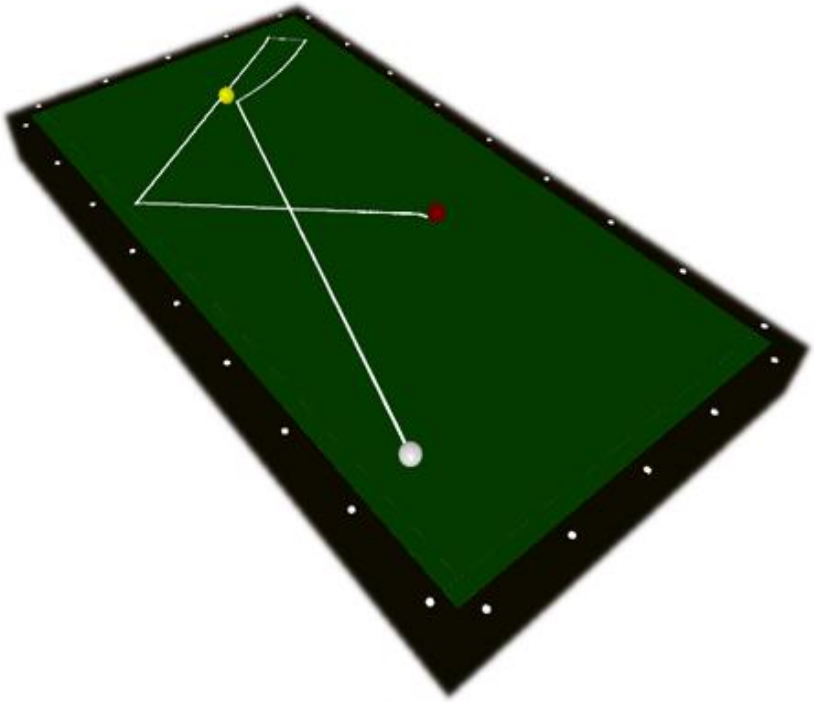
A Virtual Billiard Assistant

Joris van Balen

Student Number: 9810986

Masters Thesis



Supervisors/Graduation Committee:

dr. M. Poel
ir. H. van Welbergen
dr. Z.M. Ruttkay
prof.dr.ir. A. Nijholt

September 2009

Abstract

This thesis describes the development of a virtual billiard assistant (VBA), that should ultimately be able to replace a qualified human trainer. The scope of the project is the tactical part of three cushion billiards, where players have to solve a ball configuration, and play it accordingly, in order to score a point.

The VBA proposed in this thesis has two main tasks:

1. Find a best solution for any possible ball configuration
2. Present this solution to the user

A solution is considered best when a human trainer would recommend the same solution in the same situation.

When a user is exercising, it can ask the VBA to solve a ball configuration whenever he desires. The ball configuration is retrieved using a webcam and image processing techniques, and presented to the VBA. The VBA then systematically tries out a vast amount of shot parameter combinations, and detects which are successful. To achieve this it makes use of a physical model, which is implemented in such a way that it can accurately simulate a great amount of shots in limited time. This will typically result in a set of solutions, which are first clustered and then sorted. The top solution of this sorted set should be the same solution as a human expert trainer would recommend. The resulting solutions are presented to the user via a computer screen, using a 3d model of a billiards table, with a white line indicating the path along which the cue ball should travel.

It is shown that the system is accurate, and reasonably fast. Feasible solutions are found for almost all possible ball configurations. The method of interaction enables the user to move freely, and use the system as he or she desires. A user test was performed, indicating that the developed VBA is trusted by users, and would be a welcome innovation in billiards training.

Index

1. Introduction

Billiards is a game that has appealed to many over the last centuries. It's precise origin is unknown, but it is believed that it originated from home table versions of golf and croquet. Several accounts of famous historical figures can be found being fascinated by the game, ranging from Shakespeare to Blaise Pascal. It is even believed [1] that Mozart's permanent state of bankruptcy can, for a great part, be accredited to his love for the game, combined with the vast amount of money people used to play for at that time in Vienna. On the one hand one can argue that this was a good thing, urging Mozart to share his music to support his other addiction, on the other hand he probably would have benefited from some kind of personal billiard trainer.

Three cushion billiards[1] is a type of billiard game that is perceived as complex and difficult to master. Beginning billiard players are advised to start off with an easier form of billiards called libre, to gain basic insight, precision and technique. When players want to start learning the three cushion game, this normally happens by explanation and demonstration. The ambitious player can either watch the more skilled play their game, train for themselves or be trained by a coach. Although the latter presumably would be the best option, it is also the most expensive one.

An interesting topic in the field of Human Media Interaction is that of digital assistants. The aim of this project is to explore the possibilities for a Virtual Billiard Assistant (VBA), that should assist players wanting to master the game of three cushion, and ultimately be able to replace a human billiards trainer.

This chapter first discusses how a VBA should work, and for whom it is meant. Next the scope of the project will be defined by stating the research questions and requirements in section 1.2. Section 1.3 will gave an overview of related work and literature. Section 1.4 will explain a personal motivation for the project, and section 1.5 will give an overview of the remainder of this document.

1.1 Defining The Virtual Billiard Assistant

The ultimate goal of this project would be a complete Virtual Billiard Assistant (VBA), able to assist average three cushion players in improving their tactical insights in the game, as good as or even better than a human billiard trainer would. This goal obviously is not reachable within the limited time span of the project. In this section is presented nevertheless, what an ultimate VBA should look like. This is done by first describing a typical use case scenario, followed by a more detailed description and explanation of the specifications of the VBA. Finally the term user is specified in the scope of the project.

1.1.1 Typical scenario

Player John is an average billiards player, specialized in pool, but desiring to become a skilled three cushion player. He has mastered all the technical skills needed to aim a ball and play it fluently with the desired amount of spin applied, but he has no idea about how to solve

[1] See appendix A for an explanation of the game of three cushion billiards, and appendix B for an overview of billiard terms used throughout the project.

a ball configuration, in a manner that provides him the best odds for success. In other words, John lacks the tactical skill needed for successful three cushion billiard play.

Although he wants to do something about that, he is reluctant to spend money on a personal trainer, and does not want to bother other three cushion players too much. He knows that probably the best option for learning how to solve typical ball configurations, is to watch others play them. This has the downside that, while watching others play, John himself cannot play, and has to remember the different decisions the players make. Moreover, he will not be sure whether these decisions are the right ones, not knowing about the level of expertise of the players he is observing.

What John wants, is a Virtual Billiard Assistant. Whenever John is exercising the game of three cushion, the VBA is there to decide how each ball configuration should be solved. During the first sessions, John might just place the balls randomly on the table, and ask the VBA how to solve the shot. Or he might remember a ball configuration he wasn't able to solve successfully in last night's match, and feed it to the VBA.

After a while John becomes more skilled, and will be able to solve most configurations himself. Now when he is exercising he tries to figure out most configurations on his own, but every once in a while a configuration comes along which leaves him clueless. At that point John can address the VBA again.

Another task the VBA could carry out to assist John occurs when John misses a ball during his exercise play, but does not know what went wrong. The VBA could now provide feedback about how and why the shot carried out by John does not match the ideal shot as calculated and recommended by the VBA.

Tracking the player's skills could also be a desirable option. When the VBA is able to analyze what went wrong during a shot carried out by the player, it could accumulate this data into a profile. It would then be able to indicate shortcomings to the player, possibly advising some practice shots. The VBA might for example notice, that the player has difficulties playing draw shots. The VBA then provides some tips on draw shots, and provide exercise shots, or pick similar ball configurations from the player profile that have had a low success rate.

1.1.2 VBA specifications

In order to accomplish a scenario as described in section 1.2, the VBA should go through several steps every time the balls lay still on the table. The process consists of four main components:

1. acquire current ball configuration
2. decide how the ball configuration should be solved
3. provide advice to the player on how to solve the configuration
4. when asked for, provide feedback explaining why the last shot was not carried out (un-) successfully by the player

When balls are moving, the VBA should:

5. track the movement of the balls
6. resolve the shot parameters (aiming angle, effect, velocity) applied by the player

7. compare the shot actually being carried out by the player to the shot the VBA advised, and, if different, deduce why
8. gather data on played shots in a player profile

It is easy to see that steps 4..8 are very complicated, as (near) real-time analysis of the moving balls is required. They were not realized within the project's time span, but are recommended as future work. This thesis focuses on the first 3 main points. The aim of this project is a system that is able to recommend a solution to a player for any given ball configuration present on a billiards table as depicted in figure 1.1.

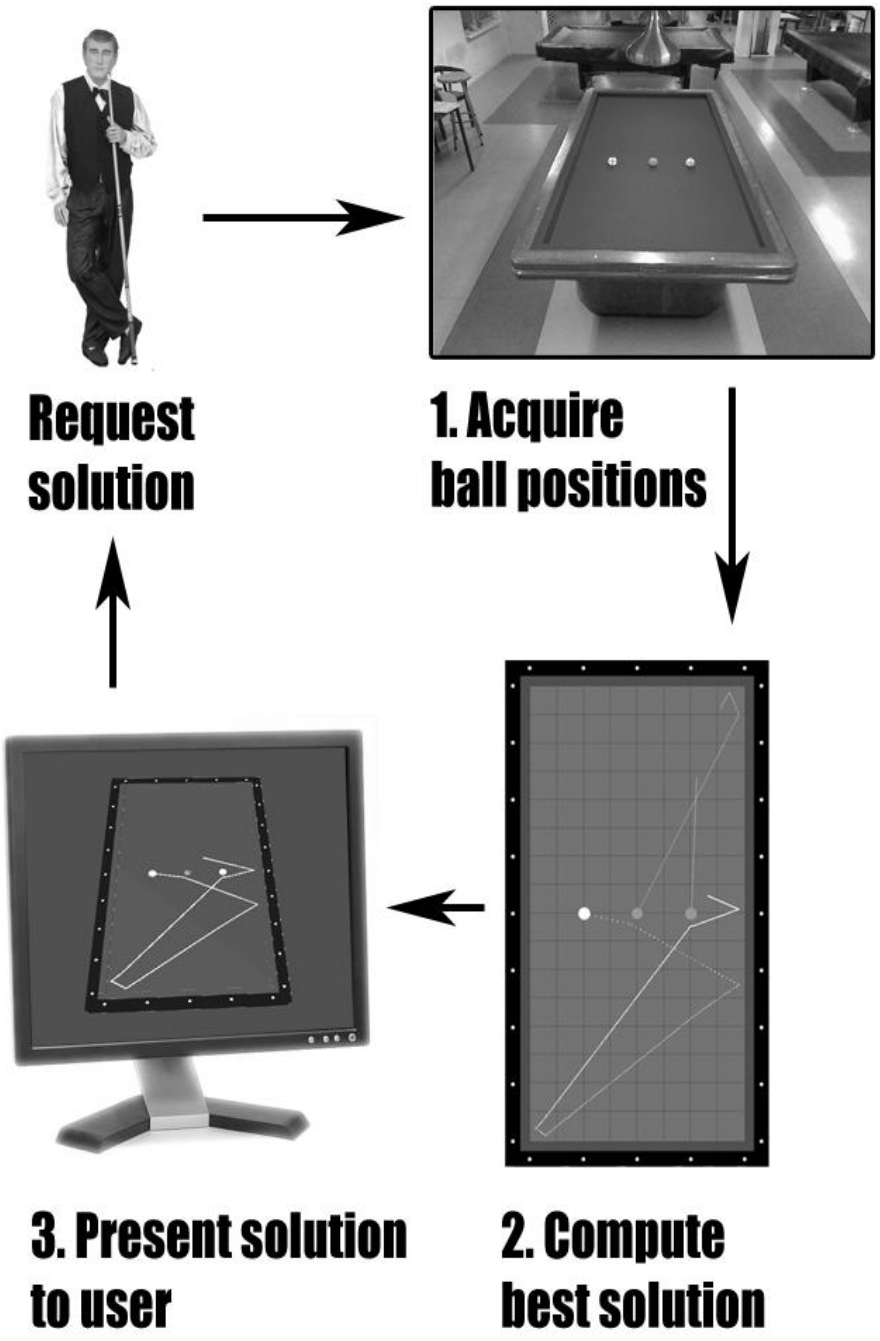


Figure 1.1: The three main tasks of the VBA

1.1.3 Target user group

Throughout the project, the person addressed as the user is the one that will be enjoying the VBA's services. By definition, the user is interested in improving his[2] billiards skills. However, different players have a different level of expertise to start with, and may desire different types of assistance/advice from a real life trainer or VBA.

Billiard skills can be split up in tactical and technical skills. By tactical skills we simply mean the insight in how to solve a ball configuration, giving the best odds for success. Technical skills consider the actual physical process of being able to carry out a solution by the player. This includes things as body posture, fluid striking technique and aiming accuracy.

Although an ultimate VBA should be able to assist the user on improving his technical skills, it will be a very complicated task, as fine details are to be detected and considered. How a VBA could assist in improving technical skills of a player is discussed very briefly in chapter 6 as possible future work. Solving ball configurations, the tactical part, is the task of the VBA this project focuses on.

Therefore the range of users that the VBA is designed for is limited to players that have a decent amount of technical skill, but lack (a full) understanding of the tactical part of the game. Typically this will be players that already have experience in other types of billiards, or players that play three cushion billiards already on an intermediate level, but wish to improve their tactical insight.

Players lacking technical skills are not expected to be able to reproduce a shot within a reasonable margin of error. On the other hand, expert three cushion players probably will not benefit from a VBA, as long as it is not incredibly fine grained and accurate.

During a match of three cushion billiards, players annotate their scores on a scoreboard. Not only are their scores annotated, also the amount of turns taken. By dividing the amount of points by the amount of turns, a player's average (or moyenne) is calculated. These averages are collected for example during the period of a year, and are a fairly solid indication of a player's level of skill. The average of the players the VBA is designed will typically be in the range of 0.25 to 0.5 points per turn. This may seem a small interval, but looking at the averages of three cushion players of the Twente District [3], most of the players fall into this category. It is not unthinkable however that players outside this range could also benefit from the services of a VBA.

The averages are also an indication of the difficulty of three cushion billiards, as an average of 0.5 implies that a player only solves and carries out successfully 1 out of 3 ball configurations, as every turn includes a miss. Players with an average of 1 are already regionally considered top players, whereas the absolute world's best players typically have an average between 2 and 3 (albeit on a match size table).

[2] The user will be addressed as male throughout the project. Obviously a user could also be female.
[3] Competition website KNBB Twente: http://competitie.knbb.nl/overview.php?District=108

1.2 Research Questions

In section 1.1 the scope for this project was established. A user should, for any possible ball configuration encountered, have the possibility to ask the VBA for advice on how to play the shot in such a way that it will result in a successful carombole. In order to develop a VBA as depicted in figure 1.1, focusing on the tactical part of three cushion play, two main research questions are explored:

R.Q. 1: How should the VBA calculate a solution to recommend to a player for any given ball configuration?

R.Q. 2: How should the VBA interact with a user?

Several general requirements are posed, that the VBA should meet:

Req. 1.1: The VBA should be able to advice the player as fast as possible. The time between a request from the user and the presentation of the recommended solution should not be perceived as too long.

Req. 1.2: The VBA should be able to give an advice for every possible ball configuration it is presented.

Req. 1.3: The VBA should be little obtrusive, and enable the player to move freely, serving as tool, that a player can use whenever he wants.

Req. 1.4: The VBA should run robustly within the testing environment.

Req. 1.5: The user should trust the VBA.

Several aspects that could be explored are left out of the project. The most important are:

- The VBA will be developed and tested using the same billiards table all the time. Therefore conditions are reasonably controlled and no explicit requirements are posed on robustness outside the testing environment.
- Although the ultimate goal of a VBA would be to improve skills of a player, there is no way to accurately measure the effect of using a VBA in this regard on a short term. Therefore it is not an explicit requirement. However, in 1.4 it is reasoned why using a VBA, even a basic implementation, is expected to improve a player's skills.

The two main research questions are very general, and therefore should be split up in several sub questions.

1.2.1 How should the VBA calculate a solution to recommend to a player for any given ball configuration?

The number of possible ways to strike a cue ball is infinite. Therefore, every ball configuration has an infinite amount of possible solutions. Obviously, a VBA should limit the amount of solutions that it presents to the user to one best, or a couple of good solutions.

