Real-Time Ball Tracking In A Semi-Automated Foosball Table

Rob Janssen, Jeroen de Best, René van de Molengraft

Eindhoven University of Technology
Department of Mechanical Engineering
Den Dolech 2, 5600MB Eindhoven, Netherlands

Abstract. In this article a method is proposed for ball tracking using 100 Hz computer vision in a semi-automated foosball table. In this application the behavior of the ball is highly dynamic with speeds up to 10 m/s and frequent bounces occur against the sides of the table and the puppets. Moreover, in the overhead camera view of the field the ball is often fully or partially occluded and there are other objects present that resemble the ball. The table is semi-automated to enable single user game play. This article shows that it is possible to perform fast and robust ball tracking by combining efficient image processing algorithms with a priori knowledge of the stationary environment and position information of the automated rods.

Key words: computer vision, automated foosball table, visual servoing, ball segmentation, object tracking, perspective projection, Kalman observer, real-time systems

1 Introduction

Tracking a ball in a highly dynamic and non predictive environment by use of computer vision becomes difficult when the ball is occluded or surrounded by similar looking objects. An example can be found in the Robocup environment, where autonomous robots have to track a ball and grab it [1]. Another example can be found in the Hawkeye computer vision system [2], where cameras are positioned alongside a tennis court to track the ball. In this article the considered environment is the soccer field used in a foosball table, and the object to be tracked is a white ball of approximately 3.5 cm in diameter. The occluding objects are the puppets and the rods on which these puppets are mounted. In the captured images the ball can cross a white field line, or it can resemble one of the white dots on the field. On one side of the table the rods are electro-mechanically controlled. These puppets need to intercept and return the ball, and therefore ball tracking must be fast and accurate. Other project groups that have been working on the development of an automated foosball table used various techniques to track the ball. The Danish University of Technology has used different colors for the ball and the environment [3]. In this setup, color segmentation [4] can be used to segment the ball from its environment.

The University of Victoria developed a laser grid to acquire the coordinates of the ball on the field [5]. They positioned the laser grid such that the lasers would go underneath the puppets and therefore only the ball could cross it. An automated foosball table that eventually became commercially available comes from the University of Freiburg and is named KiRo [6]. This table was developed to a commercial version named StarKick [7]. This group mounted a camera underneath their table and replaced the standard field with transparant glass so that the camera could see the ball from below.

Although all of these automated tables worked and some were able to defeat professional players, the methods they used to track the ball were not flawless. Color segmentation only works on a table where the ball has a different color than its environment. With a laser grid that is mounted beneath the feet of the puppets, the ball can be found by determining where the grid has been broken. However, performance lacks because in a real game the ball bounces such that robust tracking becomes cumbersome. These bounces also make ball tracking difficult in the KiRo and StarKick projects.

In this article a different approach is used to track the ball. A priori knowledge of the environment and perspective projection algorithms are used to segment the ball from its environment. An observer is used to obtain estimates for the position and the velocity. As the table is meant for professional foosball play, after any alterations it should still comply to the official USTSA regulations defined in [8]. There are two more restrictions made on the setup.

- the used camera is mounted above the table,
- the camera can only capture monochrome images,

In this article first the hardware will be explained briefly. In the second part the segmentation of the ball will be explained, and the observer that was used. In the last part the results are discussed, which will focus on the robustness of the image processing algorithms and their real-time performance. Finally, a small summary and a prospect on the overall performance on the table will be given in section 5.

2 The hardware

To have a platform that complies with the USTSA regulations a professional foosball table is acquired. A steel frame is placed on top of the table, on which a camera [9] is mounted. One side of the table is equipped with electro-mechanically controlled rods. This way the puppets can be moved towards a certain position, and they can also lift their legs or perform a kicking movement. A schematic overview of the whole setup is depicted in Fig. 1.

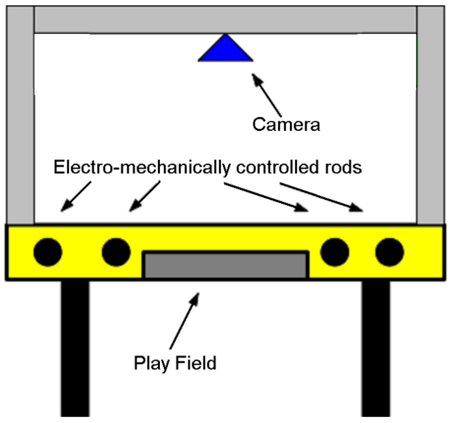

Fig. 1. Schematic overview of the table.

