BACHELOR PAPER

Term paper submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Sports Equipment Technology

Ball Tracking Algorithms for Table Soccer

Game State Analysis

By: Ivo Ackermann
Student Number: 1310327010

Supervisor: DI (FH) Stefan Litzenberger MSc

Vienna, September 19, 2016

FH University of Applied Sciences
TECHNIKUM WIEN

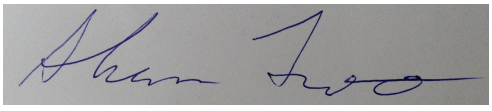
Declaration

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (for example see §§21, 42f and 57 UrhG (Austrian copyright law) as amended as well as §14 of the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien).

In particular I declare that I have made use of third-party content correctly, regardless what form it may have, and I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see §14 para. 1 Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.“

Vienna, September 19, 2016 Signature



Kurzfassung

Es wurde untersucht, mit welchen Algorithmen der Ballverlauf im Videostream eines Tischfußballspiels erkannt werden kann. Es konnte gezeigt werden, dass eine Kombination von Algorithmen für ein realistisches Setting von Videoauflösung und zur Verfügung stehender Hardware geeignet ist, den exakten Ballverlaufspfad im Profi-Tischfußball für statistische Zwecke hinreichend zuverlässig zu berechnen.

Schnelle Algorithmen erwiesen sich in der Praxis als zu unzuverlässig. Langsame, zuverlässigere Algorithmen können in dem Takt der Framerate von 30Hz nur kleinere Bildbereiche analysieren. Deswegen wurden die Methode der kleinsten Quadrate eingesetzt um rasch wahrscheinliche Ballpositionen zu berechnen. Rund um diese wahrscheinlichen Ballpositionen wurde dann lokal die Circular Hough Transform (kreisförmige Hough Transformation) verwendet. Lokal kann diese rasch genug zu hinreichend zuverlässigen Ergebnissen kommen.

Auf Basis dieser Arbeit wurde eine Software zur automatischen Erstellung von Statistiken im Tischfußball entwickelt.

Schlagworte: Videoanalyse, Tischfußball, Pattern Matching, Circular Hough Transform

Abstract

Algorithms capable of tracking the state of a table soccer game were compared in terms of accuracy and speed. For a realistic setting, given the video resolution and hardware owned, algorithms were found that are capable of computing the ball's track in real time and with enough accuracy for statistical purposes.

Fast algorithms proved to be unreliable. Slower algorithms are only capable of analysing smaller areas in a 30 frames per second setting. Therefore least squares method was used to compute likely ball positions. Around these likely ball positions Circular Hough Transform (CHT) was used to verify the position. Applied locally CHT is fast and accurate enough.

A software to automatically calculate statistics in table soccer based on this paper was developed.

Keywords: Video analysis, Table soccer, Foosball, Pattern Matching, Circular Hough Transform

Acknowledgements

Thank you to Kozoom for providing me with an internship and the equipment necessary to complete it, especially to Michael Paffrath BSc for supervising and supporting me during the internship.

Thank you to DI (FH) Stefan Litzenberger MSc for supervising this paper.

I would also like to thank Dr. Wiltrud Breuss, Dr. Louise Jottrand and Georg Sedlitz for moral support, reviewing and proof reading the paper.

Thank you to DI Gerhard Ackermann for reviewing the structure of the paper.

Contents

1 Introduction

This paper presents the results of the author's internship at Kozoom Multimedia (Andernos les Bains, France). Kozoom.com hosts streams of several sports including table soccer. This paper deals with video analysis of table soccer. The focus of the internship was to develop a piece of software which is able to automatically produce statistics out of a video of a table soccer game. Ball recognition is needed to produce statistics about ball possession, top ball speeds, to create slow motion replays, or to mark the ball in the video to make the game easier to follow.

These and similar features have been used in tennis since 2006 Baodong (2014). However, automation of these features in table soccer is non-trivial.

The main part of this paper will deal with tracking the ball in a video of a table soccer game, as it is the part that requires the most explanation as well as processing power.

1.1 Aim of Paper

The aim of this paper is to present the work and results of the author's internship project on ball tracking in table soccer.

First the premise of the project as well as its goals will be described. Then the various methods will be presented, and in particular some algorithms used to achieve these goals. Test results and their evaluation will be concluded with.

2 Project Goals

Kozoom owns video data of table soccer tournaments of the last ten years and is interested in giving the viewer more information about the game, because it can be hard to follow. Many ideas like slow motion goal replays, and highlighting the ball were considered. Once the ball's track is known it can be used for further processing.

Examples of further processing are:

- marking the ball in the video
- interpolation for slow motion replays
- recognition of begin and end of a game or game situations

- recognition of ball possession
- calculation of ball possession duration
- shot type statistics
- player statistics for players and trainers
- ball position statistics

The project goal as defined by Kozoom was to prove if it is feasible to track the ball in the given technical setting and if so, to find out the most suitable algorithms and use these to provide statistics on games for their users.

3 State of the Art

The first publication on camera based table soccer analysis was about KiRo (Weigel and Nebel (2002) followed by Nebel et al. (2005)), which was the first instance of a device capable of playing table soccer autonomously. It was later developed into the first commercially available table soccer robot called StarKick (Weigel (2005)). KiRo used a camera that was not connected to the table, but was positioned above the table's center. It calibrated the picture via the direction and size of the middle field line and circle (Weigel and Nebel (2002)). This was possible because KiRo was developed for a specific table. KiRo defined the position of occluded balls as the figure nearest to the last known position of the ball. StarKick changed the camera's position to be in the body of the table and monitoring the field through the field's base plate (Weigel (2005)).

The research group around Alsalihi et al. (2011) developed an autonomous table soccer robot based on a mesh of infra-red lasers and photo-resistors. Other non camera based solutions have been developed.

Many projects on autonomous table soccer robots that used a camera had it fixed to the table (Figure 1). Kozoom uses a camera which is not connected to the table (Figure 2). Kozoom's setup causes parameters, like figure color and artificial lighting, to be different in every video.

In order for autonomous table soccer robots to achieve their aim of playing the game, knowing the position of the ball is essential. Their methods for tracking the ball are however dependent on the setup they use (camera above the field or mounted on a table and using a specific table). Kozoom's video streams do not have a fixed camera which leads to the need for finding applicable methods for tracking the ball.

Least Squares method, which was heavily used in this paper can be traced back to Carl Friedrich Gauss (Stigler (1981)).

Figure 1: Contraption for mounting a camera over table's center (Source: Bambach and Lee (2012))

A method for shape recognition by algorithm was developed and patented by Hough (1962). A developed version of this idea, the Hough Transform, is still used for finding parametrizable shapes in images (lines (MathWorks Deutschland (2016b)), circles (MathWorks Deutschland (2016a))).

Recent developments in object recognition usually focus on classification of images in order to name them (Grauman and Leibe (2011)).

4 Background

4.1 Kozoom, the Company cooperated with

Kozoom.com is a website published by the company Kozoom Multimédia, which is part of Kozoom Group SAS (Andernos les Bains, France). Kozoom Multimédia was founded in September 1998 as a Société à Responsabilité Limitée with a capital of 8.000 Euros. Carrer (2016)

Kozoom has been streaming table soccer games since 2006. Kozoom sells subscriptions for premium access to features of their homepage including view on demand and live-streams.

Figure 2: Diagram of Kozoom's setup (Source: Paffrath (2016))

Without a subscription you get access to non premium live streams, tournament results, photos, news and a forum.

4.2 Table Soccer

Around 1890, games similar to table soccer were invented. Table soccer as we know it today was patented by Thornton (1922).