In order to solve a ball configuration, the VBA should first be able to determine what solutions are possible.

R.Q. 1.1: How to generate possible solutions for any given input ball configuration?

The method that will be used to generate possible solutions should meet the following requirements:

Req. 1.1.1: The method to generate the shots should be reasonably fast.

Req. 1.1.2: Solutions that are generated should be accurate, i.e. reproducible by a user on a real table.

Req. 1.1.3: All solutions that an expert would assess as reasonable for a particular ball configuration should be present.

Req. 1.1.4: If two solutions are considered the same by an expert, the method should cluster them and select one representative of the cluster.

Now that the VBA has a certain amount of possible solutions at its disposal, it needs to find one that should be recommended to the user.

R.Q. 1.2: How to select one solution from the set of all possible solutions that is to be recommended to the user?

The following requirements have to be met when selecting recommendable shots:

Req. 1.2.1: One solution should be recommended as the best one. The solution is considered best, when an expert trainer would recommend the same solution when he is presented the same ball configuration.

Req. 1.2.2: The VBA should be able to present good alternatives, if the user desires. An alternative solution is considered good, when an expert trainer would agree to the alternative being a plausible solution.

1.2.2 How to communicate a solution with the user?

Communication between the VBA and the outside world consists of two parts. Upon a user's request, the ball configuration should be acquired. The positions of the three balls within the table frame can then be fed to the solving algorithm. This poses the first sub-question of research question 2, dealt with in section 5.1.

R.Q. 2.1: How to present a ball configuration from a real world table to the VBA?

Several requirements have to be taken in to account while finding an answer to this question:

Req. 2.1.1: The method should be reasonably accurate, with a maximum deviation of a ball's radius. This is considered acceptable for the game of three cushion (explicitly not for other billiard games, especially libre).

Req. 2.1.2: The method should be as little obtrusive as possible, not restricting the freedom of movement for the user too much.

When the input is processed, the resulting recommended solution has to be presented to the user. The sub-question regarding this aspect is dealt with in section 5.2 and 5.3:

R.Q. 2.2: How to present a solution to a user?

The requirements that should be fulfilled regarding this research question are:

Req. 2.2.1: A billiards shot could be defined by several parameters, for example the velocity at which the cue hits the cue ball upon impact. These parameters should be chosen and presented so that the user is able to reproduce a shot accurately.

Req. 2.2.2: The user should be able to interact freely and intuitively with the VBA, he should not feel restricted by the interface.

1.3 Literature Review

A good amount of literature and related work is available, but no work was found specifically on assisting players by computationally solving three cushion shots. However projects were found that show overlap.

Lars Bo Larson describes an automated pool trainer platform [2]. This is a platform used at the university of Aalborg to experiment with multi modal user interaction. The basis of the system is a training method called Target Pool, which consists of sets of predefined practice shots. Users of the system have to carry out the practice shots, and get rewarded points for the outcome. This differs substantially from the goal of the VBA, where shots are not predefined but calculated from an input ball configuration. Furthermore, the project considers the game of pool, not the game of three cushion billiards.

However the modalities used are interesting and could be applied to the VBA. For example, input modalities range from computer vision (an interface is projected on the table, the user's hands are tracked to determine when the user interacts with this interface) to speech recognition. Output modalities tested are projection on the table cloth using a beamer, and an embodied conversational agent giving feedback.

Assisting the player by visualizing a shot using augmented reality is explored by T. Jebara et al [3]. A pool player is given a head mounted display, with a camera attached to it. The camera retrieves the ball configuration on the table, calculates a shot to recommend, and recommends it by drawing a virtual (via the display) line on the table, indicating the path the cue ball and object ball should travel. Unfortunately no usability tests were done, and no follow-up was found. The game used in the project is pool, which means the shot recommendation is more straightforward than finding three cushion solutions.

Solving shots for the game of pool billiards (which differs greatly from three cushion billiards in terms of game logic) computationally, was covered during the 10^{th} and 11^{th} ICGA Computer Olympiads. A physical model of a pool table was implemented by M. Greenspan and W. Leckie [4], on which contestants used different techniques to play a game of computational pool. This spawned some interesting game strategy techniques, most notably [5] by M.Smith. However, there is a fundamental difference in tactics between pool and three cushion billiards; pool involves looking ahead one or more shots. This implies the generation of a search tree, which is not necessary for three cushion billiards.

Multiple arithmetic shot solving systems exist that make use of the diamonds, the markers on the wooden frame surrounding the playfield. Jean Verworst presents several of these systems [6], that can be applied to different types of ball configurations. However, three difficulties arise:

- the need for categorizing of ball configurations into different types, needing their own system.
- there are irregularities in the systems that need to be fixed with complicated modifications.
- the systems are designed for a ‘match’-size billiard table (2.845m x 1.4225m). The table used by most three cushion players, and throughout this project, is of smaller size (2.30m x 1.15m). This results in the need for even more modifications.

These difficulties led to the decision not to implement Verworst’s systems but look for a more generic and flexible way to solve ball configurations (discussed in chapter 4). It should be noted however, that the book is highly respected and used by many (expert) three cushion players.

1.4 Personal Motivation

I will use this space to express my personal motivation for this project, which will be subjective and claims are speculative, but it does provide a broader context.

For years now I have been an enthusiastic three cushion player myself. Many things about three cushion I learned from an expert human trainer, Jelle Pijl (who was consulted several times during the project), but the bulk of improvement came from practicing with other players, or even alone. We would like to call this “training”, but it just came down to playing matches, seldom reasoning about the choices that were made, and why things went wrong when they did.

One of the things that struck me during these sessions, was that when a BC had to be solved, I’d purely rely on my gathered expertise, by taking a quick glance and immediately choose a solution. Then I would walk up to the table, line up for the shot, estimate where to hit the first cushion and shoot. When successful, the process repeated, when unsuccessful, the chair awaited, from where I could watch my opponent play until he in turn misses.

Now let’s split up the performing of a shot into a tactical part, that is the solution thought up (what path the cue ball should travel, and how to achieve this), and a technical part. The technical part is considered perfect when a shot is exactly produced as planned. The tactical part is considered perfect if the solution would result in a carombole if the technical part is perfect.

When a shot is successful, several things could have happened:

- The tactical part was perfect, as was the technical
- The tactical part was imperfect, as was the technical part, cancelling each other out
- Either the tactical part or the technical part or both were imperfect, but the carombole was made by shear luck, by means of a kiss or a random path leading to a valid point

Although any one of above scenario’s could have occurred, it did not really matter as the point is secured. So I get my reward, instead of punishment when one of the parts failed. This combined with the passive way of training obviously would not be the most feasible way of improving skills.

This phenomenon also works the other way around. When a shot is unsuccessful, it could either be due to failure in the technical part, as well as the tactical, or both. Often it is hard to conclude afterwards which part was wrong.

I became fascinated by the idea of using a computer to conduct either the technical part, or the tactical part, so that one of these two variables could be eliminated. Eliminating the technical part would require a robot conducting the shots perfectly. This happens to be explored for the game of pool, but is more of an electrical engineering issue. But developing a virtual assistant taking care of solving shots and recommend a solution fits well within the field of Human Media Interaction.

If the VBA is able deliver a best solution, and the user can trust to take it as 'ground truth', the user can concentrate on conducting the shot as advised, and draw conclusions afterwards. Furthermore, the less experience user that is unable to figure out for himself how to solve a shot, can now learn by playing good solutions, instead of "just trying". Also, if the recommendation would always be the 'best' according to certain measures, every player could test his own hypothetical 'best' solution to it.

Often I was told by the expert trainer, that before I conduct a shot, I should mentally project the complete shot being played out on the table, before attempting it. This mental trick is used in a lot of different sports where action has to be taken from a static situation, for example tennis, soccer (free kick) or golf. One of the reasons for this probably is also to factor out any doubt concerning the tactical part, so that the player can fully commit to conducting the technical part: scoring an ace, a goal, or a putt. This mental projection beforehand could also be greatly enhanced by the aid of a VBA.

1.5 Document Overview

The remainder of this document will discuss the different parts that were explored and tested, in order to enable the first prototype of a VBA. Chapter 2 describes the physical model that was established using the work of [4] and [7]. The physical model is key to the project, as its implementation enables the VBA to simulate billiard shots, in order to find solutions.

Chapter 3 discusses the analytical approach used for the implementation of the physical model, that ensures the implementation to be both fast and accurate.

The method that is used to find solutions to present to the user is described and discussed in chapter 4.

Now that the VBA is able to provide one or more solutions for any input ball configuration, a means has to be established to provide the VBA with the required input and to present the found solution(s) to the user. This interaction layer is explored in chapter 5.

Chapter 6 finally assesses the results collected during the project, and recommends possible improvements and paths for continuation.

Throughout the document, examples of billiard shots will be depicted like in figure 1.5.1. These images should be interpreted as follows:

The white ball is always the cue ball, throughout the whole project. Obviously, enabling the yellow ball to be appointed cue ball would be a trivial task, but it was chosen not to, as it is

not really necessary. Originating from the white ball is a white dotted line. The dots are a plot of the position of the white ball over time, in this case drawn at a resolution of 100 dots per second. The dotted line thus indicates the path the white travels from moment of impact, until stationary.

The same is done for the two object balls, which start moving immediately upon impact with the cue ball.

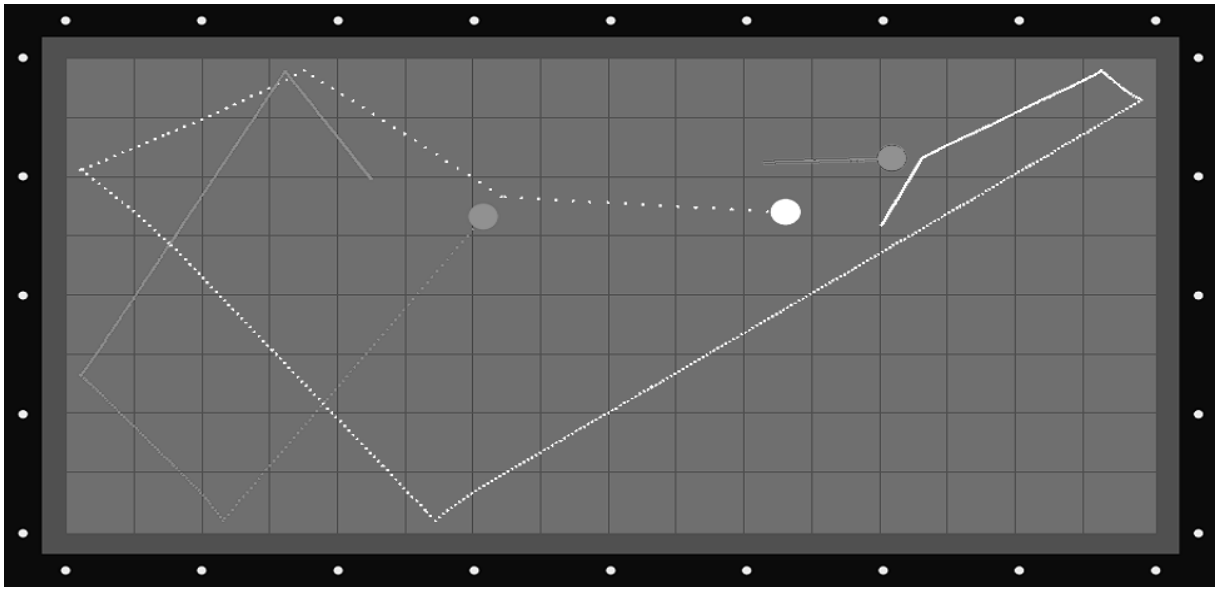


Figure 1.5.1: Example of an illustration of a three cushion shot

1.6 Acknowledgements

I would like to thank my supervisors Mannes Poel, Herwin van Welbergen, Zsófia Ruttkay and Anton Nijholt for helping me by providing useful comments, insights and guidance. Furthermore I would like to thank expert billiards trainer Jelle Pijl, who kindly shared his knowledge and assisted in evaluating shot recommendation. This research would be less valuable without the final user test, so I would like to thank the participants thereof. Finally I would like to thank family and friends for support and inspiration.

2. Physical Model of Billiards

In order to generate solutions, the VBA should be able to simulate billiard shots. In order to simulate billiard shots computationally, a physical model needs to be established. The requirements state that the user must be able to trust the advice of the VBA. This implies that the solutions the VBA produces need to be accurately reproducible on a real table. Therefore the physical model should be accurate enough to not violate this requirement.

To this extent, several literature resources were consulted. The most important are [7] and [4], in which the authors describe how they implemented a physical model to simulate billiard/pool shots. The ideas and principles used by the authors of [7] and [4] are used in this section to construct a model of the physics involved with billiards.

From the moment that the cue ball is struck with a cue until all balls are stationary again, several events can occur which need to be treated separately. This chapter starts with the modeling of the cue striking the ball, and the resulting initial velocities. Immediately after impact the ball will begin sliding. After a while, due to table friction, the sliding will evolve into a rolling motion. This and other aspects of the ball's motion will be discussed in section 2.2. Inherent to the rules of the game, the cue ball eventually has to collide with the other two balls on the table. Response to this collision is discussed in 2.3. Collision with a cushion is a different, more complicated subject as it cannot be treated as a near-perfect elastic collision. It's dealt with in section 2.4.

Three different coordinate reference frames are used in this chapter. The first is a three-dimensional ball-centric reference frame adopted from [4], which has axes ($\hat{\imath}$, $\hat{\jmath}$, $\hat{k}$) originating from the ball's center of mass (see figure 2.1.1), at t=0. This frame is used for the description of ball motion. After the ball is struck by a cue, it will start to move along the $\hat{\imath}$-axis.

Second is the 2-dimensional table reference frame, depicted in figure 2.1.2, with axes x and y, originating in the top left corner. This frame is used for the absolute position of the balls.

The third reference frame is adopted from [7], and is used to describe the collision of a ball with a cushion (figure 2.4.1). Here the y-axis is always tangential to the cushion, the x-axis perpendicular to the cushion and the z-axis perpendicular to the table cloth.

Vectors are indicated with an arrow, for example $\vec{v}$. Vector components are indicated by a subscript defining which component is meant, for example $\vec{v}_i$. Normalized and unit vectors are indicated with a caret, for example $\hat{u}$. Initial values for a vector are denoted with a subscript 0, for example $\vec{v}_0$ for velocity at time t=0.