The EtherCAT module [12] that is used for the data acquisition allows to control the rods at 1 kHz. The desktop PC that processes the images and interfaces with the EtherCAT module is an Intel Dual Core 2GHz processor, running a Linux low latency kernel. A complete overview of the connections between the hardware components and their corresponding dataflow is depicted in Fig. 2.

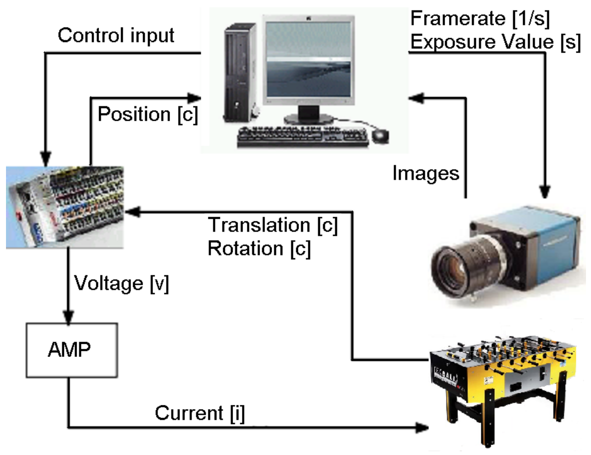

Fig. 2. The different components and their dataflow.

3 Ball segmentation

To obtain the coordinates of the ball, the ball has to be segmented from its environment. Possible methods to segment the ball from its environment are the Circular Hough Transform [10] or more advanced methods such as gradient based circle detection algorithms [11]. There are three reasons why these methods are not applicable in this setup.

- creating a 3D parameter space requires a lot of storage and is therefore not preferable in real-time applications,
- in an occluded situation the ball often does not resemble a circle,
- there are other objects present that resemble the ball such as the white dots and the heads of the puppets,
- the presence of nonconstant light intensities make:
  - the radius criterion used in the CHT difficult to define
  - the intensity gradients differ throughout the field

Therefore in this article a different method is chosen to segment the ball. The coordinate frame that is used throughout this article is depicted in Fig. 3.

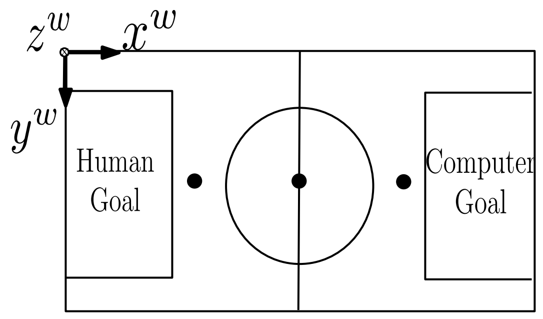

Fig. 3. The field and the coordinate system.

3.1 Defining the region of interest

The first step in the segmentation algorithm is to crop the captured images to a region of interest, or ROI, in order to remove redundant image data. The largest object that can occlude the ball is the upper torso of a puppet. This upper torso is 30x20 pixels. When the ball of 20 pixels in diameter rolls underneath, it can become completely occluded. The position of the ROI is programmed to remain static when the ball is lost. When the ball is partially occluded the camera needs approximately 1/4th of the diameter of the ball to detect it. Worst case scenario could be, that the ball rolls underneath a puppet while the ROI stays at the position where it lost the ball. This is schematically depicted in Fig. 4.

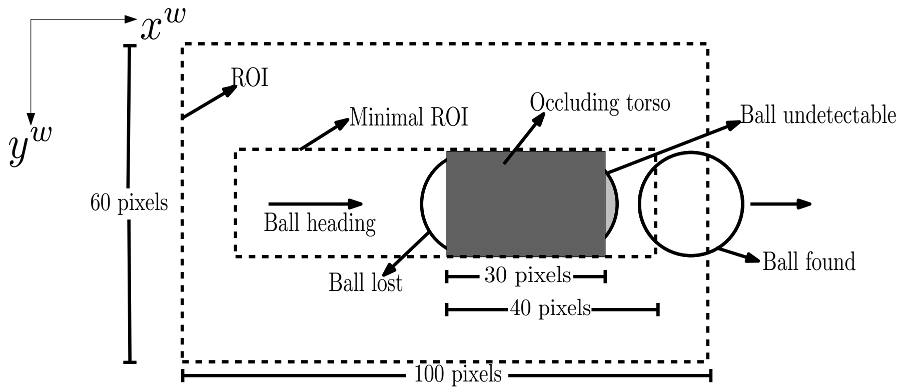

Fig. 4. Worst case loss of ball.