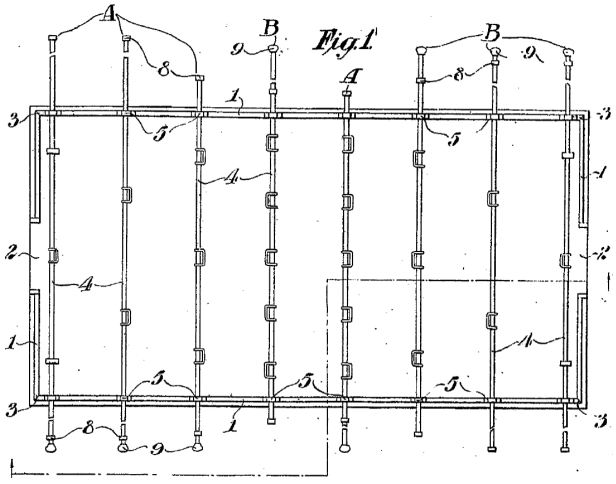


Figure 3: Drawing representing the table viewed from above as described by Thornton (Figure 1 of Thornton (1922) patent)

As seen in Figure 3 a table consists of bars attached to the table which hold a contraption to move the ball on the field.

Figure 4: Parts of a table soccer table (Source: Stefani et al. (2013))

The International Table Soccer Federation (ITSF) allows five different official table types in tournaments. Five additional table types are listed as recognised and used in tournaments.

All these tables use, as seen in Figure 4, bars (rods) with one linear and one rotational degree of freedom. Figures (or foosmen) are attached to the bars. Handles are used for comfort. The Frame of the table is the boundary of the field. The aim of the game is to manipulate the ball into the goal.

5 Methods

This chapter discusses several algorithms that were considered to track the state of the game as described by Bambach and Lee (2012). Bambach and Lee (2012) define the state of the game as the position of the ball and figures as well as the angle of the figures at any point in time. Therefore describing the state of the game requires two main steps:

1. Finding the figures
2. Finding the ball

The figures are attached to their bars, therefore an initial step was introduced to make figure finding faster: Finding the bars. Since the players sometimes move the table and the camera is not attached to the table (Figure 2) the relative position of the bars is expected to change slightly and rarely throughout the game. A recalibration of the bar position during the game is necessary, but need not be done every frame.

The following sections describe the video setup, the algorithms used to find bars, figures and the ball, ways to make these algorithms faster and more reliable and the hardware used to test those algorithms.

Figure 5: Example frame of a Paffrath (2016) video (can be found in Ackermann (2016) as bonzini.mp4)

5.1 Setup

Videos in the Kozoom database generally have similar camera angle and position related to the table as in Figure 5 and Figure 2. As stated before, ITSF allows five different official table types in tournaments. Five additional table types are listed as recognised and used in tournaments. The algorithms developed here work for all of those but focus on the official tables.

Kozoom usually generates Red Green Blue (RGB) video files with a resolution of 1280x720 pixel and 30 frames per second.

Please note that the video coordinates [pixels] can be transformed to physical table coordinates [cm] if needed.

Unless otherwise stated, all figures describing or referencing an algorithm in this paper will use the example frame (still of a video) Figure 5 as input. Note that the ball is visible between the second and third bar from the top.

5.2 Finding the Bars

Due to the setup, bars are expected to be horizontal in the picture. They are usually made out of well reflecting metal. Therefore they have many gradients of illumination on all three color layers that can be used to find them. The gradients are the difference of the RGB values of adjacent pixels. By subtracting the RGB values of adjacent rows these vertical gradients (Figure 6) can be extracted. By adding up the gradient values of each row, longer elements (bars) get more emphasis than shorter elements (field lines, feet of the figures, shadows) (Figure 7).

By filtering the results, field lines, shadows and other "noise" can be smoothed out (Figure 8). In this project, a two step moving average filter was used. The main problem in finding all eight bars is the near goal bar, which in most games is occluded by the frame of the table due to the angle of the camera.

The output of this very primitive algorithm is a data set with 7 to 8 high peaks representing the bars and usually 3 to 5 lower peaks representing the field lines. There are no field lines on the

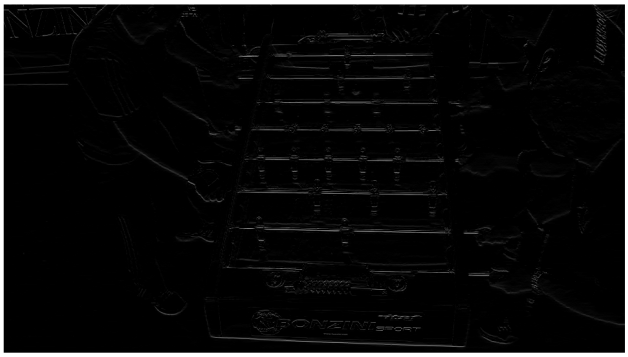


Figure 6: Vertical gradients of Figure 5

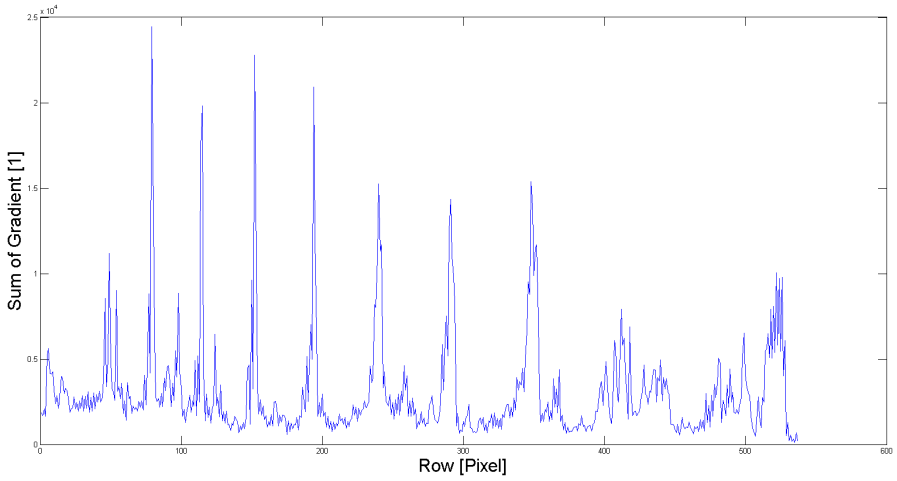


Figure 7: Sum of the pixel values of each row

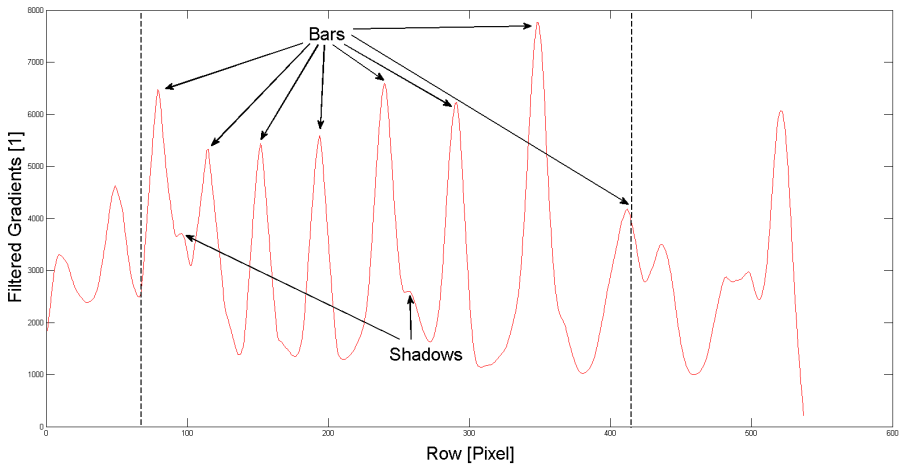


Figure 8: Sum of the pixel values of each row after filtering

table type (Bonzini) used to demonstrate the algorithm. A peak finding algorithm (e.g.: Matlab's (Natick, US) *findpeaks*) can be used to find the positions of the higher peaks while ignoring the lower peaks. The higher peaks correspond to the positions of the bars. If the position of the field is known (dashed lines in Figure 8) other strong borders like the edge of the table are easier to ignore.