2.1 Cue-Ball Impact

In [4] a method to model the impact of the cue on the ball is described. A ball is set in motion when it is struck with a cue. Let ($\hat{\imath}$, $\hat{\jmath}$, $\hat{k}$) be the ball-centric coordinate reference frame. Five impact parameters are used to describe the impact, depicted in figure 2.1.1:

1. ψ: The angle between the reference axis in the table frame (figure 2.1.2) and the initial direction of the ball, i.e. the aiming direction

2. V_0: The velocity of the cue at time of impact
3. a, b: the distance (horizontally, vertically) of the point of impact from the centre of the ball
4. θ: the angle of the cue relative to the table plane, i.e. how much the cue is tilted

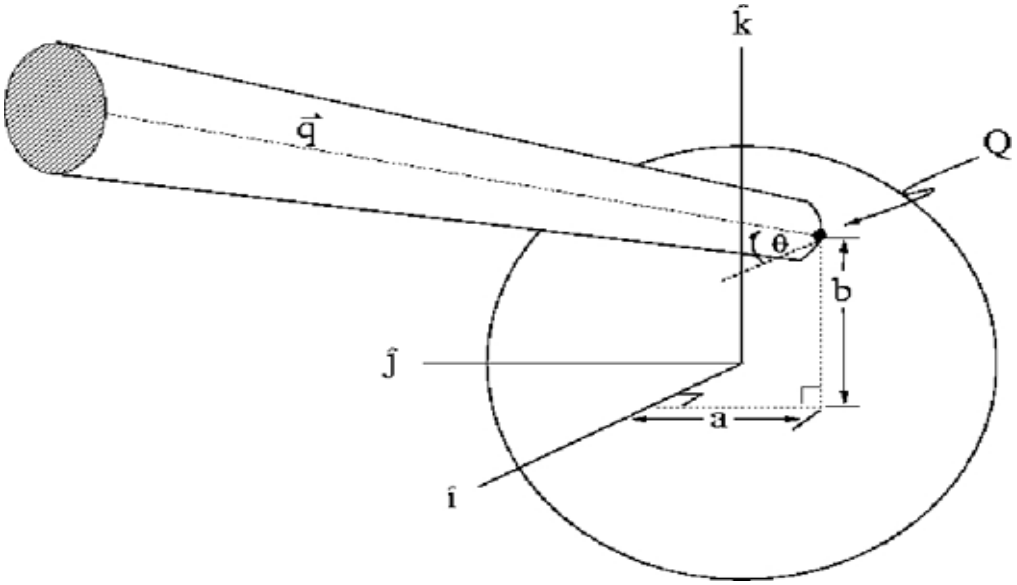


Figure 2.1.1: Taken from [4],visualization of parameters

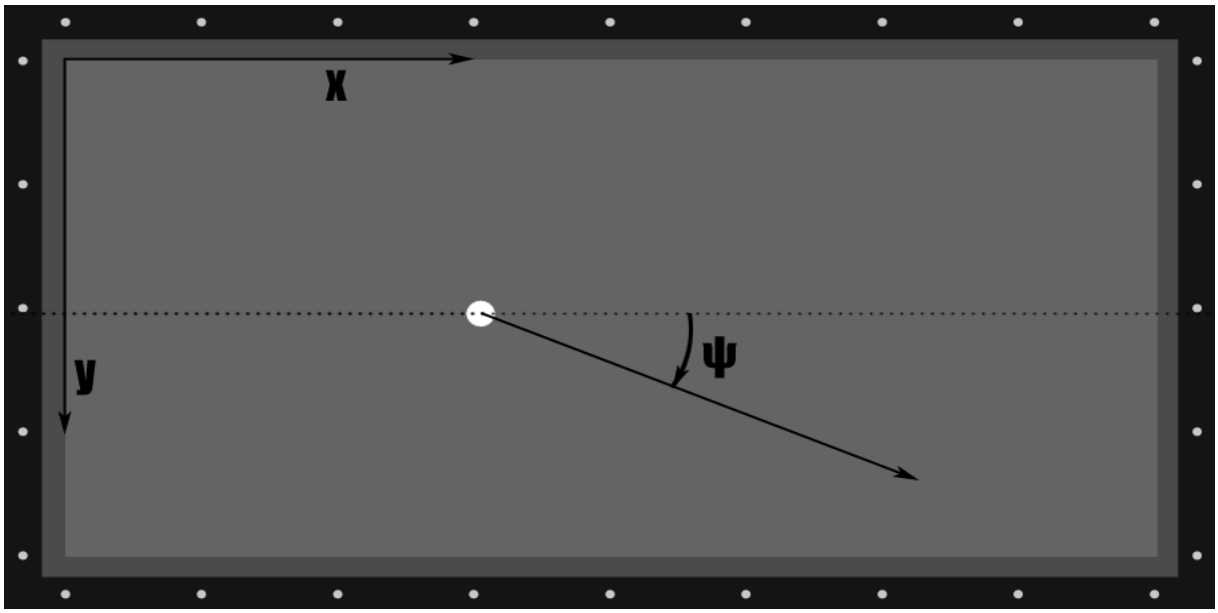


Figure 2.1.2: Table reference frame. Axes x and y originate in upper left corner. ψ is the aiming angle, relative to the line that intersects the center of the cue ball and is parallel to the x-axis.

Parameters a and b are used to control the initial rotational velocity of the ball around the $\hat{k}$ and $\hat{\jmath}$ axes. This gives the player several 'tools' at his disposal:

Variable a; side spin (or English): increasing the distance a results in a greater amount of angular velocity around the $\hat{k}$-axis. This is particularly useful for influencing the outgoing

angle of the cue ball after it hit the cushion. In three cushion play, this type of spin is used almost always (for example, see [6]).

Variable b; Top spin (follow) or back spin (draw): increasing the distance b results in a greater amount of angular velocity around the $\hat{\jmath}$-axis. This is particularly useful for manipulating the path of the cue ball after a collision with the first object ball. This is further explained in section 2.3.

Shots with $\theta > 0$, where the cue is tilted, induce a rotation around the $\hat{\imath}$-axis which results in a curvilinear path. Because of the greater difficulty shots with an elevated cue are very rarely used by three cushion players. Therefore the parameter θ is set to 0 throughout the project. A cue striking the cue ball will now be defined by four parameters, annotated as the tuple $[a, b, V, \psi]$.

The impact of the tip of the cue is modeled in [4] as an instantaneous point impact. After impact the ball will be moving along a linear path defined by the $\hat{\imath}$-axis of the ball reference frame.

For calculation of the magnitude F of the force imparted on the ball by the cue impact, the following equation is used in [4]:

Eq.2.1.1 $$F = \frac{2mV_0}{1 + \frac{m}{M} + \frac{5}{2R^2} + (a^2 + b^2 cos^2\theta + c^2 sin^2\theta - 2bc\cos\theta sin\,\theta)}$$

R is the ball's radius (0.031m), m = the ball's mass (0.21kg), M = the cue's mass (~0.5kg), and c is the $\hat{k}$-component of impact point Q , defined by (a, b, c).

Note that F in this case is an integration of Newton's second law, the change in velocity is treated as instantaneous.

Due to the restriction that $\theta = 0$, eq. 2.1.1 is rewritten to:

Eq.2.1.2 $$F = \frac{2mV_0}{1 + \frac{m}{M} + \frac{5}{2R^2} + (a^2 + b^2)}$$

The impact is assumed instantaneous. Integrating Newton's second law gives $F = m\vec{v}$, where $\vec{v}$ is the initial translational velocity of the cue ball. Expressed in the ball-centric reference frame:

Eq.2.1.3 $$\vec{v} = \left(\frac{-F}{m}, 0, 0\right)$$

The initial rotational velocity of the cue ball, immediately after impact, is (also expressed in the ball-centric reference frame):

Eq.2.1.4 $$\vec{\omega} = \frac{1}{I}(0, bF, -aF)$$

I is the standard moment of inertia for a sphere, $\frac{2}{5}mR^2$.

The equation used to calculate F suggests a strong relationship between the amount of spin applied (a, b), and the initial velocities of the cue ball. This is visualized for the initial velocities in figure 2.1.3. It shows that increasing a and b results in a strong decrease of initial translational velocity $\vec{v}$. This behavior wasn't observed in real life. Furthermore it is more convenient when generating shots for finding solutions to have initial translational velocities uniform, independent of a or b. Therefore eq.2.1.2 was simplified further to

Eq.2.1.5 $$F = \frac{2mV_0}{1 + \frac{m}{M} + \frac{5}{2R^2}}$$

Although this may seem oversimplified, it was decided to use eq.2.1.5 for modeling cue impact, as the exact relation between cue velocity,a,b and initial ball velocity is not very relevant for the tactical part of three cushions.

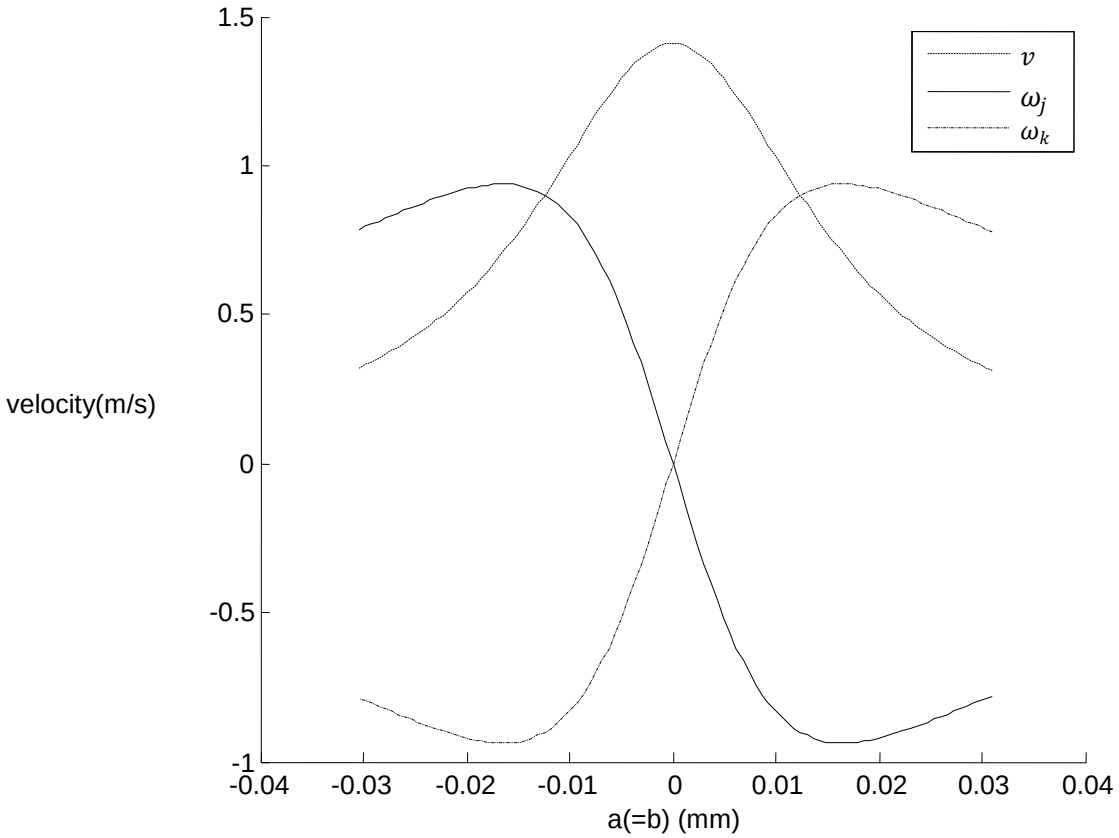


Figure 2.1.3: relation between a,b and initial velocities

2.2 Ball Motion

According to [4], a ball in motion can be in one of four different motion states: sliding, rolling, spinning or stationary. Immediately after the ball is struck, it will begin sliding along a rectilinear path along the $\hat{\imath}$-axis, with a translational velocity $\vec{v}$ and rotational velocity $\vec{\omega}$. These velocities are combined into an equation representing the concept of relative velocity, introduced by [8]:

Eq.2.2.1 $$\vec{u}(t) = \vec{v}(t) + R\hat{k} \times \vec{\omega}(t)$$

Which can be rewritten as:

Eq.2.2.2 $$\vec{u}(t) = \begin{bmatrix} v_{\hat{\imath}} - R\omega_{\hat{\jmath}} \\ v_{\hat{\jmath}} + R\omega_{\hat{\imath}} \\ v_{\hat{k}} \end{bmatrix}$$

R is the ball radius (0.031m), the reference frame is the same ball-centric frame as in fig. 2.1.1. Note that the relative velocity has a zero $\hat{k}$-component, because $v_{\hat{k}} = 0$ (shots that include a 'flying' ball are not encountered in normal three cushion play).

While $\vec{u}(t) \neq 0$ the ball will be in a sliding state. When $\vec{u}(t) = 0$ the ball is said to be in rolling state. Immediately after cue impact the ball will be in sliding state, with a translational velocity along the $\hat{\imath}$-axis, and a rotational velocity around the $\hat{\jmath}$-axis. Following from eq.2.2.2, the ball will be in rolling motion when

Eq.2.2.3 $$v_{\hat{\imath}} = R\omega_{\hat{\jmath}}$$

which corresponds to a point P on the surface of the ball traveling at the same velocity ($R\omega_{\hat{\jmath}}$) as the center of the ball ($v_{\hat{\imath}}$). The restriction $\theta = 0$ already ensures $v_{\hat{\jmath}} = 0 = R\omega_{\hat{\imath}}$ (this is different after a collision with another ball or a cushion occurs).

During the sliding state, the motion equations governing the ball's motion are [4]:

Eq.2.2.4 $$\vec{r}_B(t) = \begin{bmatrix} v_{0i}t - \frac{1}{2}\mu_s g t^2 \hat{u}_{0i} \\ v_{0j}t - \frac{1}{2}\mu_s g t^2 \hat{u}_{0j} \end{bmatrix}$$

Eq.2.2.5 $$\vec{v}_B(t) = \vec{v}_0 - \mu_s g t \hat{u}_0$$

Eq.2.2.6 $$\vec{\omega}_B(t) = \vec{\omega}_0 - \frac{5\mu_s g}{2R} t(\hat{k} \times \hat{u}_0)$$

Eq.2.2.7 $$\omega_k(t) = \omega_{k0} - \frac{5\mu_{sp} g}{2R} t$$

The subscript B's indicate that the ball-centric reference frame is used. The subscript 0 indicates initial values (at t=0). g is the gravitational constant. μ_s and μ_{sp} are frictional coefficients, discussed in section 2.2.1.

Eq.2.2.4 expresses the displacement $\vec{r}_B(t)$of the ball after time t. It shows that when the normalized vector $\hat{u}_0$ has a non-zero $\hat{\jmath}$-component, the ball moves in a non-rectilinear path. As explained earlier, this is not possible immediately after cue-impact (as $\theta = 0$). However in section 2.3 and 2.4 it is explained that it is possible after ball-ball or ball-cushion impact.

As shown in eq.2.2.7 the motion around the $\hat{k}$-axis is treated separately, while it is the same in both sliding as rolling state. The separate treatment of $\omega_k$also implies that there's another state, spinning, when the ball has no translational velocity, but is spinning around it's vertical axis.

During the rolling state the motion equations are:

Eq.2.2.8 $$\vec{r}_B(t) = \begin{bmatrix} \vec{v}_{0\hat{\imath}}t - \frac{1}{2}\mu_r g t^2 \\ 0 \end{bmatrix}$$

Eq.2.2.9 $$\hat{v}_B(t) = \hat{v}_o - \mu_r g t \hat{v}_0$$

From the definition of rolling state follows that:

Eq.2.2.10 $$\omega_{\hat{\jmath}}(t) = \frac{v_{\hat{\imath}}}{R}$$

Eq. 2.2.1 to 2.2.10 are the equations needed to update all motion variables of the balls after time t. As the displacement $\vec{r}_B$is within the ball-centric frame, it needs to be converted to a displacement within the table frame (figure 2.1.2). This is done using the following translation and rotation:

Eq.2.2.11 $$\vec{r}(t) = \vec{r}_0(t) + \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \vec{r}_B(t)$$

2.2.1 Friction Coefficients

In [4] the authors use friction coefficient determined by W. Marlow in [8]. Friction during the sliding state, denoted as μ_s is determined to be 0.2, friction during rolling state (μ_r) is said to be 0.016. These values, when implemented, resulted in a 'slow' table. Three cushion billiard tables generally have a smoother cloth than pool tables. One of the reasons for this is that the surface beneath the cloth is heated at all times, preventing moisture in the cloth. For the evaluations discussed in chapter 4 (solving shots), a value of 0.008 was empirically determined for friction during the rolling state, by comparing simulated shots with the same shots conducted on the real table.

The spinning coefficient μ_{sp} is said to be 0.044 in [4], however for the project a value of 0.022 was determined to give better results. All friction coefficients are assumed constant.

2.3 Ball-Ball Impact

The author of [7] states that it is sufficient to treat the collision between two billiard balls as an almost perfect elastic rigid-body collision ($e = 0.98$), neglecting frictional effects and using the principle of momentum conservation. It should be noted however that in reality the impact is not frictionless, see for example [9].

Figure 2.3.1 shows what happens when a moving cue ball (CB) collides with a stationary object ball (OB). The CB has an initial velocity (vector) v_1, and final velocity V_1, the OB has an initial velocity v_2 and final velocity V_2. The principle of momentum conversation states that $m_1 v_1 + m_2 v_2 = m_1 V_1 + m_2 V_2$. As the balls masses m_1 and m_2 are equal, and $v_2 = 0$, this rewrites to $v_1 = V_1 + V_2$. The direction of the post-impact CB velocity vector V_1 will be tangential to the collision surface, V_2 will be perpendicular to the collision surface. The magnitudes of $V_1$and $V_2$are now found by splitting $v_1$into its components tangential and perpendicular to the collision surface.