A length of 40 pixels allows the ROI to find the ball when it reappears. Because there can be made no distinction between ball movement in the positive x^w and in the negative x^w direction, the minimum size for the ROI in the x^w direction (as defined in Fig. 3) should be 80 pixels. The minimum size for the ROI in the y^w direction should be 20 pixels as it is the diameter of the ball, but in this direction also the rods have an influence in occluding the ball. Therefore some margins need to be taken into account. A good estimate for the size of the ROI is 100x60 pixels. An example of such an ROI is given in Fig. 5.

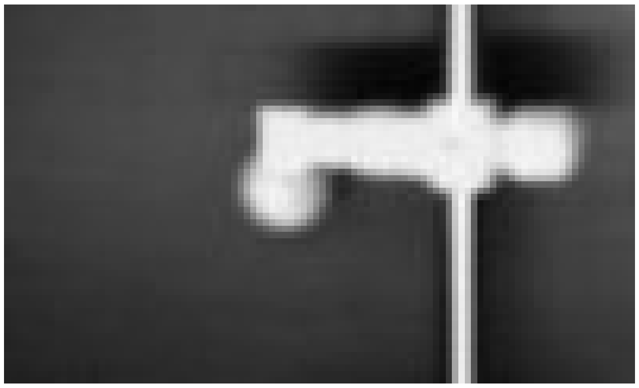

Fig. 5. Region of interest 100x60 pixels.

3.2 Determining the minimum required frame rate

With the minimum size for the ROI determined, also the minimum frame rate at which the image processing algorithms have to run can be calculated. Tests have shown that the maximum ball speed V_{max} can reach up to 10 m/s. It can be assumed that the ball experiences this maximum velocity only in the x^w direction. Because the ROI was chosen such that the ball cannot be occluded twice at its maximum velocity, the radius of the ball r_{ball} has to be added to half the length of the ROI in x^w to determine the maximum distance over which the ball is allowed to travel. The minimum frame rate FPS_{min} can then be calculated as

$$FPS_{min} = \frac{V_{max} P_{res}}{r_{ball} + \frac{1}{2} ROI_{x^w}}, \tag{1}$$

where P_{res} is the pixel resolution of 584 pixels/m. Therefore the minimum frame rate is determined to be 97.3 Hz. In the following steps a frame rate of 100 Hz will be assumed.

3.3 Undistorting the image

To find the correct coordinates of the ball on the field, the images that come from the camera have to be undistorted. To solve for this distortion, a camera calibration is carried out using the Camera Calibration Toolbox for Matlab [13]. The resulting distortion coefficients are then used to restore the images as shown in Fig. 6.

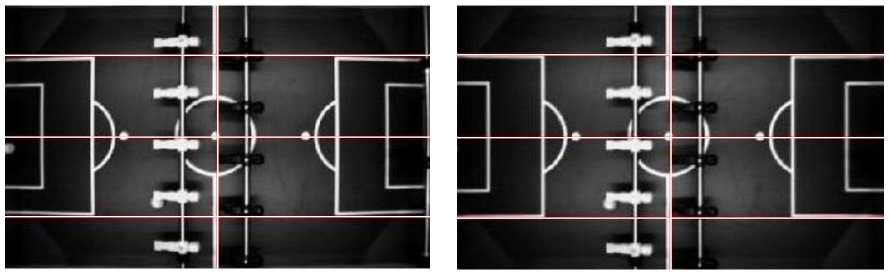
(a) Original image (b) Restored image

Fig. 6. The original and the restored image. Straight reference lines in white.

3.4 Removing static objects by creating a mask

In the foosball 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 in shape or intensity value, and should therefore be disregarded when searching for the ball. This can be done be creating a static mask. This mask can then 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. 7.