The code used to calculate these steps with Matlab is displayed in listing 1.

```
differenz=bild(1:end-3,:,1)-bild(2:end-2,:,1);
bildsum=sum(differenz,2);
windowsize=ceil(0.012*groesse(2));
filtsum=filtfilt(ones(1,windowsize),windowsize,bildsum(:,1,1));
[~,pos]=findpeaks(filtsum,'NPeaks',8,'MinPeakHeight',0.4*max(filtsum))
```

Listing 1: Code to find horizontal bars in an RGB picture of a table soccer table

5.3 Finding the Figures

The same way bars are found on the table, figures can be found on the bars. Along each bar, strong horizontal gradients of illumination are only expected where a figure meets the bar, due to the linearity of the bars. Again, the output of this is a data set with distinctive peaks at the position of the figures.

Finding the figures is part of describing the state of the game, but is not relevant in any other way to tracking the ball. Therefore, it will not be discussed further in this paper.

5.4 Finding the Ball

Finding the ball is much harder than finding the bars and figures, because the ball is highly dynamic and often occluded. There are many different techniques and algorithms in image processing which are useful in this project. The following subsections will present the algorithms tested and explain how they work. Their comparative advantages and disadvantages will be discussed in chapter 7. All of these algorithms can be used to find the ball.

5.4.1 Circle Recognition

The goal of circle recognition is to find the center and radii of circles in an image. This is often done by Circular Hough Transform (CHT) (Atherton and Kerbyson (1999)). The CHT of a binary (black and white (B/W)) image is the set of all circles around all white parts of the picture. Since current computers can not interpret infinite sets, the CHT will be computed discretely.

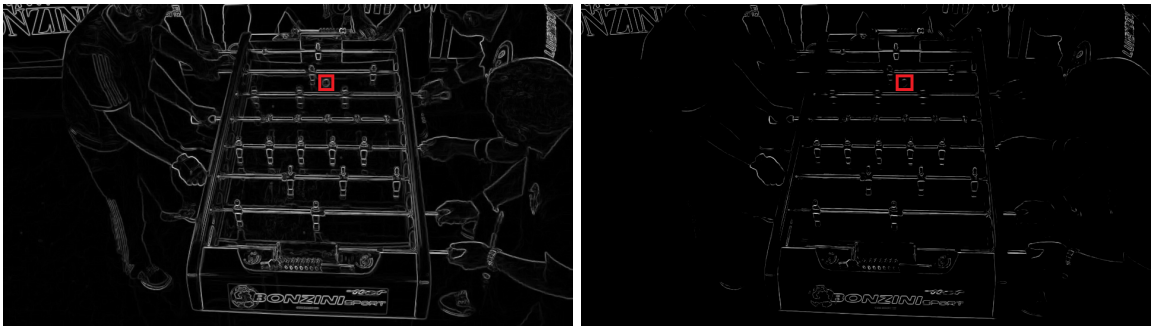


Figure 9: Output grayscale image after preprocessing and before B/W conversion with lower threshold on the left and higher threshold on the right. The ball is marked in red. Notice the ball's border disappeared on the right.

The circle will only be searched for in a user-defined radius range (i.e.: only circles with a radius within a certain range are searched for). In this subsection, modern CHT is described as a four step process:

This section will focus on Matlab's *imfindcircles* function (MathWorks Deutschland (2016a)). *imfindcircles* has two necessary parameters: An image and a circle radius or radius range. Optional parameters used in this project were *EdgeThreshold* and *Sensitivity*. The *EdgeThreshold* defines the gradient over which edge pixels are allowed to vote (Figure 9). *Sensitivity* defines the threshold over which possible circle centers are counted as actual circle centers (Figure 11 and 12).

1. Preprocessing

   The CHT is only defined on a B/W image. In this project, the input, as stated in chapter 5.1, is an RGB image which therefore needs to be converted. To convert a color image to B/W using Matlab, one goes through the following steps.

   - use the *rgb2gray* function to compute a grayscale image,
   - compute the illumination gradients,
   - use the *graythresh* function to find a threshold
   - split the pixels into black and white pixels using the gradient threshold, where pixels with gradients below/above the threshold become black/white (*im2bw*)

   The output of this process is a B/W image in which areas which had a high gradient in the initial (RGB) image appear white and those with low gradients appear black. This means that, ideally, the boundary of the ball will appear white, with its interior and the background being black. Figure 9 gives examples of outputs for different threshold values.

   For the next step to work as intended, the border of the ball needs to be white. If the gradients of the ball are not strong enough (fall under the threshold) most/all information of the ball's position will be lost in this step (Figure 9 right).

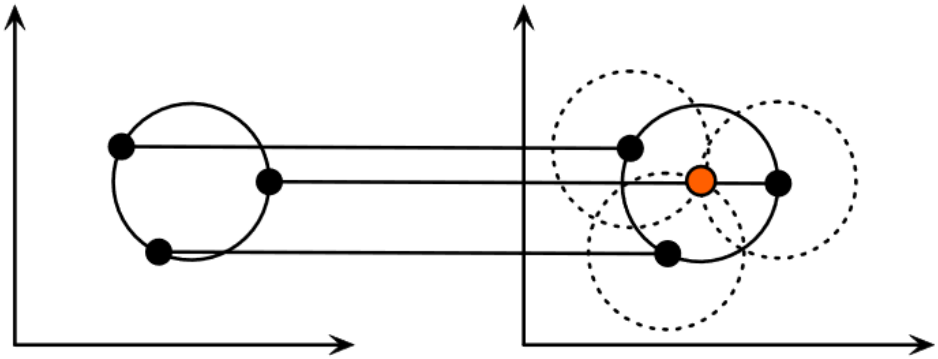


Figure 10: Three points on a circle (black, representing white pixels) vote for dashed circles. Votes accumulate at the red dot (the black circle's center). (Source: Rhody (2005))

The number of white pixels greatly influences the time this algorithm needs. Reducing the number of white pixels (by manipulating the threshold) also reduces the likelihood of finding fainter circles.

2. Accumulator Array Computation:

   The accumulator array (as used in *imfindcircles*) is a two dimensional array used to gather data on probable circle center positions.

   Every white pixel votes for all pixels in a fixed radius of itself (the vote radius)(Figure 10), which is normally the (arithmetic) mean of the minimum and maximum radius of the range defined by the user. The accumulator array is the sum of all these votes (normalized to the interval $[0, 1]$).

   When the computation is finished, each entry (or pixel) of the accumulator array will contain normalized information about how many circles built around white pixels pass through this accumulator array entry. Non-zero entries will accumulate at centers of circles i.e. a high density of high values indicates a likely location of a circle's center. Figure 11 is a visualisation of an accumulator array. More examples can be found in appendix a.

   In classical CHT, the accumulator array has three independent variables: u, v (coordinates within the image) and r (radius within the radius range). Computing the accumulator array through all possible (discrete) vote radii takes a lot of computational power. In modern CHT, the circle radius estimation step was introduced to find the actual radius of the searched circle after computing only one vote radius' CHT.