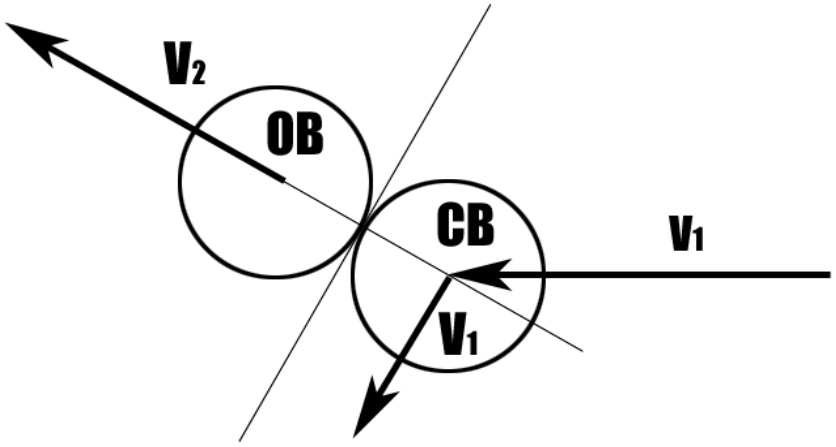


Figure 2.3.1: Ball-ball Collision Response

After collision, the ball-centric reference frame is reset so that the $\hat{\imath}$-axis is again defined by the direction of the ball immediately after impact, and ψ is updated.

Only collisions where the cue ball is moving and the object ball lies still are resolved. There is no need to resolve collisions between two moving balls, as producing valid three cushion shots involving two moving balls colliding is not considered normal practice, and thus should not be recommended as a solution by the VBA.

As a result of a ball-ball collision, the translational velocity will change in magnitude as well as direction (relative to the table frame). Because of the frictionless impact the angular velocities don't change relative to the table frame. Therefore the rotational velocities do change relative to the ball-centric reference frame. This results in a non-zero angular velocity around the $\hat{\imath}$-axis.

For example, if the difference between direction ψ of the ball immediately before and after impact is 1/2π, a rotation matrix must be applied to calculate the rotational velocities relative

to the rotated reference frame. If for example $\vec{\omega} = \begin{bmatrix} 0 \\ 30 \\ 15 \end{bmatrix}$ before impact, the new rotational velocities within the ball reference frame are: $\overrightarrow{\omega'} = \begin{bmatrix} \cos(\frac{\pi}{2}) & -\sin(\frac{\pi}{2}) & 0 \\ \sin(\frac{\pi}{2}) & \cos(\frac{\pi}{2}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 30 \\ 15 \end{bmatrix} = \begin{bmatrix} -30 \\ 0 \\ 15 \end{bmatrix}$

The cue ball will be in a sliding state again, because the relative velocity is non-zero. During this sliding state, the angular velocity around the $\hat{\jmath}$-axis influences the direction of the displacement due to friction with the table cloth (see eq.2.2.4), resulting in a curvilinear path. This phenomenon is used frequently by billiard players, especially pool, snooker or libre players, but also in three cushion billiards.

A way to manipulate the path of the cue ball after impact with the first object ball is for example the draw shot, where b<0. A draw shot will result in an angular velocity of the ball that is opposed to the translational velocity of the ball. After impact this angular velocity will result in the ball being “drawn back” to the player. This is illustrated in figure 2.3.2.

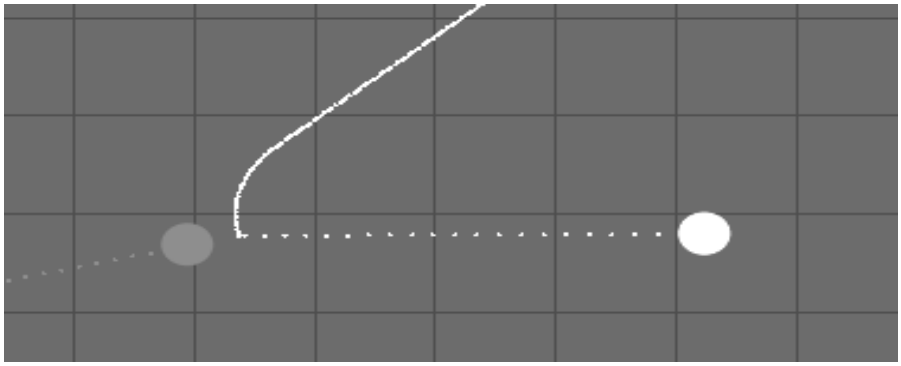


Figure 2.3.2: draw shot

2.4 Ball-Cushion Impact

The modeling of the collision between a ball and a cushion is described in [7] by Prof. Inhwan Han. The research is the only one found on this specific topic. The results of the implemented model in [7] are said to agree well with experimental data.

The collision between the ball and a cushion is treated in [7] as a three dimensional near instantaneous point-impact, at point a (figure 2.4.1). The model draws upon earlier work in the field of impact analysis of rigid body dynamics where different cases of impact are considered, depending on relative sliding and compressive velocities. Two cases are considered, ‘Sticking and Sliding’ as well as ‘Forward Sliding’. Both are implemented for the VBA, but it is observed that the case of ‘Forward Sliding’ rarely occurs. Therefore only the case of ‘Sticking and Sliding’ will be discussed here.

Figure 2.4.1 depicts a collision between a ball and a cushion. $\epsilon = 10mm$. Note that the reference frame changes from ball-centric to cushion centric, with an Y-axis tangential and an x-axis perpendicular to the cushion. The velocities of the ball therefore have to be rotated using a rotation matrix, to obtain velocities v_X, v_Y, ω_X and ω_Y.

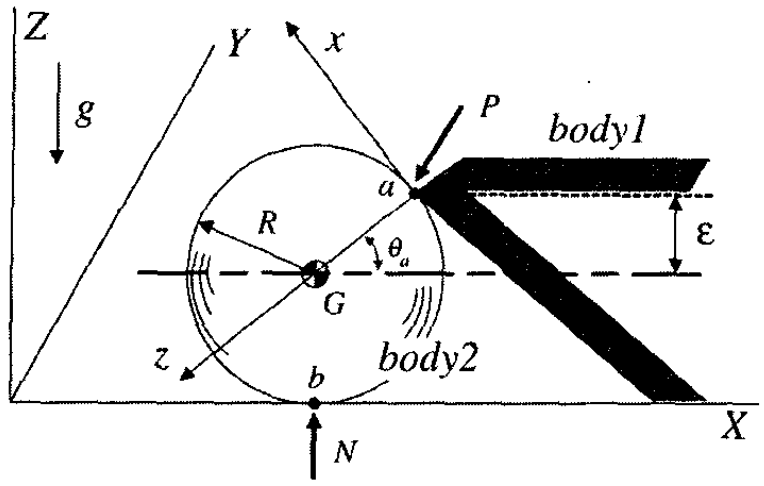


Figure 2.4.1: Taken from [HAN05], Impact between ball and cushion rail

Linear and angular impulse-momentum laws govern the motion of the bodies during the small impact period:

Eq.2.4.1 $m(v_X - v_{X0}) = P_x, m(v_Y - v_{Y0}) = P_Y, P_Z = 0$
$I(\omega_X - \omega_{X0}) = -RP_Y \sin(\theta_a)$

Eq.2.4.2 $I(\omega_Y - \omega_{Y0}) = RP_X \sin(\theta_a) - RP_Z \cos(\theta_a)$
$I(\omega_Z - \omega_{Z0}) = RP_y \cos(\theta_a)$

For the sticking and sliding case the impulses are obtained using the following equations:

Eq.2.4.3 $P_X = -\frac{s_{x0}}{A}\sin(\theta_a) - (1+e)\frac{c_0}{B}\cos(\theta_a)$
$P_Y = -\frac{s_{y0}}{A}$

s_{x0},s_{y0} and c are the relative velocity of sliding and compression of the points in contact (a) :

Eq.2.4.4 $s_x = -v_{ax} = v_X \sin(\theta_a) - v_Z cos(\theta_a) + R\omega_Y$
$s_y = -v_{ay} = -v_Y - R\omega_Z cos(\theta_a) + R\omega_X sin(\theta_a)$
$c = -v_X cos(\theta_a) - v_Z sin(\theta_a)$

A and B are constants (I is the moment of inertia of a solid sphere: $\frac{2}{5}mR^2$):

$$A = \frac{1}{m} + \frac{R^2}{I} = \frac{7}{2m}$$

Combining the above equations gives outgoing translational and rotational velocities.

These are transformed back to the ball-centric reference frame. As with ball-ball collision, a cushion collision often results in a rotational velocity of the ball around the $\hat{\imath}$-axis of the ball reference frame, resulting in a curvilinear path. See for example figure 2.4.2.

Although the curving is not very strong, neglecting it would result in too big an inaccuracy.

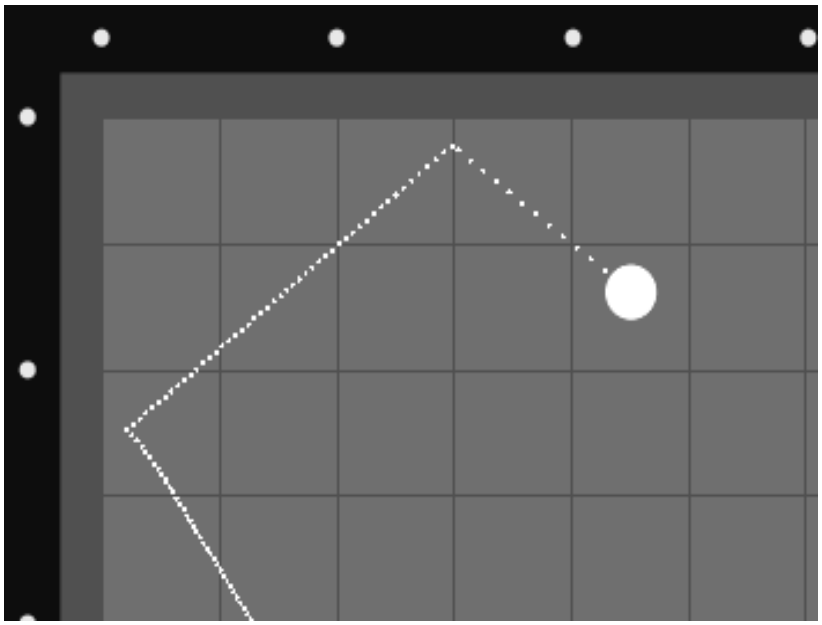


Figure 2.4.2: Slight curvilinear path after (second) ball-cushion collision

In eq. 2.4.3 e is the coefficient of restitution between ball and cushion, experimentally determined by Prof. Inhwan Han as the following relation:

$$e = 0.39 + 0.257v_{x0} - 0.044{v_{x0}}^2$$

I.e., the coefficient depends on the relative normal velocity of the ball at time of impact. However, this relation did not lead to good simulation results. A test set of 15 three cushion paths was constructed on the real table, and projected on a simulated table that implemented the model. Figure 2.4.3 depicts the result of one simulated shot using the e-coefficient as derived from [7].

The straight line is the path the cue ball traveled on the real table, the dotted line shows the path the cue ball travels within the simulation.

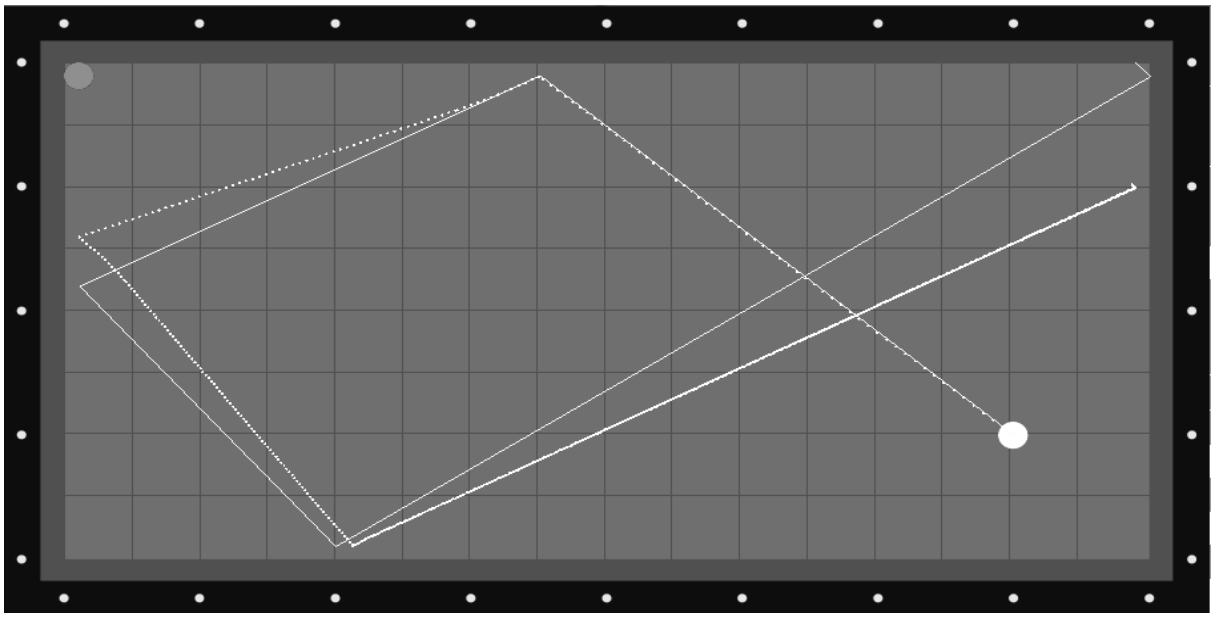


Figure 2.4.3: e-coefficient from [HAN05]

Using the test set several values for the parameters of the cubic equation used to establish e in [7] were tried, but leaving dependency on relative normal velocity v_{x0} out of the equation and just using a static value of $e = 0.85$ yielded much better results altogether. It is unknown why this is. It could be the formulae from [7] were misinterpreted/wrongly implemented by me. However, it could also be that the billiard table used to determine the equation for e in [7] differs too much from the table used for this project. The path simulated in 2.4.3 resembles a shot on an old table, on which the rubber cushions are worn out. However, if this is the case for the table used to determine e in [7] remains speculation. Using $e = 0.85$ however was found to yield sufficient accuracy, enough to continue the project with. Figure 2.4.4 shows the same shot as in figure 2.4.3, with $e = 0.85$. Appendix C shows more shots from the evaluation, all are played with left-hand side spin.

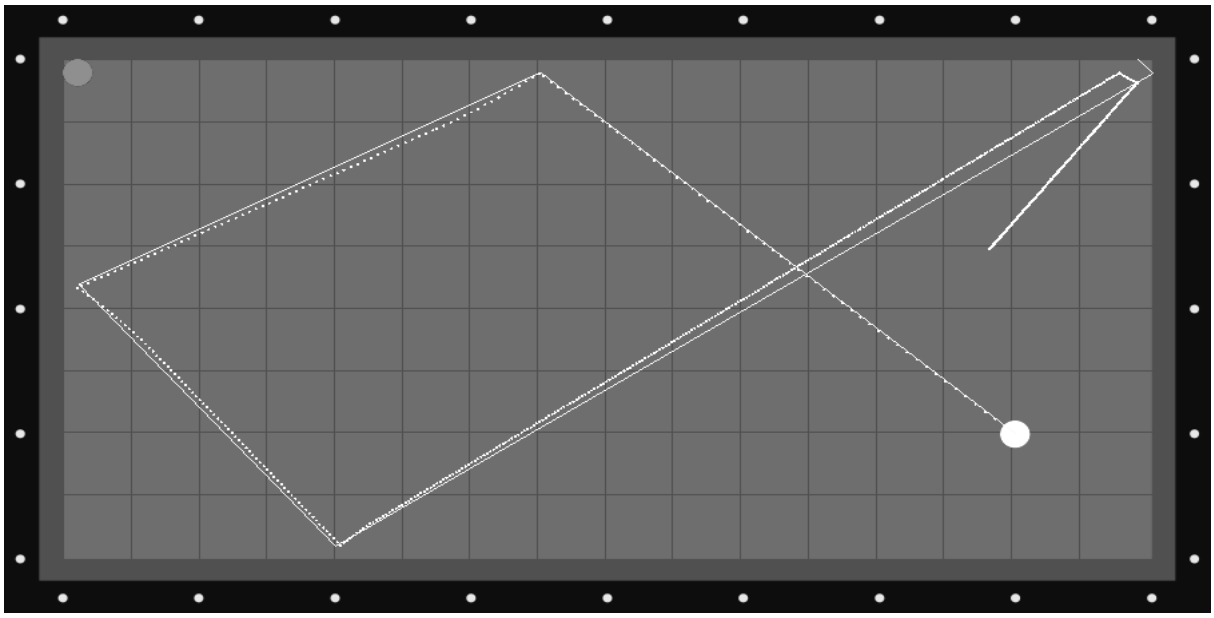


Figure 2.4.4: e = 0.85

3. Simulation of Billiards

In [4] the authors present an event-based approach to simulate a game of pool. Instead of numerically simulating a pool shot, recalculating and updating motion variables and ball positions after a fixed time interval Δt repeatedly until all balls are stationary, the authors define all events that can occur, and calculate analytically the time these events occur. The time of the first event to occur is determined, and all motion variables are updated. After that, the next event is determined and processed, repeating until all balls are stationary.

