text{if } i_k^{y^w} < 1_r \text{ and } T_{k-1}=1; \\ 3, & \text{if } i_k^{y^w} > 2_r; \\ 3, & \text{if } i_k^{y^w} > 3_l \text{ and } T_{k-1}=3; \\ 2, & \text{else} \end{cases} \tag{15}$$

Create rod reference point Now that a puppet is selected the equation for the reference point of the rod is

$$r_k^{y^w} = i_k^{y^w} + P_b(1 - T_k) \tag{16}$$

where P_b is the equidistant pitch between the puppets.

Kick ball A kick is simply performed when the estimated position of the ball is near the rod.

B. Low level scheme

In the *low level* scheme the references for the rods are executed. Due to discretization first the references have to be smoothed. After this a stabilizing controller is used to execute the references. **Reference smoother** The reference trajectory for the rods and the kick command are determined at a rate f_i of 100 [Hz]. The low level control scheme runs at a rate of 2000 [Hz]. To create a smooth reference for the rod in the low level scheme, a second order smoothing algorithm is implemented [7]. 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

$$\ddot{w}(t) = q_w(t) \tag{17}$$

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}. In Fig. 14 this smoothing is depicted.

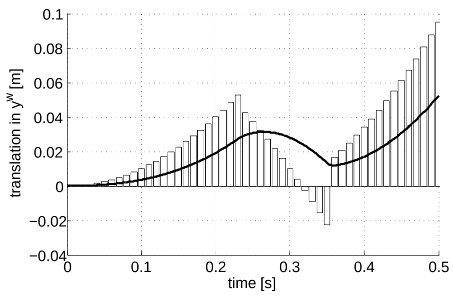

Fig. 14: Input trajectory and smoothed trajectory.

Controllers Because the system was designed to be decoupled, have little friction and very stiff connections, two SISO leadlag filters for each DOF are sufficient [6] to stabilize the system.

V. Results

The results are categorized in two sections, the *ball tracking performance* and the *success rate of the intercepts*.

A. Ball tracking performance

When the ball gets outside the ROI 3 possible causes can be examined.

- motion blur of the puppets causes,
- high velocities of the ball,
- occlusion by a puppet.

1) Motion blur puppets: Motion blur causes the projection of a fast moving puppet to 'stretch out' in the captured image, and appear outside the bounds of the mask. The ROI will then focuss on the puppet instead of the ball. Testing has shown that the mask fully covers the puppet at all times. Motion blur is therefore not the cause for losing the ball.

2) Ball velocities: In Fig. 15 the thick line indicates when the ball is lost. What can be noted from this picture is that

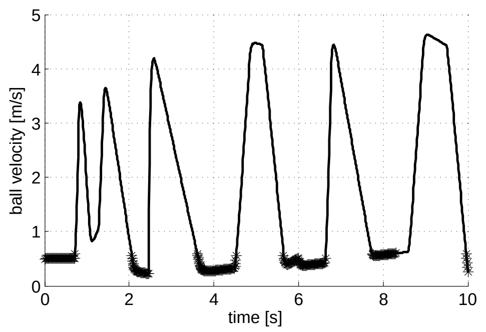

Fig. 15: Ball found versus ball speed.

the ball is never lost when the velocity is at its maximum, 4.5 [m/s].

3) Occluding rod movements and ball velocities: From Fig. 15 the ball seems to be lost at low velocities. Low velocities typically occur when the ball is controlled by a puppet. If the occluded ball then moves outside the ROI, it is lost.

B. Success rate of intercepts

Sometimes an intercept fails, which can be caused by 3 different factors:

- the estimation accuracy of the interception point,
- control accuracy of the rod,
- process time delay.

These factors are individually evaluated.

1) Estimation accuracy of interception point: In Fig. 12 the maximum error in the point of interception was determined to be 15 [mm].

2) Controlling the rod: The error of the translational DOF in the low level control loop discussed in Section IV-B 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. 16.