3. Center Estimation:

   Votes of points on a circle will accumulate around the center. Therefore areas of the accumulator array with a higher number of votes can be used to approximate centers of circles in the original picture. Figure 12 displays possible circle centers as grey pixels,

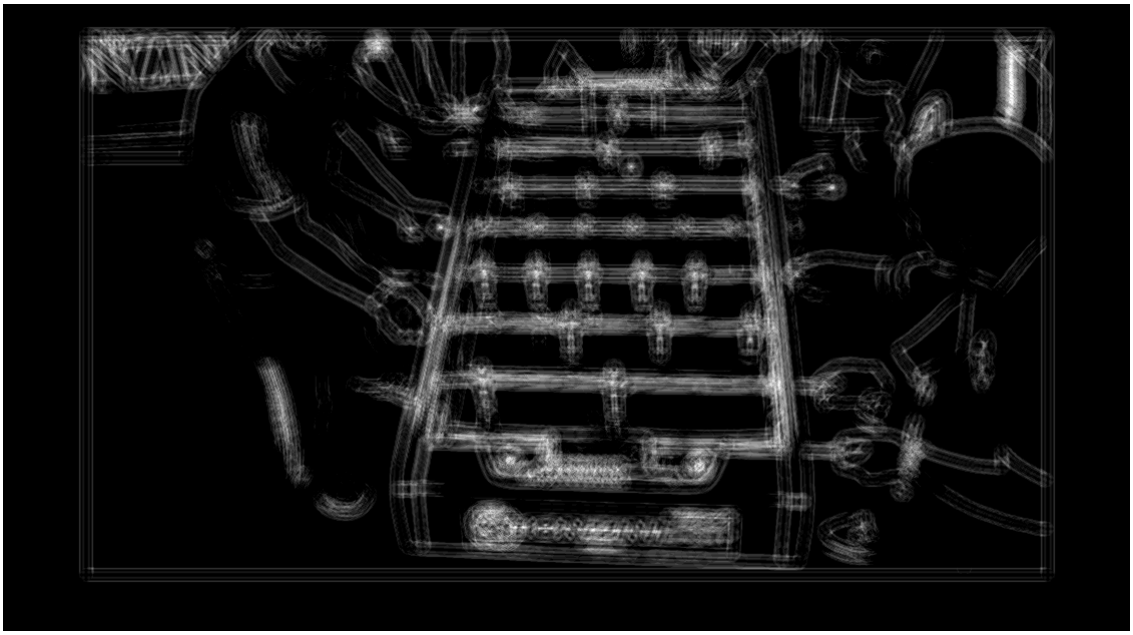


Figure 11: CHT of Figure 5, preprocessed as described in step 1

where the brightest pixels are the most likely. (Please note that the correct center might not appear due to the current resolution of the picture.)

4. Radius Estimation:

   Usually the radius of the searched circle is unknown but lies in a known interval. If it is known exactly, this step is not necessary.

   Since the accumulator array was only computed for one vote radius the original circle's radius will most of the time not be computed exactly. This step estimates the difference to the original circle's radius.

   Matlab offers two solutions to the radius estimation step:

   - "Two-Stage

     Radii are explicitly estimated utilizing the estimated circle centers along with image information. The technique is based on computing radial histograms; [...]

   - Phase-Coding

     The key idea in Phase Coding is the use of complex values in the accumulator array with the radius information encoded in the phase of the array entries. The votes cast by the edge pixels contain information not only about the possible center locations but also about the radius of the circle associated with the center location. Unlike the Two-Stage method where radius has to be estimated explicitly using radial histograms, in Phase Coding the radius can be estimated by simply decoding the phase information from the estimated center location in the accumulator array."

   (MathWorks Deutschland (2016a))

Figure 12: probable circle centers of Figure 11. For this particular setup, the center of the ball appears among the set of possible centers.

5.4.2 Pattern Recognition

Pattern recognition is the process of finding a predefined template in an object. The template has to have the same data structure as the object. Usually the data structure is an image (grayscale), but it can be anything. Other common examples are voice signals or 3D graphics (a google scholar search returns more than 250K hits in both cases). In this case, RGB images were used and the template used was an image of the ball (Figure 13).

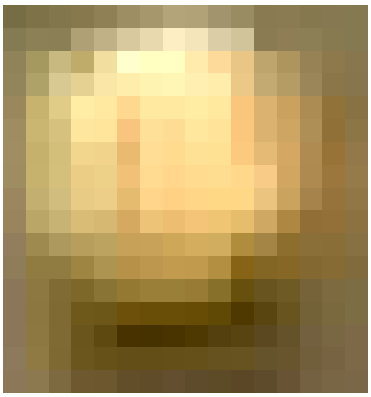


Figure 13: Possible template of the ball for Figure 5

Pattern recognition is usually a two step process:

1. Correlation Array Computation

   A correlation array displays how well the picture correlates with the template in every possible position of the template in the picture.

2. Finding Extrema

   The entry of the correlation array with the highest (or lowest) value is the most likely to be the ball. (Although this need not actually be the ball (chapter 5.4.4)).

The first step is different depending on which characteristic is most important to match. If the values of the RGB information are important, the RGB information of the template should be used. If the frequency of the object is more important, the Fourier transform of the template and image can be used as described in Wood (1990).

Since most template matching algorithms are programmed for grayscale images RGB images are usually preprocessed into grayscale images. In this project, each color layer was computed separately to not loose valuable color information.

Either way, both of the following algorithms to compute the correlation array can be used.

Least Squares (pattern matching)

Least Squares method can be traced back to Carl Friedrich Gauss (Stigler (1981)). Since then many applications were developed.

In this project, Least Squares method is used to find the closest match with the template. The image is compared with the template in every position the template can fit in. The comparison is made by summing the squared differences of the RGB pixel values of the template and the frame (Formula 1 - which was created specifically for this purpose). This offers two pieces of information: Where the ball probably is, and a correlation value that can be compared to previously calculated values to validate the finding (chapter 5.4.4).

Each entry of the correlation array γ is calculated by

$$\gamma(u,v) = \sum_{x,y}[f(x+u, y+v) - t(x,y)]^2 \tag{1}$$

where

- u and v are the coordinates within the image
- x and y are the coordinates within the template
- f is the image ($f(x,y)$ is the RGB value at x, y)
- t is the template ($t(x,y)$ is the RGB value at x, y)

Cross Correlation

Cross correlation is the sum of products of correlating indices. This only leads to relevant results if the input averages to zero, therefore it is previously decreased by its arithmetic mean. The Cross correlation value increases the more similar the image is to the template. Formula 2 is the formula Matlab uses in *normxcorr2* (MathWorks Deutschland (2016c)).

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}][t(x-u,y-v) - \bar{t}]}{\{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}]^2 \sum_{x,y}[t(x-u,y-v) - \bar{t}]^2\}^{0.5}} \quad (2)$$

where

- f and t are as above (image and template)
- $\bar{t}$ is the mean of the template $\frac{\sum t(x,y)}{\#\ pixels \in t}$
- $\bar{f}_{u,v}$ is the mean of $f(x,y)$ in the region under the template

The Least Squares method is faster than cross correlation due to fewer multiplications.

5.4.3 Least Squares (Non Pattern Matching)

The aim of Least Squares Non Pattern (LSNP) method is to quickly find areas the ball might be in. A single ball color will represent the whole ball, even though the RGB values are not all the same in an image of the ball. This way the operations needed to compute a correlation array are minimized while getting a useful result.

$$\gamma(u,v) = [f(u,v) - t]^2 \quad (3)$$

Formula 3 was developed for this step of the project and is very similar to the Pattern matching Least Squares' Formula 1, but note that t is now a single color and therefore does not have coordinates in Formula 3. The sum was removed entirely.

```
for colorfields=1:3
colordiff(:,:,colorfields)=(int16(framecroped(:,:,colorfields))...
-handles.color(1,colorfields)).^2;
end
summe=sum(colordiff,3);
```

Listing 2: Least Squares Non Pattern (LSNP) summed over the colors

Listing 2 first calculates the squared differences for each color and then sums them up for each pixel. This way, big differences will be noticed more easily.