This method offers various benefits over the classical numerical approach, namely:

- Greater accuracy. The numerical approach can detect whether an event occurred during a particular time step. This can then be refined using smaller time steps, but the exact time of occurrence still can only be approximated. Using the analytical approach results in exact times of event occurrence, accuracy is only dependent on round-off errors.
- Less calculations needed to process a complete shot, from moment of cue impact until all balls are stationary, thus less computational cost[4].

Requirement 1.1.1, generation of shots should be fast, is met by using the event based approach of simulating shots.

The implementation of the model detects the following events:

- Motion state transition
- Ball-ball collision
- Ball-cushion collision

The remainder of this chapters discusses these events and how their time of occurrence is calculated.

3.1 Time of Motion State Transition

Calculating the time of the first motion transition is done the same way as in [4]. Equations 2.2.1, 2.2.5 and 2.2.6 are combined to eq.3.1.1. Using eq. 3.1.1 the equation to calculate the time that the relative velocity $\vec{u}(t) = 0$, which by definition is the time the ball enters the rolling motion state. This time is given by eq.3.1.2.

Eq.3.1.1 $$\vec{u}(t) = \vec{u}_0 - \frac{7}{2}\mu_s g t \hat{u}_0$$

Eq.3.1.2 $$\tau_S = \frac{2|\vec{u}_0|}{7\mu_s g}$$

[4] For example, a test run was performed calculating 100.000 random shots (balls placed random, cue ball hit in random direction with velocity between 2.5 and 3.5 m/s). This resulted in 1,033,149 events, all detected and processed in 6536ms.

The ball will be in rolling state until $\vec{v}(t) = 0$. Using equation 2.2.9 the equation for the duration of the rolling state is derived to equation 3.1.3.

Eq.3.1.3 $$\tau_R = \frac{|\vec{v}_0|}{\mu_r g}$$

Again, the rotational velocity around the $\hat{k}$-axis is treated separate, following from equation 2.2.7:

Eq.3.1.4 $$\tau_{SP} = \frac{2R\omega_k}{5\mu_{sp} g}$$

3.2 Time of Ball-Cushion Collision

To determine when a ball will hit a cushion the collision time of the ball with all of the four cushions is determined. The smallest non-zero value this produces is the time of the first cushion collision.

To determine when, for example, the ball will hit the left short cushion, we simply calculate the time at which the x-component of the position of the ball in the table frame equals R:

Eq.3.1.5 $$\vec{r}_0(t) + \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \vec{r}_B(t) = \begin{bmatrix} R \\ .. \end{bmatrix}$$

Which boils down to solving the equation (assuming the ball is in rolling state, see eq.2.2.8):

Eq.3.1.6 $$\vec{r}_0(t) + \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} \vec{v}_{0\hat{\imath}} t - \frac{1}{2}\mu_r g t^2 \\ 0 \end{bmatrix} = \begin{bmatrix} R \\ .. \end{bmatrix}$$

Eq.3.1.7 $$\cos(\psi)\left(\vec{v}_{0\hat{\imath}} t - \tfrac{1}{2}\mu_r g t^2\right) - \sin(\psi)\left(\vec{v}_{0\hat{\imath}} t - \tfrac{1}{2}\mu_r g t^2\right) - R + \vec{r}_{0x}(t) = 0$$

This results in a quadratic equation:

Eq.3.1.8 $$-(\cos(\psi) - \sin(\psi))\tfrac{1}{2}\mu_r g t^2 + (\cos(\psi) - \sin(\psi))\,\vec{v}_{0\hat{\imath}} t + \vec{r}_{0x}(t) - R = 0$$

Which can be solved computationally using standard techniques. The outcome will be one or 2 real roots. If two solutions are provided, the smallest positive one is picked. No positive roots indicates the ball is moving away from the cushion.

The same method is used for finding collision times with the upper, right and lower cushion. From these four again the smallest positive one is picked, representing the time of the first cushion collision.

This approach looks as if it could benefit from culling, i.e. predicting which cushions could and which could not be hit before calculating collision times, but that is not the case, as balls can travel curvilinear paths. Therefore it is not easily determined when a ball is moving away from a cushion, that this cushion will not be hit first.

3.3 Time of Ball-Ball Collision

The time of ball-ball collision is somewhat more complicated, as we need to find a time t at which the distance between two balls is exactly $2R$:

Eq.3.1.9 $$2R = \|r_1(t) - r_2(t)\|$$

R is the ball radius, $r_1(t), r_2(t)$ represent the positions of the two balls within the table frame at time t. Rewriting this equation like in 3.1.2 would result in a quartic equation (as done in [4]), which is more computationally complex and costly to solve than a quadratic equation.

In the next chapter it is explained that the event-based simulation is used to simulate a great amount of shots. Only a small percentage of these simulated shots will result in a valid three cushion shot. Because of the vast amount, and the requirement that shots should be generated fast, the simulation should be fast. Therefore a trick is applied, enabling calculation of ball-ball collision using a quadratic solver.

Assuming one ball is moving on a straight line, and one ball is stationary, we can make use of a common collision detection technique to find the distance the cue ball travels before it collides with the object ball (see figure 3.1.1):

The angle α between the direction of the motion of the cue ball (CB) and a straight line between CB and the object ball (OB) is known. e is the line originating from and perpendicular to the balls motion direction to the center of the OB. The length of e can be calculated: $e = \sin(\alpha)\, b$. When $e > 2R$, with R the ball radius, the two balls will not collide and further calculation can be discarded. If not the next step is to determine f. Because c is also known, $c = 2R$, f can be determined by: $f = \sqrt{e^2 + c^2}$. The last step is to determine d. $d + f = \cos(\alpha)\, b, f = \cos(\alpha)\, b - d$.

Now that the distance to point of impact d is known (as far as a collision will occur), the time the collision will take place can easily be determined. For example, when the ball is in rolling motion the following quadratic equation should to be solved:

Eq.3.1.10 $$-\frac{1}{2}\mu_r g t^2 + \vec{v}_{0i} t - d = 0$$

This method is only precise when the ball is following a rectilinear path, however slight curves in the ball's path caused for example by a cushion collision did not result in much loss of accuracy. Stronger curves caused by ball collision however result in failing to correctly predict a collision.

Another downside of this method is that only one ball must be moving, the other must be stationary.

However, these flaws are detected by numerical simulation of the billiard shot, explained in next in section 3.4.

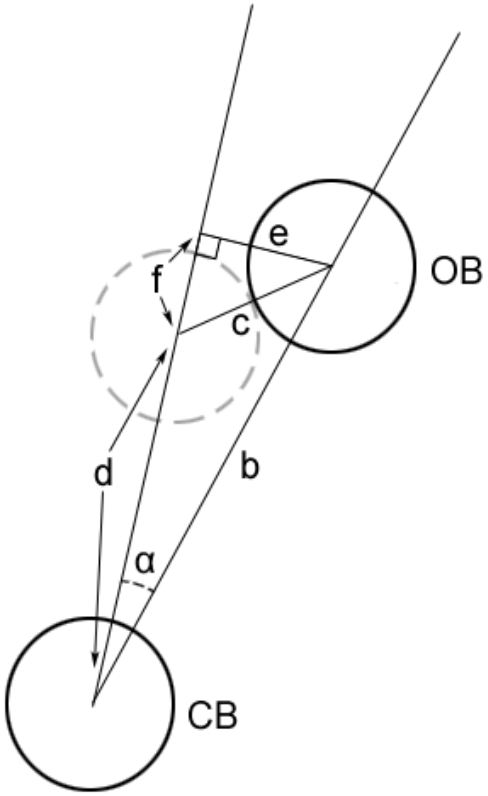


Figure 3.1.1: Distance to Impact

3.4 Numerical Simulation

The analytical simulation is used to simulate a great amount of shots fast, numerical simulation is implemented for 3 purposes:

- Checking for ball-ball collisions that were not detected by the analytical simulation
- Plotting the path of the ball
- Animating the path of the ball

Numerical simulation makes use of the event-based approach, by predicting the time of a next event and then find the positions of the balls in steps of size Δt from t_0 until the time of the next event. This process repeats until all balls are stationary and no new event will occur.

3.5 Test Application

The physics discussed in chapter 2 were implemented using the event based method. An application was built to test the implementation of the model, simulate shots and try out different friction and elasticity coefficients, comparing simulated shots to shots carried out on the real table. A screen shot of the application can be found in Appendix D. The test application was implemented using Java, and implements all the physics discussed in Chapter 2, using the event based approach discussed in this chapter.

4. Finding Solutions for Ball Configurations

Now that an implementation of a billiards physics model is available, it can be used to simulate three cushion shots in order to find valid ones, that can be presented to the user. The research questions that are answered in this chapter are: *How to generate possible solutions for any given input ball configuration?* and *How to select one solution from the set of all possible solutions that is to be recommended to the user?*

In an earlier attempt to solve three cushion shots some ideas from [6] were implemented. The solving method implemented was the so called LKL-system, but this system could only be applied to a limited faction of all possible ball configurations. A short evaluation was carried out, by playing 50 shots. Of these, 21 could possibly be solved using the LKL-system, and in many cases these would not have been a preferable solution (according to own expertise). Therefore generating solutions using these already available solving methods was found not suitable, as it does not meet the requirement that state that the VBA should be able to recommend a solution for every possible ball configuration, nor the requirement that the shot recommended as best is the same as a human expert would recommend.

It was decided a more generic approach is preferred. The method now used is discussed in this chapter and consists of two steps. The first step is generating a set of possible solutions using the possibility to simulate shots fast and accurately, the second step is ordering this set and selecting a shot that a real life trainer would advice a player to play. This chapter first deals with the generation of solutions in section 4.1, followed by the sorting of found solutions in section 4.2.

4.1 Generating Possible Solutions

As explained in chapter 2, the implementation of the physical model using an event based approach enables the VBA to simulate a great amount of shots in a short time. A straightforward approach to shot generation would be using every possible combination $[a, b, V, \psi]$ as input shot parameters, and select the ones that result in a valid 3-cushion carombole.

However, because there is an infinite amount of possible shots due to the continuous nature of the shot parameters a, b, V and ψ, it is impossible to just simulate all possible shots and select only the shots that are successful three cushion solutions.

Therefore, the amount of different input shot parameters values needs to be limited. Section 4.1.1 will discuss the decisions made on limiting and discretizing the parameters.

The limited input shot parameters are combined into a finite amount of input $[a, b, V, \psi]$ tuples. These are simulated and searched for valid shots, which is described in section 4.1.2.

4.1.3 describes the organizing of the resulting solutions into clusters, 4.1.4 finally describes an evaluation of the method of generating shots, to find out whether or not found solutions can be reproduced on a real life table.

4.1.1 Limiting Input Shot Parameters Values

This section discusses the discretization and limiting of the input shot parameter (ISP) values.

Initial cue velocity (V)

During the construction/testing of the physical model, comparing it with shots played on a real table, it was empirically determined that most three cushion shots are struck with an impact velocity between 2.5 and $4.0\ m/s$. Generating shots with V less than $2.5\ m/s$ thus is not very useful, as it would result in few valid three cushion shots. Generating shots with $V > 4.0\ m/s$ produces solutions that will be very hard if not impossible to reproduce. Although these boundaries are not exact, intuitively searching for shots outside the boundaries will not result in more/better solutions.

For the generation of shots, one or more different values for V should be chosen within these boundaries.

Initial direction (ψ)

The range of ψ is $-\pi < \psi <= \pi$. A crude approach would be to define a number of steps to cycle through the possible values of ψ. A smarter approach is to make use of an existing heuristic in three cushion billiards:

1. If OB1 and OB2 are positioned close to each other, a cushion first solution could be an option (see 4.1.1a).
2. If they are not positioned close to each other, the CB should be played first hitting one of the two object balls (see 4.1.1b).

In the cushion first case, the value ψ is cycled from –π to π in n_{cf} steps of size $2\pi/n_{cf}$. (Note that for certain values of ψ the cue ball will hit an object ball first instead of a cushion, however these could result in valid solutions and don't need to be discarded).

For the object ball first case, the yellow ball is chosen as the ball to hit first. A minimum and maximum value are chosen for ψ, representing the full range of ψ where the yellow ball is hit directly. This value is then discretisized into n_{bf} steps. The same is done for the red ball as the ball to aim at.

The limiting of possible ψ is now established by choosing appropriate values for the number of steps n in both cases. Figure 4.1.2 illustrates the range of ψ, with the yellow ball as OB1. 4.1.2a is at low initial velocity, $n_{bf} = 10$ steps. 4.1.2b shows shots generated with higher, but still low velocity and $n_{bf} = 180$ steps. Finally 4.1.2c shows the same shot with high initial velocity. You can see that for a portion of the shots generated in 4.1.2c the white collides with the red in the corner. These are the ones that could be valid shots, if three cushions were hit before the collision with the red.