In this figure it can be seen that the maximum error is 7 [mm]. From the Power Spectral Density (PSD) plot of the

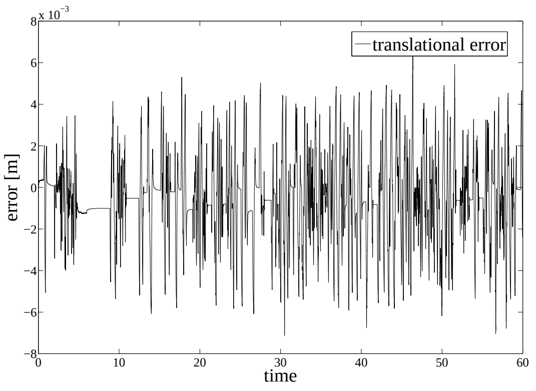

Fig. 16: Error in time.

error in Fig. 17 can be seen that a large peak occurs at 100 [Hz]. This peak is due to a zero-order hold effect caused by

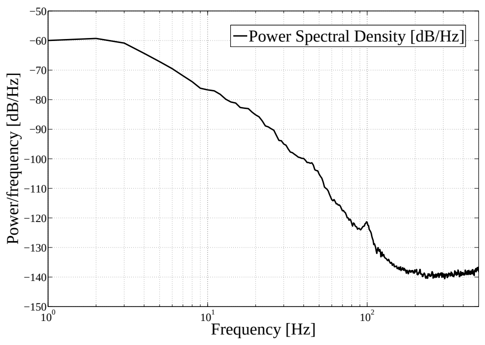

Fig. 17: PSD plot of the error.

the sample rate conversion of the high level update rate in the low level motion scheme.

3) Time delay: Time delay contains the total amount of time it takes to capture, process and actuate the rod after the ball was positioned on the field. An estimate of the worst case delay can be done by evaluating all the individual processing steps that were performed.

	Max. possible delay [ms]
exposure time	5
image buffering	15
high level reading	5
high level processing	10
low level reading	10
low level processing	0.5
low level output	0.25

The maximum amount of time delay therefore is 45.75 [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 114 [mm].

Conclusion on intercept accuracy Three different factors were examined. The corresponding errors are listed in the tabel below.

	Max. possible error [mm]
interception point accuracy	15
control of the rod	7
time delay	114
Total worst case error	**136**

The maximum error is therefore 136 [mm], where half the width of a puppets feet was 15 [mm]. The main cause is time delay. Without time delay the maximum error would be 22 [mm], which is fairly around the allowable margin.

VI. Conclusions

In this paper several steps where dscribed to design an automated football table:

1) a hardware design was made,
2) a software design was described that holds two schemes,
3) the *high level scheme* processes the images and creates references for the rod,
4) the *low level scheme* executes this reference in a stable and safe manner.

Fundamental to the overall result of the table was the ability to keep the ROI located on the position of the ball. The main cause for losing the ball was occlusion by one of the puppets. By far the most important cause for a wrong intercept was time delay. For a maximum possible error of 136 [mm], time delay was responsible for 114 [mm]. Creating an online estimator for the delay and using another type of observer for the ball [10], [11] could greatly improve the accuracy of the interceptions and the overall performance of the table.

References

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

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

[3] T. Weigel and B. Nebel, KiRo - An Autonomous Table Soccer player, *Robocup 2002: Robot Soccer World Cup VI*, vol. 2752, 2003.

[4] Gauselmann GmbH, StarKick, 2005, http://www.merkur-starkick.de/.

[5] Object Management Group, Unified Modeling Language, Pasadena California, 1997, http://www.uml.org.

[6] Y. Oded and M. Nagurka, Automatic Loop Shaping of Low Order QFT Controllers, *IEEE Electrical and Electronics Engineers in Israel*, 2004, pp 29-32.

[7] O. Purwin and R. D'Andrea, Trajectory Generation for Four Wheeled Omnidirectional Vehicles, *IEEE American Control Conference*, vol. 7, no. 0743-1619, 2005, pp 4979-4984.

[8] R. Janssen, Perspective Projection of the Puppets, Youtube, http://www.youtube.com/watch?v=FhK2ua1hpo0.

[9] G. Franklin and D. Powell and A. Emami-Naeini, Feedback Control of Dynamic Systems, Addison Wesley, ed. 3, 1995, ISBN 0-201-52747-2.

[10] S. Arulampalam and S. Maskell and N. Gordon and T. Clapp, A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking, *IEEE Transactions on Signal Processing*, vol. 50, 2001, pp 174-188.

[11] D. Simon, Optimal State Estimation: Kalman, H infinity, and Nonlinear Approaches, John Wiley & Sons, vol. 1, 2006, ISBN 978-0-471-70858-2.