The result will be very noisy, but by applying a filter, peaks can be made more easily detectable. A 2D moving average filter (Listing 3 and Listing 4) of windowsize small enough that it can be fully contained in the ball is useful, as it will only contain relevant information. A moving average filter builds the mean of all values in a predefined window and then moves the window to the next pixel. The values in the window can be weighted differently if wanted/needed. One stage moving average filters lead to a phase shift, but it can easily be corrected (by adding half of the windowsize to the index numbers).

```
windowsize=5;
filtered=d2filtfunction(summe,windowsize);
[ypeak, xpeak] = find(filtered==min(min(filtered(:))));
clear colordiff %needed because of changes in size
```

Listing 3: Filter and extraction of minima

```
function [corr2]=d2filtfunction(data,windowsize)
%2d moving average filter
groesse=size(data);
corr=NaN(groesse);
corr2=NaN(groesse);

%adds first windowsize values
corr(floor(windowsize/2),:)=sum(data(1:windowsize,:),1);
%subtracts first sum entry and adds next
for i=1:(groesse(1)-windowsize)
    corr(i+floor(windowsize/2),:)=corr(i+floor(windowsize/2)-1,:)...
    -data(i,:)+data(i+windowsize,:);
end

%same in second dimension
corr2(:,floor(windowsize/2))=sum(corr(:,1:windowsize),2);
for i=1:(groesse(2)-windowsize)
    corr2(:,i+floor(windowsize/2))=corr2(:,i+floor(windowsize/2)-1)...
    -corr(:,i)+corr(:,i+windowsize);
end

%getting rid of entries needed for calculation
corr2(:,1:(floor(windowsize/2)-1))=NaN;
corr2(1:(floor(windowsize/2)-1),:)=NaN;
corr2(:,end-(floor(windowsize/2)-1):end)=NaN;
corr2(end-(floor(windowsize/2)-1):end,:)=NaN;
```

Listing 4: 2D non-weighted moving average filter

5.4.4 Controlling

Controlling, in this case, refers to making sure that the ball was found using any of the algorithms mentioned above. All of the algorithms return some value that can be compared to a calibrated value.

Calibration

Since there are five tournament relevant tables and the lighting will be different depending on the time and place there has to be some type of calibration for each game (not only once in a laboratory). In table soccer there are three main states of the ball:

1. visible
2. occluded
3. hidden/not on the table

By feeding the algorithm several images of each state, an expected output value for each state can be computed. The calibration values are the mean of several values of each state. Most important will be the algorithms' output value when the ball is not on the table because this offers a upper bound on the occluded value.

Another problem here is when to calibrate. The options for this are:

- Calibrating while the game is going, and therefore maybe missing some frames at the beginning of the game,
- Doing it before the game, and maybe missing out on accuracy at the end of the tournament due to different lighting.
- Not doing it live

Calibrating the algorithm during the game is difficult because another source of validation of the ball's current state or enough data to be able to distinguish the states safely is needed.

Controlling

Once these calibration values are computed, they can be compared to the values from the accumulator/correlation array to control the validity of the match. By doing so, the ball can be defined as found, occluded or not found, which is important to get rid of wrong matches.

By comparing results of color based algorithms (pattern matching) and shape based algorithms (circle recognition) the likelihood of finding the ball correctly can be increased.

In addition, circle recognition as described in chapter 5.4.1 can be used again in the estimated area. In smaller areas there are fewer gradients therefore weaker gradients can become comparatively "strong" as defined by the *graythresh* function. This can lead to different results.

Additional algorithms were developed to control the validity of the match. One algorithm that controls if certain pixels in the matched area (chapter 5.5) lie in an RGB interval around the ball color proved not to be effective at the moment, but looks promising for validating the balls position after some improvements.

5.5 Reducing the Search Area

The setup described in chapter 5.1 contains the table in the center part of the image. There is no point in searching for the ball outside of the area of the image that contains the field. The search area can be reduced to the area of the image containing the field.

In most cases, the ball will not move much between frames due to the playstyle of professional players. Therefore the ball can be searched in the area around the position of the ball in the last frame. This area will be referred to as 'cropped'. This saves a lot of time. If the ball is not found somewhere around the last position of the ball, the whole field can still be searched.

If a fast and unreliable method found a possible position of the ball, it can be verified by searching the area directly around the ball (a square with side length of the ball's diameter around the ball) with a different, more reliable algorithm. This area will be referred to as 'match'.

5.6 Generating Statistics

The most interesting statistic for Kozoom is the percentage of successful passes. A successful pass is defined by the ball being controlled on the middle bar, then passed directly to the attacker bar while only passing through the area of the opposing middle bar, and then being controlled at the attacker bar. (The ball is generally not passed from the defenders to the middle bar.)

"Controlled" was defined as the ball being within the reach of a "player figure" (as defined for International Table Soccer Federation (2016) section 24) for at least one second.

The biggest challenge was to determine when the ball is within reach of a figure. Since the figures are attached to the bars the reach of the figures correlates to the position of the bars. Determining the position of the bars in the image was covered in chapter 5.2. Because of the camera angle, the position of the bars, as they appear in the image, does not necessarily correspond to their vertical projection onto the table. It is assumed that all bars are off by the same distance. It is known that (on the table) the field's middle line is exactly in between the two middle bars. The bars' vertical projection onto the field can be calculated by subtracting the distance between the far middle bar and the middle line (b in Figure 14) and adding half of the distance between both middle bars (a in Figure 14) to each bar's position in the image.

This is not strictly accurate because the calculation relies on a parallel projection where the camera creates a focused perspective, but this is good enough for this use case. The distance at which the ball is within reach of a figure is known for each table (as it can be measured). For tables without a middle line no automatic solution for adjusting the bars has been developed. Results without adjustment are still usable, but not as accurate as with adjustment. If the adjustment seems necessary the defender bars position related to the edge of the field can be used instead.

Once the bar within reach of the ball is known defining if the ball is controlled is just a matter of counting frames. If the number of frames in which the ball is within reach of one bar reaches

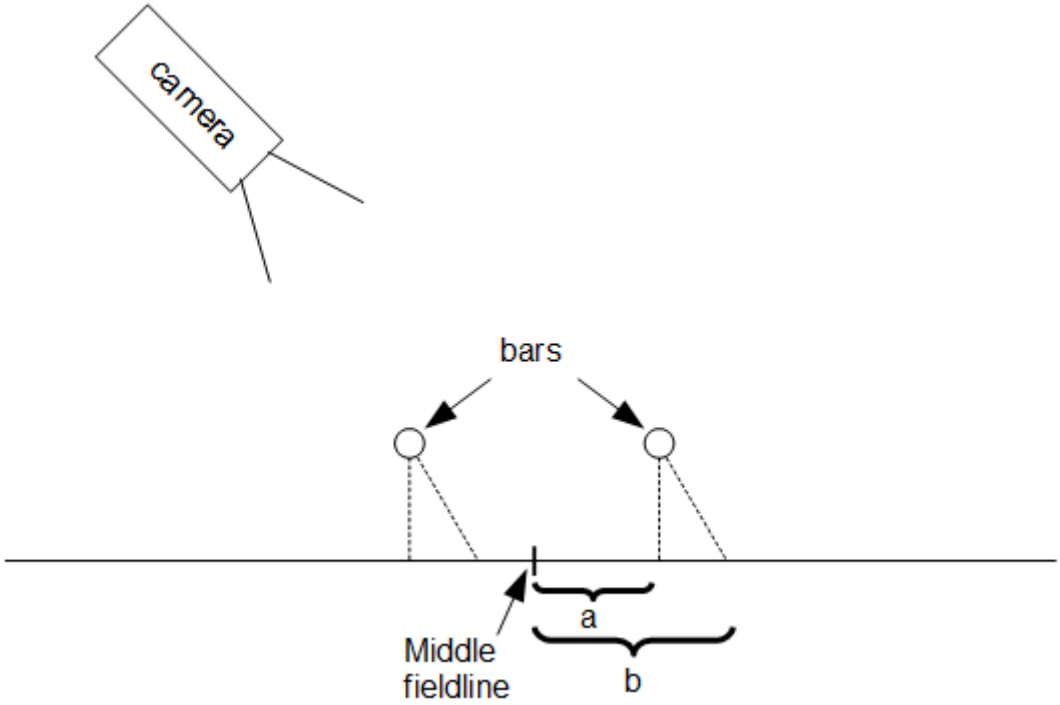


Figure 14: vertical and angled parallel projection of the middle bars on the field; a indicates the distance of the vertical projection and the middle line. b indicates the distance of the angles projection and the middle line.

the frame rate of the video the ball is counted as controlled.