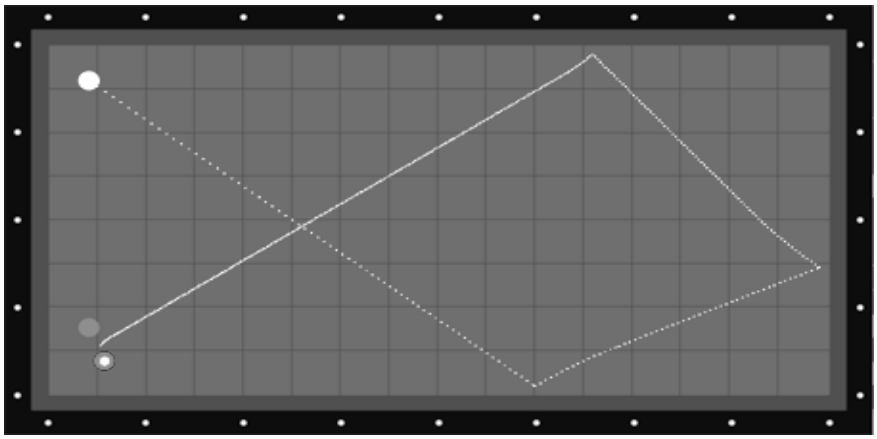


Figure 4.1.1a: Cushion First Example

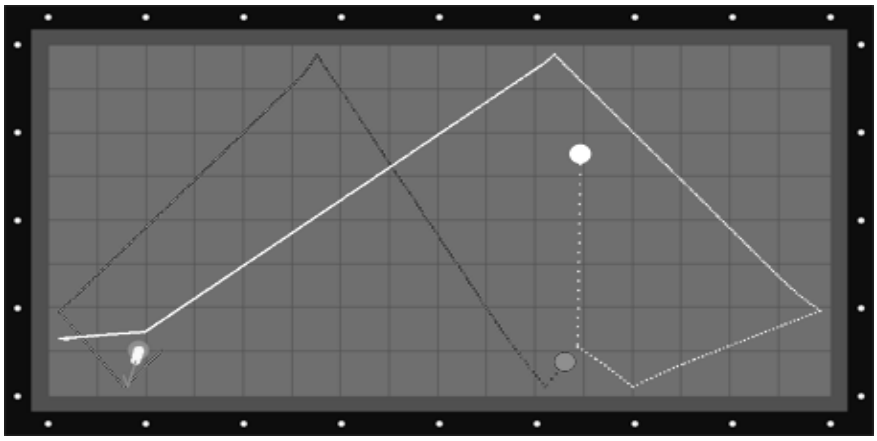


Figure 4.1.1b: Ball First Example

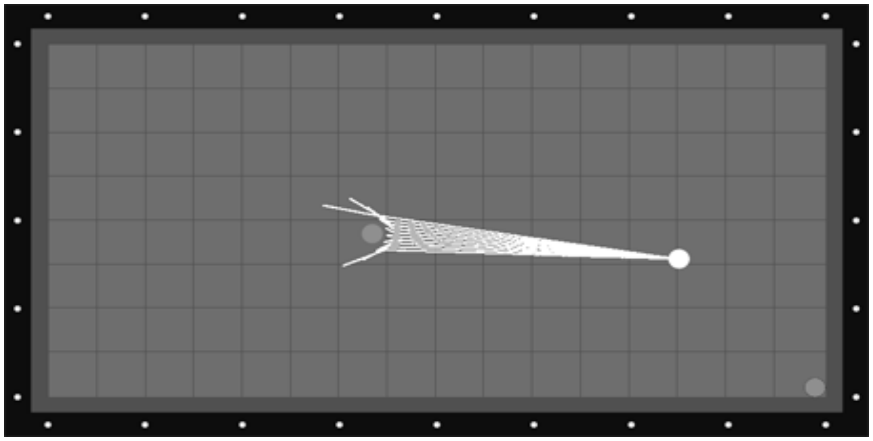


Figure 4.1.2a: 10 steps of ψ

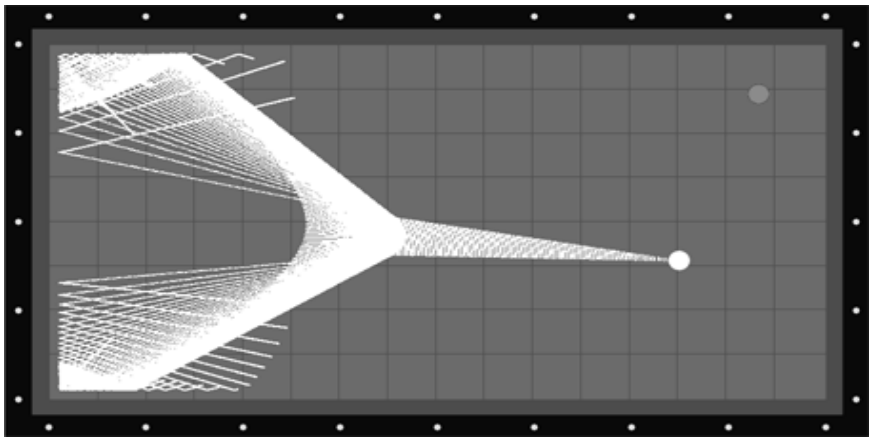
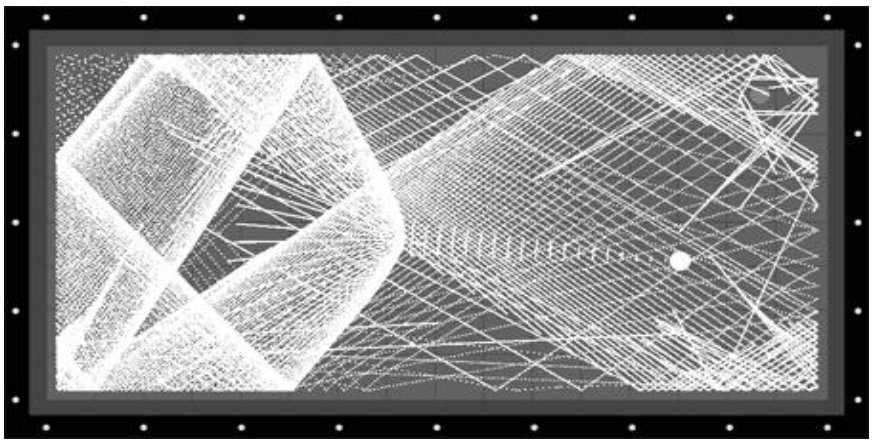


Figure 4.1.2b: 180 steps of ψ, with small initial velocity



Figuur 4.1.2c: 180 steps of ψ, with big initial velocity

Amount of side spin (a) applied

Side spin is an important part of three cushion play. Side spin has little effect on the outcome of a ball-ball collision. In the model it has no effect at all, due to the frictionless treatment. However, equations 2.4.4, 2.4.3 and 2.4.1 show that the spin (ω_Z in the cushion reference frame) induced by a non-zero a does influence the Y-component (in the cushion reference frame) of the outgoing velocity. It is also clear from these equations that ω_Z is of no influence on the X-component of the outgoing velocity. So side spin can be used to influence the outgoing angle of a ball immediately after impact with a cushion.

Figure 4.1.2 shows two different situations. The left situation is called ‘contra-effect’ or contra spin, the right situation ‘mee-effect’. On the left the ball hits the cushion with $\omega_Z < 0$, the right ball with $\omega_Z > 0$. Observing equations 4.3.4, 4.3.3 and 4.3.1, we see that the situation on the left will result in a smaller outgoing velocity in the direction tangential to the cushion (Y-component) when compared with the situation on the right. As the perpendicular velocity is the same for both situations, the translational velocity of the ball directly after impact will be the greatest for a cue ball played with ‘mee-effect’.

As a consequence, playing a ball with ‘mee-effect’ has the advantage that the cue ball can be struck with less initial cue velocity, decreasing the chance of deviation of the input shot parameters due to human error.

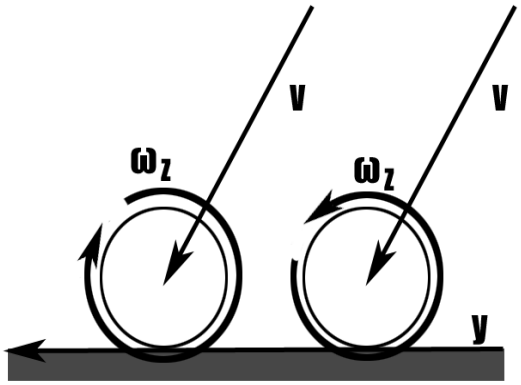


Figure 4.1.2: Spin direction illustrated, view from above. Left is contra-effect, right is mee-effect.

Amount of top/bottom spin (b) applied

Whereas side spin is mainly used to influence the outgoing angle of the cue ball after impact with a cushion, top spin ($b > 0$) or draw ($b < 0$) is used to influence the path of the white after collision with the first object ball, as explained in section 2.1. Draw shots are extensively used in libre, and can be used in three cushion shots. The main reason for using draw or top spin, is to be able to hit the first object ball thicker or thinner, in order to prevent a kiss. However, most shots can be performed without top spin or draw, and although the implemented model can simulate it, it was chosen not to include these kind of spin in the recommendation system.

It could be added easily, but this would add complexity and cost to the shot generation, as well as difficulties to the shot sorting. Therefore it is recommended as future work.

4.1.2 Shot Generation

The shot generation now takes place by combining the different possible values for the input shot parameters and simulate the shots resulting from these parameters. For every simulated shot it is evaluated whether the result is a valid three cushion shot. This is done using a straightforward method. A counter keeps track of the amount of cushion collision events for the cue ball. At the moment the cue ball hits the second object ball, if the number of cushion collisions is 3 or more, the shot is valid, otherwise not. What happens after the collision with the second object ball is irrelevant.

As explained in chapter 3 a trade off was made in the event based simulation regarding collision detection: collision between moving balls is not predicted.

The situation where two moving balls collide before the shot is a valid three cushion is called a kiss, and usually prevents a successful carombole. Therefore the resulting set of shots should be filtered, removing shots where a kiss occurred. At the moment this is established by numerically simulating the solutions, and check for a kiss at each step. It could be useful to simulate the shots again analytically, making use of a quartic solver to detect ball-ball collision. This is expected to be faster and more accurate, however is considered future work.

4.1.3 Clustering of Solutions

Requirement 1.1.4 states that if two solutions are considered the same by an expert, the method should cluster them and select one representative of the cluster.

Experienced billiard players will categorize shot solutions into different types, often giving them names like “lang kort lang”, or “renversé” (for examples of such types see [6]). Therefore it seems intuitively correct to organize the found solutions into clusters, representing variations on a type of shot. The clustering is done in a straightforward manner. The assumption is made that solutions of which the first 4 cushions the white collides with are equal, are perceived as of the same type. The amount of 4 cushions might appear to be strange in the context of 3 cushion billiards. However, it was determined that less meant too little distinction between shots. See for example figure 4.1.3a and figure 4.1.3b. These two are considered to be of different types, but would be clustered together when the criterion would be only the first 3 cushions. More would result in too many clusters. The amount of 4 resembled the way humans would cluster the shots closest.

An example of a solution cluster containing 4 solutions is shown in fig. 4.1.4. (The path of the white is drawn only until the yellow ball is hit, for clarity).

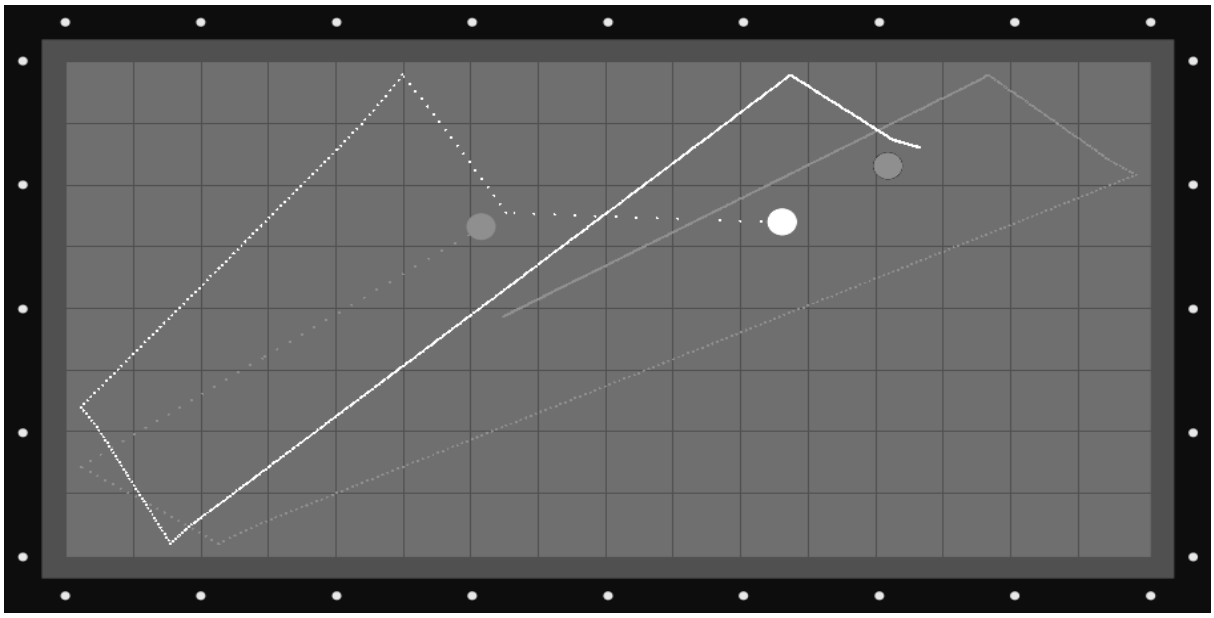


Figure 4.1.3a: Cluster criteria example

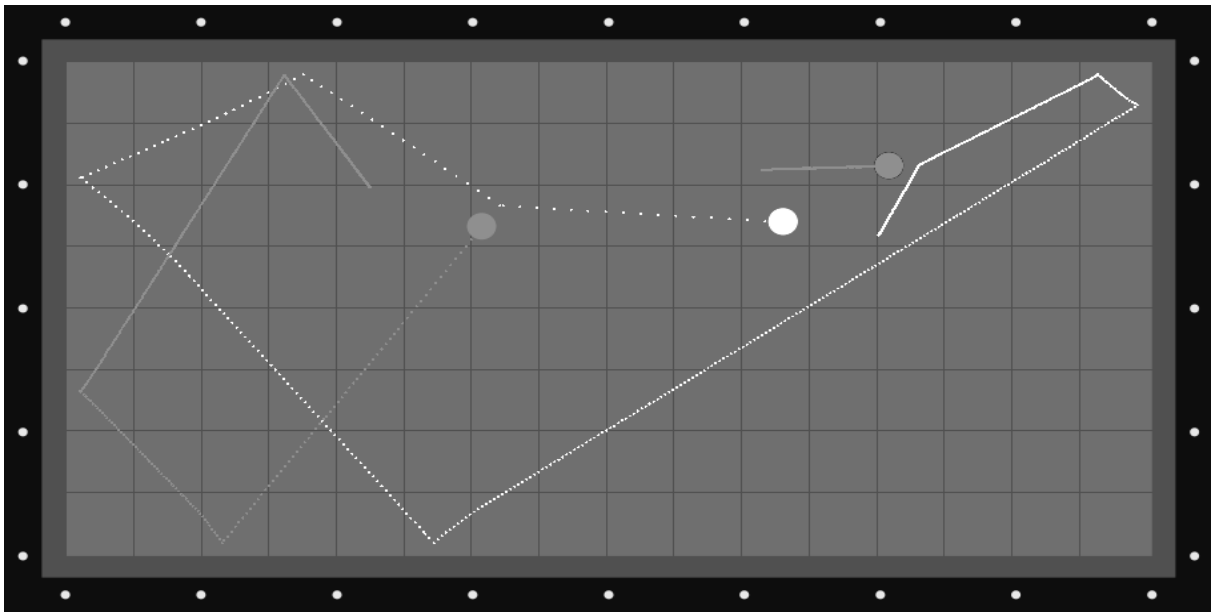


Figure 4.1.3b: Cluster criteria example

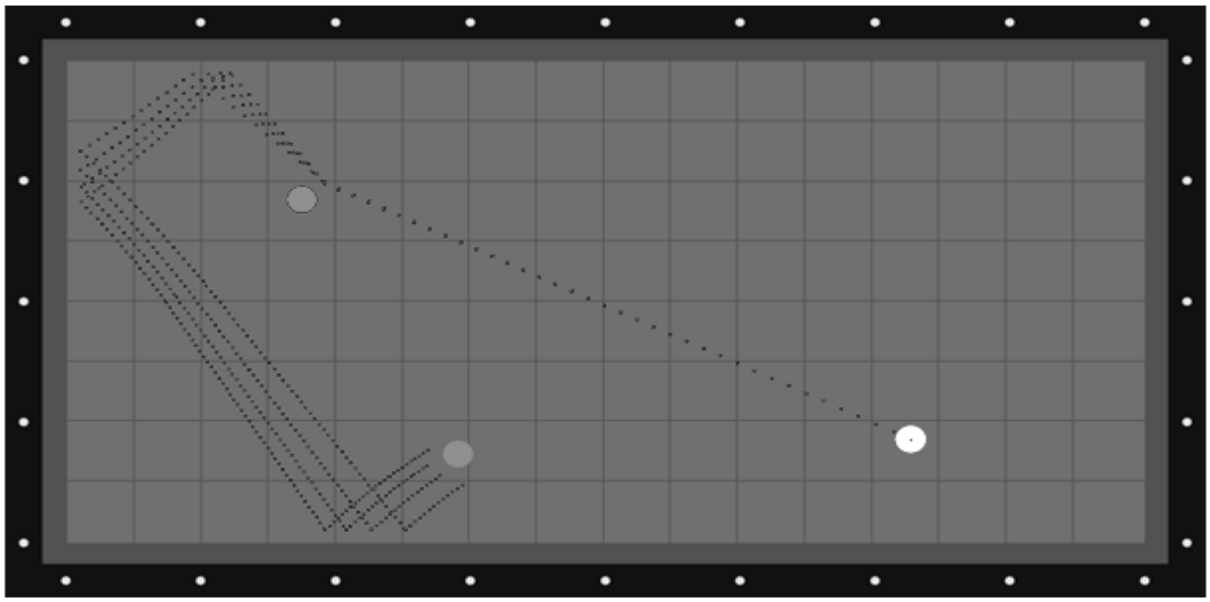


Figure 4.1.4: Solution Cluster example

4.1.4 Evaluation of Solution Generation

Requirement 1.1.2 states that generated solutions should be accurate and reproducible by the user. If this is not the case, i.e. a theoretical solution derived by the method cannot be mimicked by a player on a real table, trust (Req.1.5) would be violated. An evaluation was carried out to check whether the valid solutions found by the shot generation procedure are reproducible in the real life situation. So besides testing the shot generation process, this evaluation also determines if the implemented physical model is accurate enough. The shot generator was configured very basic. This section discusses the experimental setup and the results.