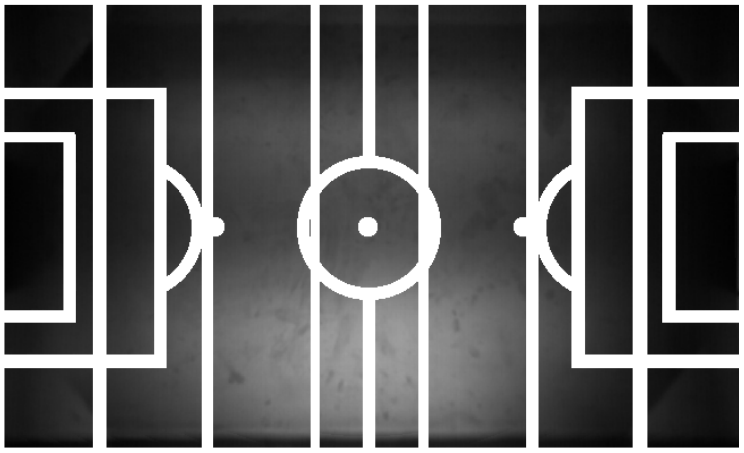

Fig. 7. The mask that removes the static objects.

3.5 Masking the yellow puppets

As can be seen in Fig. 6 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 during a game their position and orientation will be available for processing. With this information the 3D position and orientation of the puppet 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, scanner. To perform this 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 given below.

$$\underline{D}_i^c = \mathbf{R}_s^r(\underline{O}_p^s + \underline{D}_i^p) + \underline{O}_r^c \tag{2}$$

A corresponding graphical interpretation is given in Fig. 8.

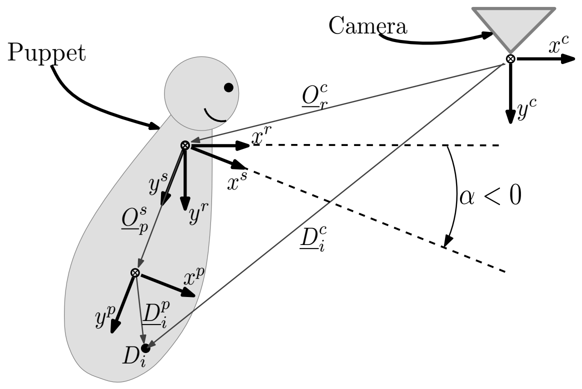

Fig. 8. The transformation of the scan coordinates D_i^p to camera coordinates D_i^c.

The vector $\underline{O}_r^c$ describes the translation from camera frame to rod frame and consists of $[x_r^c\ y_r^c\ z_r^c]$ where 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 computer controlled variable α.

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

After the pixel transformation the pixels can be placed into the image plane $[b_x^I\ ,\ b_y^I]$ as depicted in Fig. 9

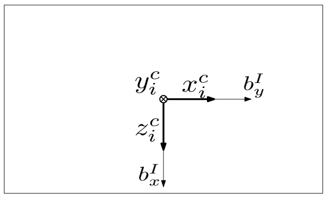

Fig. 9. 3D camera pixel coordinates to pixels in the image plane.

according to

$$b_x^I = \frac{\mathbf{f}}{P_s}\frac{z_i^c}{y_i^c} \tag{3}$$

$$b_y^I = \frac{\mathbf{f}}{P_s}\frac{x_i^c}{y_i^c} \tag{4}$$

where $\mathbf{f}$ is the focal length parameter of 6 mm and P_s is the camera pixel size and is 9.9 μm.

A result of the perspective projection is given in Fig. 10. This way the mask is thus fully constructed from the known geometry of the puppet and the measured position and orientation.

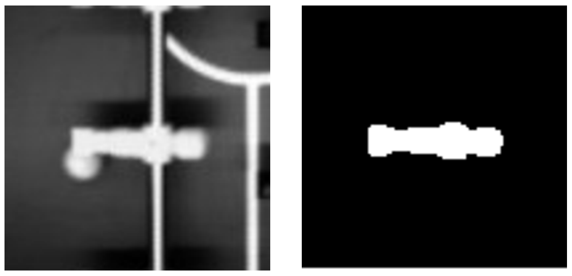
(a) Original image. (b) 2D projected mask.

Fig. 10. The original image of the puppet and the 2D projected mask.

3.6 Thresholding the dark puppets

The dark puppets are removed by simply thresholding all that is left over in the image.

3.7 What is left over

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. An example of how the ball looks like when the masking and thresholding has been done, is shown in Fig. 11.