If these conditions are met at the beginning and the end of the pass and the ball was not found outside the area of the opposing middle line in between, a successful pass was executed. If the ball is controlled on the middle bar a pass is initiated. If either the ball gets into a different area or it is not controlled at the attacker bar afterwards the pass is counted as unsuccessful.

5.7 Environment

The algorithms were developed and tested using the following environment:

- Input: RGB Video file in a data format supported by Matlab (http://de.mathworks.com/help/matlab/ref/videoreader.html)
- Output: Matrix with position and state of the ball (visible/occluded/not visible)
- Operating system: Windows 10 Pro 64 bit
- Processor: Intel(R) Core(TM) i5-4690K CPU @ 3.50GHz 3.50GHz
- Matlab R2013a

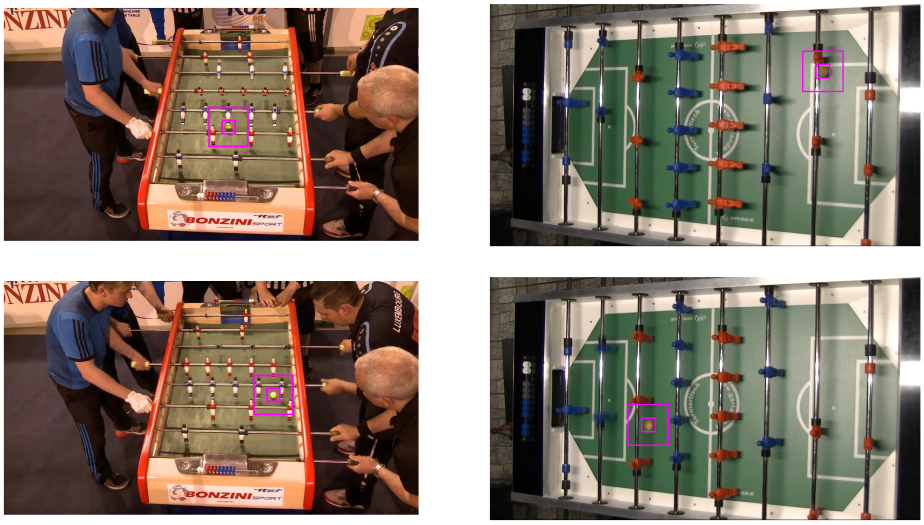


Figure 15: Frames tested in chapter 6. Magenta lines represent cropped and match area; Bonzini Table on left, Garlando Table on the right; occluded ball on the top, visible ball on the bottom (Source: Paffrath (2016))

6 Measurement Results

In this section the results of several algorithms applied to single images (Figure 15) are presented. The computation time as well as if the ball was found were observed. Measurements are computed on the area of the whole field, the cropped area, and the match area (as described in chapter 5.5).

Measurements are computed on two different table/resolution combinations:

- Bonzini (France) table with 960x540 pixels
- Garlando (Italy) table with 1280x720 pixels

All inputs used were videos from the Kozoom database (Paffrath (2016)). The camera used as well as the exact format of the video (.mp4 files can have various encodings) are not known. The videos that were used for testing can also be found at Ackermann (2016).

Measurements are computed on the environment described in chapter 5.7.

The times were measured using Matlab's *tic* and *toc*. The timer was started at an RGB image state and was stopped when the output (coordinates u and v of the balls position) was calculated. All times that differed too much from the expected results were measured four times to confirm their accuracy (Appendix b).

After it became clear that the calculation time for most of the algorithms was too long, only LSNP and CHT were examined further. Accuracy and validity were measured by hand for certain cases. CHT was only tested in the match area.

Two video outputs demonstrate the results of the algorithm and offer the opportunity to control them visually. The validity video (Ackermann (2016), Validity test.mp4) is the current standard output of the application developed during this project. The accuracy video (Ackermann (2016), Accuracy test.mp4) is a visual output of the ten most likely centers of the ball calculated by the LSNP algorithm.

Validity video

- A blue rectangle indicates that this is the most likely position for the ball according to the LSNP algorithm.
- A red rectangle indicates that *imfindcircles* found one circle in this area.
- A cyan rectangle indicates that both of the above are true.
- A black rectangle indicates that only this area will be searched (cropped area).
- A red rectangle around the whole field indicates that the ball's acceleration is bigger than humanly possible (this was roughly tested at the Garlando world series 2016 in Salzburg). These cases will be ignored.
- A yellow rectangle as well as a red circle indicate that the algorithm described in chapter 5.4.4 sees this location as the most likely.

Accuracy video

The ten most likely positions of the ball's center of every frame according to the LSNP method are indicated. Around every calculated position a square of five rows and columns will be ignored to show less likely positions too. This was originally designed for (and still used for) finding more options to verify by CHT. Note that a fast moving ball appears blurry.

The accuracy of CHT was mainly observed while defining the template/ball color/ball radius, as it is displayed in the process.

The found/not found column of Table 1 only indicates whether the center of the ball, as defined by the algorithm, is somewhere within the ball.

6.1 Circle Recognition

In Circle Recognition the computation time is dependent on the number of pixels with high gradients (chapter 5.4.1). This will decrease with the size of the image. When searching in a

smaller area, different parts of the image can appear as relevant pixels due to relatively stronger gradients being outside the search area. The computation time will increase with every figure in the image.

The algorithms tested were
imfindcircles(image,[4,10],'Sensitivity',0.9) for Bonzini
imfindcircles(image,[7,20],'Sensitivity',0.9) for Garlando.

as well as a fixed edge threshold which is the *EdgeThreshold* used by default by *imfindcircles* on the visible match area.

960x540 Bonzini		
occluded, imfindcircles fixed edge threshold		
field	0.094108 seconds	not found
cropped	0.014597 seconds	not found
match	0.007458 seconds	not found
occluded, imfindcircles default		
field	0.054349 seconds	not found
cropped	0.009300 seconds	not found
match	0.006777 seconds	not found
visible, imfindcircles fixed edge threshold		
field	0.109658 seconds	not found
cropped	0.018720 seconds	not found
match	0.008710 seconds	not found
visible, imfindcircles default		
field	0.055404 seconds	not found
cropped	0.011696 seconds	not found
match	0.007188 seconds	not found

1280x720 Garlando		
occluded, imfindcircles fixed edge threshold		
field	0.479957 seconds	found
cropped	0.020184 seconds	found
match	0.009059 seconds	found
occluded, imfindcircles default		
field	0.324257 seconds	not found
cropped	0.015164 seconds	not found
match	0.008029 seconds	found
visible, imfindcircles fixed edge threshold		
field	1.019829 seconds	not found
cropped	0.055463 seconds	found
match	0.010018 seconds	found
visible, imfindcircles default		
field	0.324022 seconds	not found
cropped	0.014795 seconds	not found
match	0.007953 seconds	found

Table 1: Measurement results of Matlab's *imfindcircles* function for different tables, ball visibility states, edge threshold and search areas

If the ball is slightly occluded sometimes it can still be found.

No convincing explanation has been found for why the time difference in the visible match area between default and fixed edge threshold exists (if any difference exists, fixed edge threshold should be slightly faster for this case only).

6.2 Other Methods

In all methods other than circle recognition, the computation time does not depend on the visibility of the ball.