4.1.4.1 Experimental setup

10 random ball configurations were generated. The BC's are fed to the shot generator. The shot generator was also fed with the following combination of shot parameters:

- $V = 3.0$
- $n_{bf} = 90$
- $a = -0.005$ and $a = 0.005$
- b = 0

As can be seen only ψ and the direction of side spin are varied. For the ball-first shots the amount of simulations (i.e. different input shot parameters tuples) now adds up to 360. Cushion-first shots (where yellow and red are less then 25cm apart) were not part of the 10 random ball configurations.

The generated shots were stored. Then for each of these solutions attempts were made by the author to reproduce the path of the white. The amount of random BC's chosen seems quite low. However, for each of the 72 resulting solution clusters an average 10 to 15 real shots were performed, before a conclusion was drawn. Note that it is impossible to exactly reproduce the shot parameters, but the goal of this evaluation was to find out whether it would be useful to show the path the white should travel to a user, i.e. the user can be expected to reproduce this path. Therefore sometimes it took a great amount of attempts before a conclusion could be drawn, as some shots are very difficult to reproduce.

All solutions then are labeled with one of three classifications:

- Exactly reproducible
- Approximately reproducible
- Not reproducible

When not reproducible, a note is made why.

4.1.4.2 Results and Discussion

Appendix E shows the results for the 10 random ball configurations, along with an example solution cluster for each. For the 10 random BC's, the amount of solution clusters found per BC ranged from 3 to 15, averaging 7,2. The sum of all clusters thus was 72. The amount of time needed to generate the solutions ranged from 266ms to 742ms, averaging 530ms.

Of The 72 solution clusters 23 were found to be reproducible. 3 were approximately reproducible, the remaining 46 were not reproducible.

From the notes of the shots that were not reproducible or difficult the following reasons were extracted:

- Contra-spin on the first cushion reacts very different in real life, when compared to the simulated shots. This indicates a flaw in the model. This was the cause for failure in 23 of the 46 not reproducible solutions.
- Prediction of kisses does not work good enough. Kisses caused failure in 10 cases.

Although results seem quite pessimistic, for 9 of the 10 configurations at least one useable and reproducible solution was found. It should also be noted that solutions involving contra spin on the first cushion are very rarely necessary nor preferred during a real billiard match.

This method of generating solutions, with a basic configuration, is found sufficient to find solutions that are reproducible in most cases. Moreover the algorithm is reasonably fast, which is a requirement as the VBA is supposed to be interactive.

4.2 Ordering the Set of Possible Solutions

A real life trainer will figure out what shots are possible, and select a shot from the set of possible solutions for his pupil to carry out. The decision which shot is to be selected is made intuitively, and heuristics-based.

An attempt has been made to find these heuristics, in order to attribute a score to the solutions. This score is used to sort the set of solutions, the shot with the highest score is chosen as the solution the player should be recommended to play.

This section first describes a simple sorting algorithm, using only solution cluster size. An evaluation of this baseline algorithm is carried out on several random ball configurations. The same ball configurations, and their solutions, are shown to an expert, who is asked to comment on the solutions and also sort them. Results are compared. Based on the results, the sorting algorithm is improved.

When the prototype of the VBA was finished it was tested during a real training session, by comparing the solutions recommended by the trainer to the ones found and recommended by the VBA. This evaluation is discussed in section 4.2.3.

4.2.1 Sorting Solutions Using Cluster Size

Whereas the physical model implementation can apply shot parameters exactly, a shot carried out by a human player will always introduce inaccuracies. Therefore a solution with a bigger margin of error for the shot parameters would be more preferable.

Conveniently, the method of clustering solutions offers an obvious way to detect the margin of error for the parameter psi. The bigger the cluster size, the bigger the margin of error is for the center solution, as can be seen in figure 4.1.3, where the white can hit the first cushion with a margin of approx. 12cm., while still producing a valid 3-cushion shot. It is expected that this margin of error will be the most important factor for a human trainer to decide which shot is selected as most favorable for his trainee.

So now solution clusters are sorted according to their sizes. From each cluster the center solution (in terms of ψ) is picked to represent the solution cluster.

4.2.2 Evaluation of Shot Recommendation

The goal of the shot solution finder is to recommend a solution that also would be recommended by an expert. This would be the top solution from a sorted solution set. To evaluate the simple sorting method, as well as the solution finder in general, an expert human trainer was consulted.

13 random ball configurations were presented to the expert. For every BC the solutions were generated, yielding solution clusters. The parameters for the shot generator are the same as for the evaluation as discussed in 4.1.4. Out of every cluster the middle solution is selected, according to the protocol. The set of solutions for a BC is presented, in random order, to the

expert. The expert then is asked to select, rank and motivate a top 3 of solutions, and assert whether the set of solutions is complete.

The rankings as provided by the expert are compared to the results of the automated sorting using the baseline protocol.

General remarks are also noted. For example, it could be that the expert disagrees with the reproducibility of a certain solution.

The set of 13 BC's is stored, to help find a better sorting algorithm if needed.

Results and Discussion

Table 4.2.1 shows for every ball configuration the top 3 as determined by the expert (2nd column), as well as the top 3 based on cluster size (3rd column). The numbers shown are indices of solutions. When the expert desired a solution that was not part of the shown solutions, this is indicated by an x or y. For BC 3 and 4 only 1 solution was found appropriate by the expert.

In only 1 case the solution selected by the expert was the same as the solution that surfaced having the biggest cluster size.

For 5 BC's the best solution according to the expert was not found by the algorithm. For 4 of these 5 BC's the solution ranked second best by the expert was found.

Although with a sample size of 13 random ball configurations is quite small, results seem to indicate that for most ball configurations the shot solution finder is able to find a solution that would be advised as best or second-to-best by the expert (only for BC 7 a solution found by the method is ranked 3rd).

BC	Expert	Cluster Size
1	8,1,4	1,8,9
2	0,5,x	2,3,1
3	3	0,2,3
4	x	0
5	24,7,x	23,11,29
6	x,11,2	4,?,2
7	x,y,2	0,1,2
8	3,2,4	2,4,0
9	4,1,0	0,4,1
10	x,1,3	3,4,2
11	x,0,y	0,1,2
12	0,1,x	0,1,3
13	15,9,11	1,20,9

Table 4.2.1: Results expert evaluation

Analysis of Motivation

The rankings in table 4.2.1 provide no information about the motivation behind the expert's choices. During the evaluation when the expert studied the set of solutions and ranked a top 3, every individual solution was discussed, as well as the decisions made when ranking.

This provided information that is useful to improve the solution finding process.

Contra-spin

As expected, because shots with contra-effect are not simulated accurately, solutions having contra-effect are impossible to carry out on a real table. This was confirmed by the expert in all cases. However the expert agreed that BC's where a solution using contra-effect would be best are rare, with the notable exception of a type of shot called renversé, illustrated in figure 4.2.1.

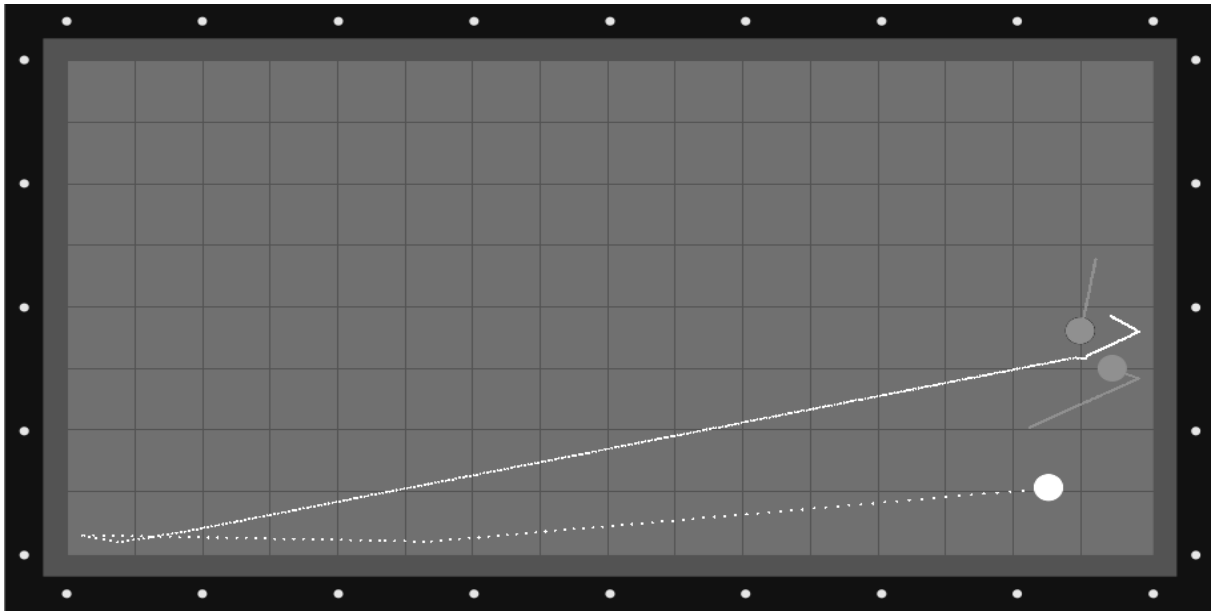


Figure 4.2.1: Renversé

"Using width"

Another motivation for preferring one solution over another was often that "it is preferred to use the width of the table". Figure 4.2.2a shows such a solution, whereas figure 4.2.2b shows a solution "using the length". It appears that a solution where the cue ball firsts hits a long cushion, then a short and then a long cushion again is preferred above solutions first hitting a short cushion, then a long, then a short.

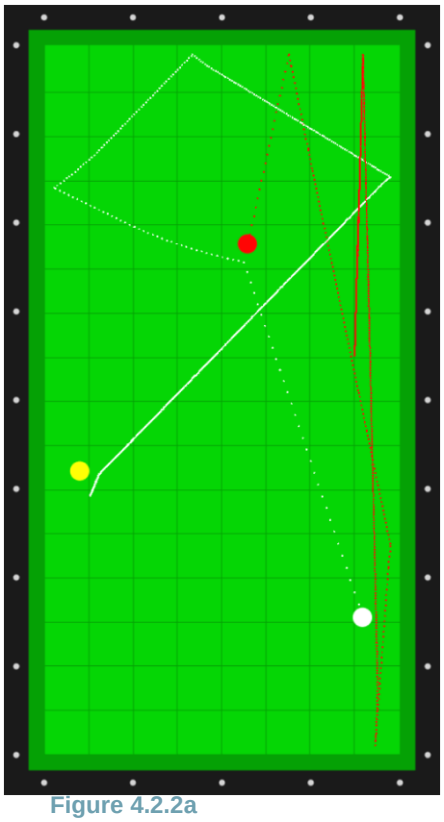


Figure 4.2.2a

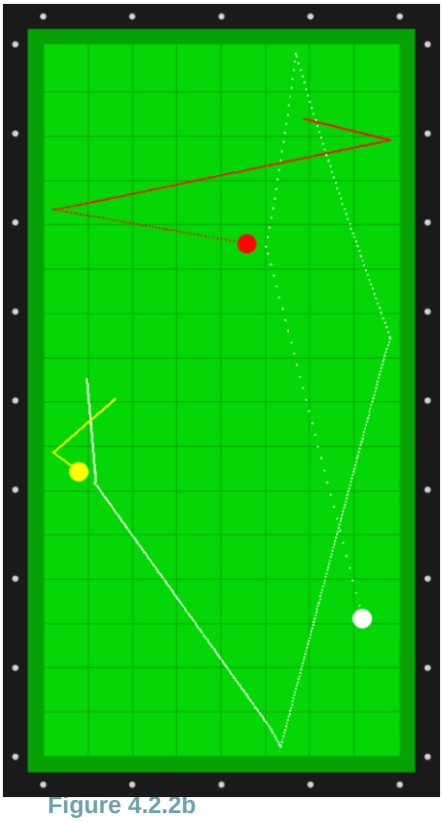


Figure 4.2.2b

Thickness

The term thickness (dutch: aanspeeldikte) describes the angle between the line defined by the centers of the CB and OB1 and the aiming direction ψ (visualized in figure 4.2.3a and 4.2.3b).

When a player aims a ball, he does so by focusing on an aiming point. Solutions that involve a very thin contact are difficult for players to realize because the aiming point is not on OB1, but 3 cm away from the center of OB1. Therefore an aiming point on a ghost ball has to be visualized by the player.

Solutions involving a thick contact also pose difficulties, as a very thick contact will result in a complete transfer of translational velocity of the CB into OB1, due to laws of impulse. The angular velocity of the CB then takes over. This raises several difficulties:

- The ball has to be struck with relatively great force
- The amount of spin applied to the ball becomes crucial
- Differences between the physical model and the real table are more evident

The VBA should advice solutions involving extremely thin or thick contact only when no other suitable solutions are found.

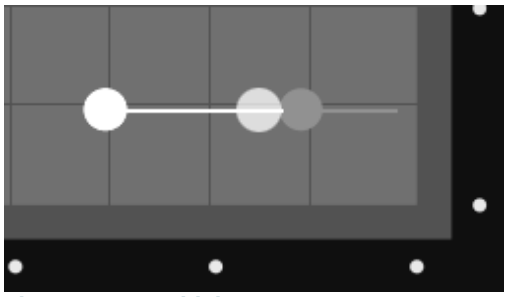


Figure 4.2.3a: Thick Contact

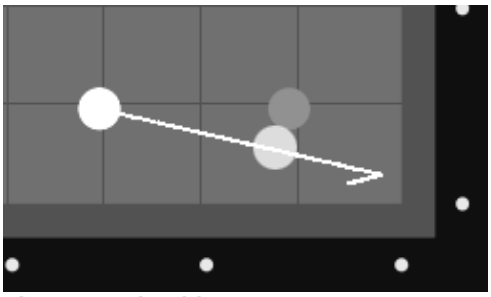


Figure 4.2.3b: Thin Contact

The ‘thickness’ of a solution was only mentioned as a criterion by the expert in extreme cases. The need to play extremely thin or extremely thick normally would make a solution non-preferable.

Strategy

As producing successful shots in three cushion alone is conceived very difficult for a player, strategy involving the possible next moves in a game is mostly only used by advanced players. However, a rule of thumb is to play a shot at such a pace that the white ball will not roll too far after hitting OB2. Offensively this implies that the cue ball will be left in close range of at least one object ball, making the next shot easier to aim (as explained in 2.1.4).

When the red is used as OB2 playing the shot at just the right pace has the advantage that the CB will be left in the neighborhood of the red ball. When the shot is just missed, the opponent will be faced with his two object balls positioned close to each other, which in most cases is conceived as a difficult to solve BC.

As the outcome of a shot will vary widely, due to inaccuracies in shot parameters as well as the simulation using the model, looking further ahead than one shot is not considered useful.

In some cases, where multiple solutions where found more or less equally preferred, strategy was considered by the expert. This was made clear with statements like “when a match is in its final stage, and your opponent is almost finished, it would be better to play defensively towards the red ball”.

Conclusions and improvements

Contra-effect solutions are mostly false, and not useful. However only in some rare cases not using them would lead to a sub-optimal solutions. Thickness only acts as a disqualifier, when a shot has to be played extremely thick or thin. Shots that first hit a short, then a long, then a short cushion are preferred over shots that first hit a long, then a short, then a long cushion. Strategy is only used in specific situations during a match.

The motivations were used to implement a new scoring method, which is used to sort the set of solutions:

1. Solution clusters that require contra-effect on the first cushion are discarded.
2. The solution clusters are sorted according to their size. Then they are awarded a score. When there are n clusters, the cluster with the biggest size gets n points. The next biggest cluster gets $n - 1$ points, and so on. Clusters of the same size get the same amount of points awarded.

3. When the first three cushions of a solution in the cluster are respectively a long, a short and a long cushion, two points are added to the score.
4. The thickness of a shot is expressed as a number. Extreme thin or thick has value 0, exactly between thin and thick has value 1. As explained earlier, extreme thick or thin solutions are not preferred. Therefore solutions with a thickness value < 0.05 are punished. One point is subtracted from the score. In a cluster, the thickness value of the center solution is chosen.
5. Finally strategy is considered, playing towards the red ball is awarded with 0.1 bonus point. The value is small so that it can only make a difference between two solutions that have the same score.

The resulting algorithm was used to order the solutions for the ball configurations found during the evaluation again. The results are shown in table 4.2.2.