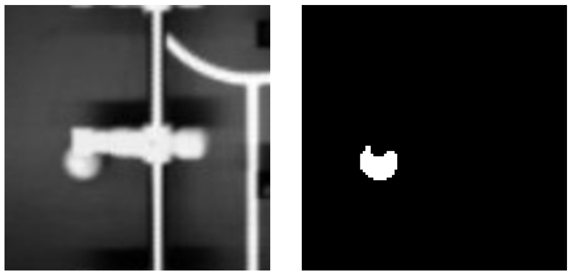
(a) Original image. (b) The left over.

Fig. 11. The original image and what is left over after masking and thresholding.

Finding the center of the ball is done by take the average of the minimum and maximum pixel coordinates x^w_{min} and x^w_{max} for the x^w direction, and the minimum and maximum pixel coordinates y^w_{min} and y^w_{max} for the y^w direction. Therefore the center of the ball $[x^w_c\ ,\ y^w_c]$ becomes

$$x^w_c = \tfrac{x^w_{min}+x^w_{max}}{2} \tag{5}$$

$$y^w_c = \tfrac{y^w_{min}+y^w_{max}}{2} \tag{6}$$

In case the ball is occluded this formula will not give the exact center of the ball. This will be solved for by using a Kalman observer as explained in the next section.

4 Controlling the rods

To predict the movement of the ball and to calculate the point of interception, the velocity of the ball is determined. For this a linear Kalman observer [14] is implemented.

4.1 The Kalman observer

To let the Kalman observer converge as fast as possible to an accurate state estimate, the estimates for the process noise covariance $\tilde{Q}$ and measurement noise covariance $\tilde{R}$ have to be accurate. Estimating the measurement noise covariance $\tilde{R}$ can be done by looking at the error between a static ball position and the corresponding measurements. In case the ball is almost fully occluded, the actual measurement of the bal will be completely on the outer edge of the ball. Therefore the maximum value that the measurement noise covariance $\tilde{R}$ can have is the radius of the ball squared. This value has to be given in pixels and is determined to be

$$\tilde{R} = \begin{bmatrix} {r_b}^2 \\ {r_b}^2 \end{bmatrix} \tag{7}$$

where r_b is the radius of the ball and is approximately 10 pixels. The estimate for the process noise $\tilde{Q}$ is more difficult to determine. The easiest way is to perform an off-line analysis of the logged measurements. The distance between the human controlled attacker and the electro-mechanically controlled keeper is 30 cm, and the maximum velocity of the ball can be 10 m/s. Therefore tuning of the process noise covariance should lead to a maximum convergence time of 0.03 s. In this period the error between the measurement and the actual position estimate should converge to less then the width of the puppet's feet, which is 3 cm. Because the ball bounces and the acceleration was not estimated it can be assumed that $\tilde{Q}$ depends on a deviation of the acceleration of the ball. Movement of the ball in x^w and y^w is decoupled, and therefore $\tilde{Q}$ will only have values on its diagonal.

When the position data is evaluated a good estimate for $\tilde{Q}$ becomes

$$\tilde{Q}_k = \begin{bmatrix} \frac{1}{2}\tilde{a}\delta t^2 & 0 & 0 & 0 \\ 0 & \tilde{a}\delta t & 0 & 0 \\ 0 & 0 & \frac{1}{2}\tilde{a}\delta t^2 & 0 \\ 0 & 0 & 0 & \tilde{a}\delta t \end{bmatrix} \tag{8}$$

where $\tilde{a}$ is about 20.000 pixels/s^2. With this value the outcome for the Kalman filtered estimate in x^w and y^w (as defined in Fig. 3) is given in Fig. 12. When a

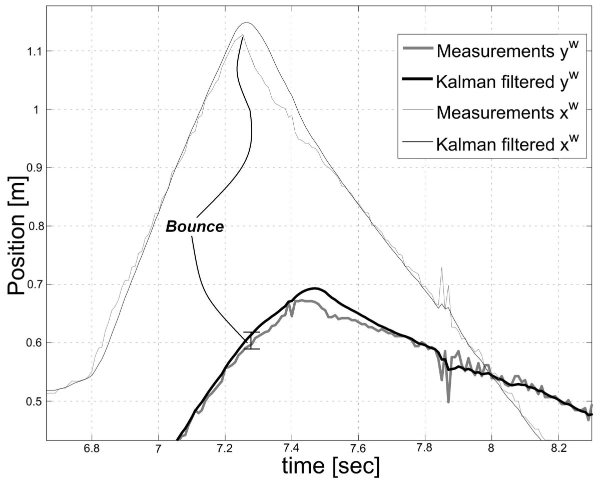

Fig. 12. Measurement data and the Kalman filtered estimate in x^w and y^w direction.