960x540 Bonzini		
visible, Least Square non pattern matching		
field	0.009473 seconds	not found
cropped	0.006416 seconds	found
match	0.005985 seconds	found
visible, color cross correlation		
field	0.055212 seconds	not found
cropped	0.011026 seconds	found
match	0.005567 seconds	found

1280x720 Garlando		
visible, Least Square non pattern matching		
field	0.025330 seconds	found
cropped	0.010125 seconds	found
match	0.006310 seconds	found
visible, color cross correlation		
field	0.423136 seconds	found
cropped	0.165027 seconds	found
match	0.008262 seconds	found

Table 2: Measurement results of color based methods

7 Evaluation

Every algorithm tested can not fulfil reliably valid real-time analysis of 1280x720 resolution videos with a frame-rate of 30 frames per second (fps) on its own.

7.1 Calculation Time

As seen in Table 1 and Table 2, only the LSNP method is fast enough to compute the whole field 30 times per second (having a calculation time per frame < 0.0333 s = 1/30 s). This is true for both resolutions tested. In this project, speed is not the most important aspect, but it is nice to have. A calculation time bigger then 1/30 s would lead to the generation of statistics taking longer then the game. The highest calculation time for the whole field in one frame was 1.019829 s which would lead to a calculation time of 30 times the original video (this is unacceptable).

7.1.1 Influencing factors

Higher resolutions directly lead to more computations being needed. Not all of the algorithms react the same way. LSNP methods calculation time grows linearly with the number of pixels. Pattern matching calculation time grows quadratically with the number of pixels. Circle recognition grows with the resolution because of more pixels being allowed to vote.

Illumination does not influence the time needed to compute pattern matching algorithms or LSNP method. It greatly influences CHT due to influencing the number of pixels allowed to vote. Both too strong and too shallow illumination will lead to shorter computation times. This is because both too shallow and too high illumination will minimise the variation of pixel colours (extremes of this are a completely black image or a blinded camera). Both will lead to few pixels with high gradients.

Absence of the ball leads to not finding the ball. Longer calculation times in the following frame result because the ball can not be searched around a known position. If the ball was not found in the previous frame all algorithms need longer calculation time due to searching the whole field.

7.2 Validity

Finding the ball proved to be as hard as expected due to the ball not being fully visible at most times and other objects resembling the ball being present on the field.

In this project, validity is a check whether what was found is actually the ball or not.

CHT's capability of recognizing circles with a radius below 5 pixels is very limited. This is one of the reasons why the tests on Bonzini were all negative (Table 1, left). Matlab's *imfindcircles* has a built in validation by only recognizing results over a threshold (*'Sensitivity'*) as valid results. A well chosen sensitivity can validate findings. Table 2 shows that the findings of the LSNP method is viable. It does not check itself. The way this check is done in this project is by checking if the area the ball is estimated in contains a round object via CHT.

7.2.1 Influencing factors

The color contrast between the ball and the field influences CHT. If the contrast between any combination of bars, field and figures is much stronger than the contrast between ball and field the ball will not be found by CHT due to the edge of the ball not being recognized as an edge.

If the color of the ball is similar to the color of the field or figures, template matching algorithms (or LSNP method) can mistake the figures or the field for the ball. This is especially true for Bonzini tables, as the pants of the figures resemble the ball in size and color, and the field's color is very similar to the ball's.

If the ball was not found, the software should not validate the result because it can not verify the position via circle detection.

Illumination and surface structure can influence the colors received by the camera. A green field can look brown if the lighting is red. The surface structure can also cause reflections which produce an uneven field color.

Fields with colored corners (Garlando) cause problems for pattern matching algorithms due to the template containing background colors which are usually the color of the field.

All these problems cause the correlation array to be different. If the difference is too big the position containing the ball (even when it is the most likely) can be counted as occluded or not found due to the correlation array value.

The resolution influences the validity of both color based and algorithms and *imfindcircles*. Color based algorithms can either use bigger templates or a bigger windowsize of their filter. CHT works more reliable on bigger circles.

7.3 Accuracy

The only tested algorithm with a theoretical accuracy higher then one pixel is the CHT.

Both the LSNP method as well as CHT can not define the balls center reliably. Both algorithms will define the center somewhere on the ball, which is good enough for this Project. CHT usually gets results closer to the center then LSNP.

7.3.1 Influencing factors

Higher resolution increases the real life accuracy of all algorithms. This only means that the same real life distance can be divided into more pixels. This is nearly irrelevant in this project.

A higher resolution will, as described in chapter 7.2, increase the quality of CHT's output.

Colors of the ball, figures, table and bars influence the edge threshold for CHT.

8 Conclusion

The ball's path as well as corresponding probabilities of the ball's position was extracted out of a video. This data can be used for further processing. The ball's path in a video can usually be calculated in less time than the video is long.

The hardware environment described in chapter 5.7 is not fast enough calculate every frame in 1/30 of a second, but will average over 30 frames per second. This is achieved by searching for the ball in a cropped image as defined in chapter 5.5 using the Least Squares Non Pattern method described in chapter 5.4.3 for possible ball positions and verifying them by circle recognition (chapter 5.4.1). This combination of algorithms meets Kozoom's requirements.

A resolution of 1280x720 is expected by viewers. This resolution is also important for circle recognition, because in lower resolutions the ball's outline can not be detected as circular. This is crucial to verify the ball's position.

Most of the time, the ball is slow enough to find it near it's position in the previous frame. The ball can travel up to 49,4 cm in one frame (which corresponds to 53,38 km/h (KickerProfis (2016))).

The ball is more or less hard to track depending on the table (hardest on Bonzini). Sometimes the ball can be seen by a human, but the software does not find it. In most such cases, the software also realizes that the ball was not found.

Bibliography

Ackermann, I. (2016). Tischfussball videos. https://www.dropbox.com/sh/5c8x84hg0s4xr3m/AADmWai2PYjwe-SbLNB-foPqa?dl=0 (accessed September 18, 2016).

Alsalihi, A., K. Najjar, B. Van Scoy, and J. Zifer (2011). Automated foosball table. https://uakron.edu/dotAsset/1e2fb3d4-8c59-475e-9473-ed98b2504f17.pdf (accessed September 10, 2016).

Atherton, T. J. and D. J. Kerbyson (1999). Size invariant circle detection. *Image and Vision computing 17*(11), 795–803.

Bambach, S. and S. Lee (2012). Real-time foosball game state tracking. http://iu.svenbambach.de/foosball-tracking.pdf (accessed September 11,2016).

Baodong, Y. (2014). Hawkeye technology using tennis match. COMPUTER MODELLING & NEW TECHNOLOGIES 2014 18(12C) 400-402.

Carrer, X. (2016). Legal notice table soccer. http://www.kozoom.com/en/table-soccer/page/legal-notice.html (accessed September 6, 2016).

Grauman, K. and B. Leibe (2011). Visual object recognition. *Synthesis lectures on artificial intelligence and machine learning 5*(2), 1–181.

Hough, P. (1962, December 18). Method and means for recognizing complex patterns. US Patent 3,069,654.

International Table Soccer Federation (2016). Itsf rule book. http://www.table-soccer.org/rules/documents/2016_Rulebook.pdf (accessed September 13, 2016).

KickerProfis (2016). Speedkicker vermietung - kicker.profis. http://www.kickerprofis.de/vermietung/speedkicker (accessed May 23, 2016).

MathWorks Deutschland (2016a). Find circles using circular hough transform - matlab imfindcircles - mathworks deutschland. http://de.mathworks.com/help/images/ref/imfindcircles.html (accessed May 20, 2016).

MathWorks Deutschland (2016b). Hough transform - matlab hough - mathworks deutschland. http://de.mathworks.com/help/images/ref/hough.html (accessed September 16, 2016).

MathWorks Deutschland (2016c). Normalized 2-d cross-correlation - matlab normxcorr2 - mathworks deutschland.
http://de.mathworks.com/help/images/ref/normxcorr2.html (accessed May 20, 2016).

Nebel, B., T. Weigel, and J. Koschikowski (2005). Tischfussball, Hockey oder dergleichen und Verfahren zur automatischen Ansteuerung der an Stangen angeordneten Spielfiguren eines Tischspielgeräts für Fussball-, Hockey- oder dergleichen. *Patent DE 102 12 475*.

Paffrath, M. (2016). Kozoom Multimedia
http://www.kozoom.com/ (accessed May 22, 2016).

Rhody, H. (2005). Lecture 10: Hough circle transform. *Chester F. Carlson Center for Imaging Science, Rochester Institute of Technology*.

Stefani, J., A. Herpy, B. Jaeger, K. Haydon, and D. Hamel (2013). Automated foosball table.
http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1237&context=mesp (accessed June 10, 2016).

Stigler, S. M. (1981). Gauss and the invention of least squares. *The Annals of Statistics*, 465–474.

Thornton, H. S. (1922). GB205991 (A) - Apparatus for playing a game of table football. GB Patent, London: KINGS PATENT AGENCY LIMITED.

Weigel, T. (2005). KiRo - A table soccer robot ready for the market. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 4266–4271. IEEE.

Weigel, T. and B. Nebel (2002). KiRo – An autonomous table soccer player. In *Robot Soccer World Cup*, pp. 384–392. Springer.

Wood, E. J. (1990). Applying fourier and associated transforms to pattern characterization in textiles. *Textile Research Journal 60*(4), 212–220.

List of Figures

List of Tables

List of Abbreviations

B/W	Black and White
CHT	Circular Hough Transformation
cm	centimetre
fps	frames per second
HZ	Hertz= 1/second
ITSF	International Table Soccer Federation
KiRo	Kicker Roboter
LSNP	Least Squares Non Pattern
m/s	meter per second
RGB	Red Green Blue
US	United States of America

9 Appendix

9.1 Appendix a

Further illustrations of the CHT of simple objects:

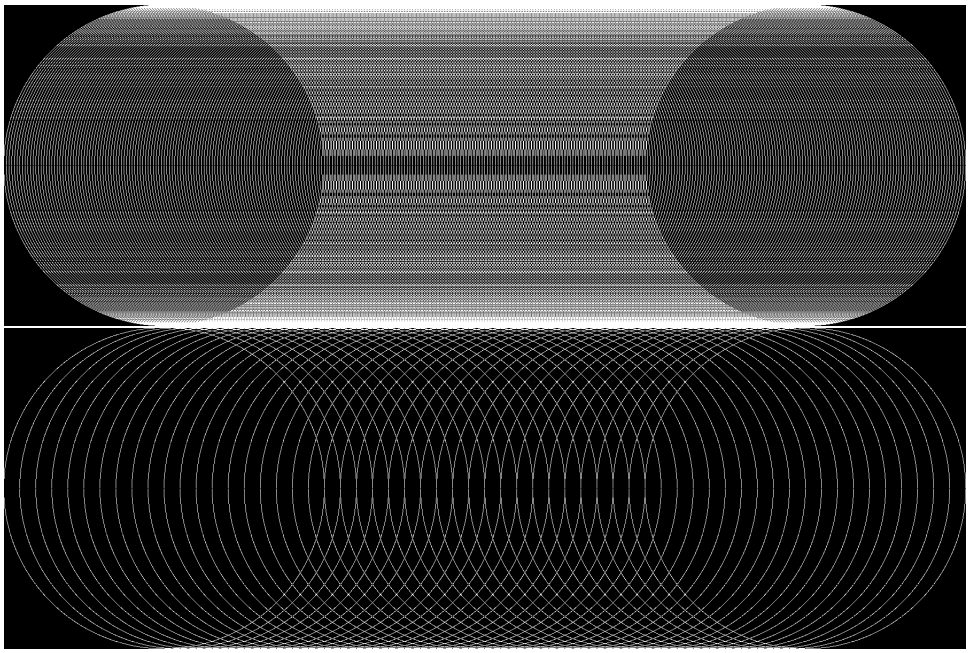


Figure 16: CHT of a line with different circle density for easier recognition

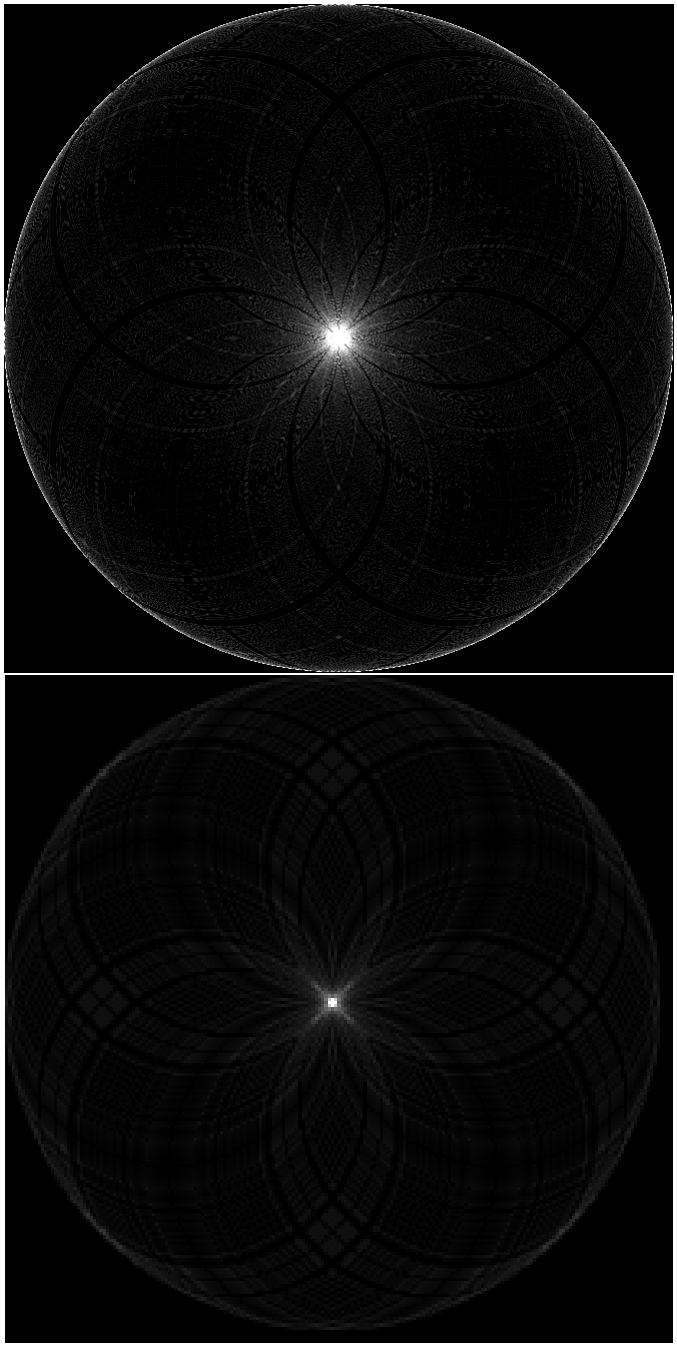


Figure 17: CHT of a circle with different circle densities

9.2 Appendix b

inconsistency in measuring times

```
>> tic
[center, radii]=imfindcircles(framematched,[10 21]);
toc
Elapsed time is 0.089624 seconds.
>> tic
[center, radii]=imfindcircles(framematched,[10 21]);
toc
Elapsed time is 0.010557 seconds.
>> tic
[center, radii]=imfindcircles(framematched,[10 21]);
toc
Elapsed time is 0.008218 seconds.
>> tic
[center, radii]=imfindcircles(framematched,[10 21]);
toc
Elapsed time is 0.008212 seconds.
```

Figure 18: Inconsistency in the same time measured four times. Note that the first measurement is about nine times as long as the others