bounce occurs in the x^w direction at t=7.26 s it can be seen that the error in the y^w direction converges to less than 3 cm in 0.03 s, which is accurate enough for the mechanically controlled keeper to intercept the ball. An animated illustration where the points of interception are determined can be found at [15].

5 Summary and prospect

In this article a method was proposed for ball tracking using 100 Hz computer vision in a semi-automated foosball table. A region of interest was defined to reduce the dataflow. Undistortion and static masking were used to remove the static objects. Perspective projection and dynamic masking were applied to mask the bright puppets that resemble the ball in intensity values. Finally an observer was implemented to direct the rods smoothly to their point of interception with the ball. This observer was fast and accurate enough for the keeper to prevent a goal from an opponent that shoots from its most forward attacker at the maximum assumed velocity of 10 m/s. Currently the table is being tested and the results are promising. At this point the image processing algorithm has a 100% found-ratio of the ball. This means that nobody was able to play the ball in such a way that it went out of the region of interest as defined in 3.1. Even a professional player with tactical skills was not able to mislead the algorithm. A video is made available at [16] which indicates the region of interest and the mask of the electro-mechanically controlled puppets. More videos will be made available during the testing phase. When the table has successfully undergone this phase, it can also serve as a test bed for high level strategic control schemes and learning algorithms. The results from these tests can then be used in other research areas such as the Robocup robot soccer domain.

Acknowledgements The author wants to thank G.J.Strijkers for his assistance in creating the 3D scan of the puppets that was used in section 3.5.

References

1. Bruijnen,D.J.H., Aangenent,W.H.T.M., Helvoort,J.J.M., Molengraft,M.J.G.v.d., Steinbuch,M.:From Vision To Real-time Motion Control For The Robocup Domain. IEEE Conference on Control Applications, Singapore (2007)
2. Owens,N., Harris,C., Stennet,C.: Hawk-Eye Tennis System. International Conference on Visual Information Engineering, ISBN 0-85296-757-8, pp 182–185, (2004)
3. Danish University of Technology: Automated Foosball Table, `http://foospmp.myl.dk/`
4. Gonzalez,R.C., Woods,R.E.: Digital Image Processing. ISBN 978-0-13-505267-9, Pearson Prentice Hall, 3rd edition, pp 445–450, (2008)
5. Extreme Automation Foosball Table, `http://web.uvic.ca/~tiran/`
6. University of Freiburg: KiRo - The Table Soccer Robot, `http://www.informatik.uni-freiburg.de/~kiro/`
7. Gauselmann StarKick, `http://www.merkur-starkick.de/`
8. USTSA Foosball Rules Of Play, `http://www.foosball.com/learn/rules/ustsa/`
9. Prosilica GC640 Gigabit Ethernet Camera , `http://www.prosilica.com/products/gc640.html`
10. Ballard, D.H. : Generalizing The Hough Transform To Detect Arbitrary Shapes, Pattern Recognition vol.13 no.2, pp 111–122 (1981)
11. Rad, A.A., Faez, K., Qaragozlou, N.:Fast Circle Detection Using Gradient Pair Vectors, Proc. 8th Digital Image Computing: Techniques and Applications, Syndey, (2003)
12. Beckhoff Ethernet for Control Automation Technology , `http://www.beckhoff.com/english.asp?ethercat/`
13. California Institute of Technology: Camera Calibration Toolbox For Matlab, `http://www.vision.caltech.edu/bouguetj/calib_doc/`
14. Kalman, R.E., Bucy,R.S.: New Results In Linear Filtering And Prediction Theory. Transactions of the ASME. Series D: Journal of Basic Engineering, vol.83, pp 95–108, (1961)
15. Eindhoven University Automated Foosball table, animated video 1 `http://nl.youtube.com/watch?v=d7xgL5mEyeY`
16. Eindhoven University Automated Foosball table, animated video 2 `http://nl.youtube.com/watch?v=TN2ENdw3Gac`