BC	Expert	Cluster Size	Improved
1	8,1,4	1,8,9	1,8,9
2	0,5,x	2,3,1	3,1,0
3	3	0,2,3	0,3,-
4	x	0	-
5	24,7,x	23,11,29	
6	x,11,2	4,?,2	
7	x,y,2	0,1,2	0
8	3,2,4	2,4,0	3,5,2
9	4,1,0	0,4,1	0,4,1
10	x,1,3	3,4,2	2,3,1
11	x,0,y	0,1,2	0,2,-
12	0,1,x	0,1,3	0,1,3
13	15,9,11	1,20,9	1,20,3

Table 4.2.2: Results 'improved' sorting method

The sorting is slightly improved, but still far from perfect, according to the data in table 4.2.2. These suggest that work should be done to find a better way to mimic the decision making process of a human expert trainer. It was decided to keep the sorting method as it is, and focus on the other aspects. The evaluation discussed next, which is different in setup, was performed in a later stadium and yields significantly different results.

4.2.3 Second Evaluation of Shot Recommendation

In the final stage of the project, when the VBA prototype was finished, it was decided to evaluate how the VBA's shot recommendation would compare to a human expert in a real world situation. An evaluation was performed during a training session with two of the trainer's pupils. The training session was a typical training session, where the two players play freely, sometimes choosing their own solutions, often discussing them with the trainer and sometimes asking the trainer for advice.

During this process, the VBA was requested to come up with solutions for every ball configuration encountered by the players. This solution was neither shown to the player, nor to the trainer. The solutions that were preferred by the players/trainer were compared to the solution recommended by the VBA. Table 4.2.3 shows the results. The columns with header BC contain the index number of the ball configuration. The columns with header played

shows what the ranking of the solution recommended by the human trainer to carry out was according to the VBA. A hyphen ('-') indicates that the solution recommended by the trainer was not found by the VBA.

Note that the big difference in setup when compared to the evaluation discussed in 4.2.2 is that for the latter the expert was shown a set of solutions, represented in a 2d visualization, which he was asked to sort and add solutions to when needed. In this evaluation the trainer does not see the set of solutions, and sees the ball configuration in 3d, like he is used to.

BC	Played	BC	Played	BC	Played	BC	Played
1	1	**12**	1	**23**	-	**34**	1
2	1	**13**	1	**24**	-	**35**	-
3	1	**14**	1	**25**	-	**36**	-
4	-	**15**	-	**26**	-	**37**	1
5	1	**16**	2	**27**	-	**38**	1
6	-	**17**	1	**28**	-	**39**	1
7	-	**18**	6	**29**	-	**40**	1
8	-	**19**	-	**30**	2	**41**	-
9	1	**20**	-	**31**	-	**42**	1
10	1	**21**	-	**32**	1	**43**	-
11	-	**22**	-	**33**	2	**44**	3

Table 4.2.3: Results VBA vs. human expert trainer

As shown in table 4.2.3, for 22 of the 44 ball configurations encountered the VBA actually found the solution that was recommended by the trainer. Furthermore, for 17 of these 22 cases the solution recommended by the trainer was also the top recommendable solution according to the VBA.

Looking at the BC's where no solution was found, several causes (not in the table) were found. For 2 cases the ball positions could not be determined by the VBA. 3 times the trainer recommended a solution where contra-spin on the first cushion is required. In 11 cases the VBA was unable to find a solution.

Interesting is the gap from BC 19 to BC 29. The trainer agreed that the ball configurations encountered there were very difficult to solve. The players were in a situation where one difficult to solve ball configuration missed by one player results in a difficult to solve ball configuration for the other. The exact reason in this case was not noted, but often these situations occur when the red ball is situated against a cushion.

The observation that BC's that are perceived as difficult by the expert are also difficult for the VBA to solve is interesting, and could explain part of the difference in results for this evaluation, when compared to the evaluation in section 4.2.2. The ball configurations in 4.2.2 were completely random, and relatively few. During the evaluation in 4.2.2 the expert also hinted that the random generated ball configurations were relatively difficult to find a feasible solution for, when compared with typical configurations encountered during a real match.

What also contributed to the better results is that the set of values for V as input for the shot generation was extended. Shots were generated for $V = 3.0\ m/s$, $V = 3.5\ m/s$ and $V = 4.0\ m/s$, leading to more solutions, and bigger clusters.

To conclude the chapter we will take a look at the requirements that were posed regarding the solving of shots. The requirement that *the method to generate the shots should be reasonably fast* is met, as shots at most take a couple of seconds to be solved.

The requirement that *solutions that are generated should be accurate, i.e. reproducible by a user on a real table*, is fairly met. This is shown by the evaluation in paragraph 4.1.4.2. The biggest violation of this requirement was caused by solutions requiring contra-spin on the first cushion. Therefore it is decided that these type of solutions will be ignored. Although these type of shots sometimes are feasible solutions, it does not occur very frequent. Another problem concerning reproducibility was that the prediction of kisses sometimes failed. This can be attributed to inaccuracies in the physical model.

The requirement that all solutions that *an expert would assess as reasonable for a particular ball configuration should be present* is covered in the evaluation in section 4.2.2. As can be deduced from table 4.2.2 in the first column, for 8 out of 13 ball configurations the solution appointed as the top solution was found in the set of generated solutions. For the remaining 5 ball configurations, the solution according to the expert that would be considered second best was generated in 3 cases. In 1 case the third best solution was generated, and only for ball configuration 4 no solution was generated that was assessed as being recommendable by the expert.

The requirement that *if two solutions are considered the same by an expert, the method should cluster them and select one representative of the cluster*, cannot really deduced by the data from table 4.2.2 or 4.2.3. However, from the comments made by the expert during the evaluation (and own observations during the project), it was determined that the clustering method based on the first 4 cushions was good enough.

Requirement 1.2.1 stated that: *one solution should be recommended as the best one. The solution is considered best, when an expert trainer would recommend the same solution, when he is presented the same ball configuration.* Table 4.2.2 shows that this was the case for one ball configuration (#12) using the basic sorting method, and for two ball configurations (#8, #12) using the improved method. However the second evaluation, discussed in section 4.2.3 yielded better results, 17 out of 44. Because the VBA's top recommendation is not always the best, it was decided that the VBA should enable the user to select an alternative.

Requirement 1.2.2 stated that: *The VBA should be able to present good alternatives, if the user desires. An alternative solution is considered good, when an expert trainer would agree to the alternative being a plausible solution.* Table 4.2.2, column 1, shows that only for ball configuration 4 no good alternative was found. The others all had an alternative that also showed up in the top 3 expert solutions. This shows that when the VBA's top recommendation is not desired by the user, feasible alternatives are present in the solution set.

Concluding, the method proposed in this chapter was found suitable to use for computing solutions for any random ball configuration encountered, and is considered to answer research question 1: *How should the VBA calculate a solution to recommend to a player for any given ball configuration?*

5. Assisting Players

In the previous two chapters a method was described to enable the VBA to process an input ball configuration, and output a solution that should be recommended to a user. This chapter describes how the VBA could interact with a real life situation and user. First the problem of detecting what ball configuration is present on the table at a given moment is discussed in 5.1. The next section, 5.2, discusses how the found solution could be presented to the user. This presentation method is finally tested in a real life situation, where the possibilities of interaction between user and VBA are explored. The setup of this experiment and the results are shown in section 5.3.

5.1 Inputting Ball Configurations

When a user requests a solution for a ball configuration, he must present this ball configuration to the VBA. One way to present the ball configuration to the VBA could be by means of manual input. The user determines where on the table the balls are positioned, and feeds the positions to the VBA.

Another method would involve the use of a webcam and image processing techniques, enabling the VBA to acquire the ball positions upon request without the user's help.

As the requirements 2.2.1 and 2.2.2 state, the method for inputting ball configurations should be both accurate and non-obtrusive. The second proposed method, automatic acquisition of ball configurations using computer vision techniques obviously meets these requirements better then the manual solution. Fortunately, due to the simple shapes and contrasting colors, retrieving ball configurations is a fairly straightforward image processing task. The method used is discussed in this section.

Ideally a capture device should be placed above the table, offering a top view. However, the accommodation[5] where the table is located does not have a ceiling high enough to enable a complete view of the table surface from a mounted camera (with a normal lens). The current solution uses a HD webcam, taking snapshots at a resolution of 1600x1200 pixels. The webcam is mounted on a wall behind the short cushion. The method to retrieve a ball configuration from a captured image is implemented using the image processing toolkit of Mathworks Matlab.

Throughout this section, the position (in cm.) of the balls on the table are called real world positions. The positions (in pixels) of the balls in the captured image are called image positions. The positions of the ball's determined using the discussed method (in cm.) are called acquired positions.

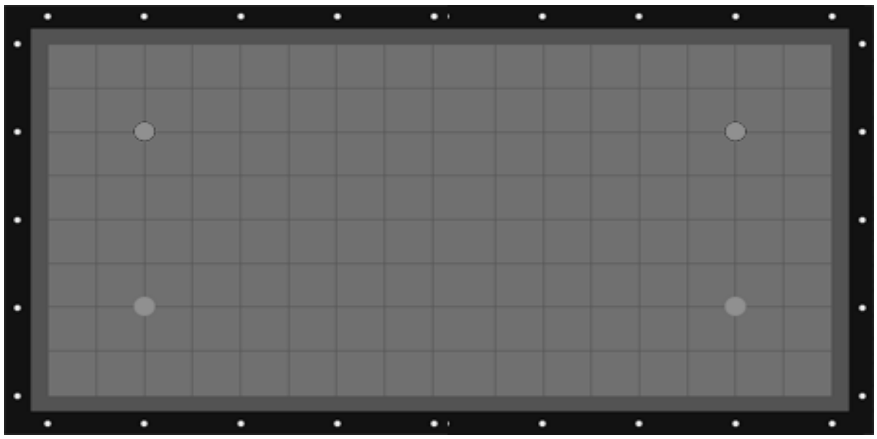
5.1.1 System Calibration

Before detection can take place the system is first calibrated. 4 balls are placed on the table cloth, at the points shown in figure 5.1.1, which are easy to find using the diamonds. The coordinates of these real world positions are known in cm., clockwise starting at the upper left position: (28.75,28.75), (28.75,201.25), (86.25,201.25), (86.25,28.75).

[5] Clubroom of BSV de Stoottroepen, Bastille 201, University of Twente Campus

Now a snapshot of the table is taken (figure 5.1.2a), and a region of interest is selected manually (figure 5.1.2b). Furthermore, a region of the cloth is selected, used to determine its color (by averaging the hue of all pixels within the region).

For the last step of the calibration, the positions of the 4 balls are determined following the method described in 5.1.2. By using Mathlab's *cp2tform* function, a transformation matrix is constructed, with the known real world positions and the image positions as input. This matrix is later used to transform coordinates from image positions to acquired positions during BC detection.



Figuur 5.1.1: real world positions for calibration

Figuur 5.1.2a: Snapshot taken by webcam, for calibration purposes

Figuur 5.1.2b: Region of Interest selected

5.1.2 Acquiring Ball Configuration

Once the system is calibrated, ball configurations can be retrieved. The process from a users request to an output set containing the positions of the balls within the table frame that can be used as an input for the VBA will be described here briefly.

First an image is captured (fig. 5.1.3a). The image is converted from RGB color space to HSV. HSV is a more convenient color space, in which pixels are attributed values for hue, saturation and value. Only the hue information is used (fig. 5.1.3b). During the calibration an average value for the hue of the table cloth was collected. Now for every pixel's hue value the distance from this average value is calculated (fig. 5.1.3c). The next step is to blacken out the scenery outside the region of interest marked during calibration (fig.5.1.3d). Every pixel with a hue value smaller than 0.2 is now set to 0, the remaining are set to 1 (fig.5.1.3e), resulting in white circles where the balls are situated.

The resulting binary image is passed to Mathlab's *bwboundaries* function, which traces boundaries of objects in an image. This results in an array of regions, for which several properties are available. One of these properties is the smallest bounding box surrounding each region. This bounding box is used to:

1. Remove noise by looking at the width, height and diameter of the regions, filtering out regions that are too small or big to be a ball
2. Find the center of the ball (fig.5.1.3f)

Now the positions of the centers of the three balls in the image are known, their real positions can be estimated by using the transformation matrix established during the calibration.

The last step is to determine the colors of the found balls. This is done by calculating the average hue of all the balls' pixels. The ball with the highest average hue (in the example ~0.91) is the red ball, the lowest (~1.6) is the yellow, the remaining (~1.9) the white. Although the difference in hue between yellow and white is small, during the user tests the ball colors were determined correctly almost every time. The few times the yellow and white got confused might be corrected easily, for example by using different lighting settings.

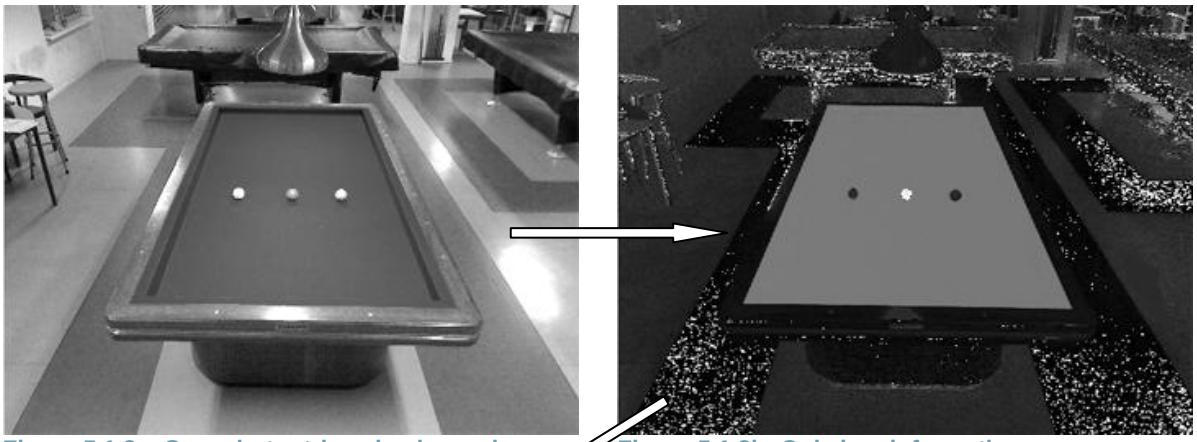


Figure 5.1.3a: Snapshot retrieved using webcam

Figure 5.1.3b: Only hue information

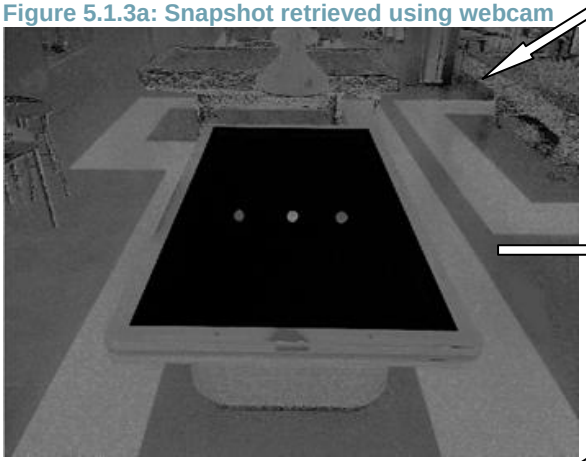


Figure 5.1.3c: Distance from average hue of table cloth

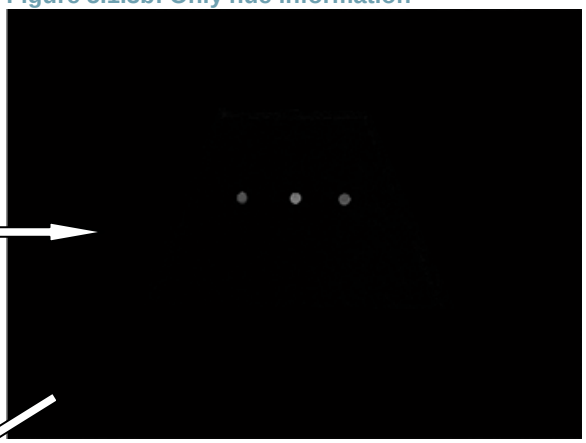


Figure 5.1.3d: Information outside region of interest discarded

Figure 5.1.3e: Thresholded to binary image

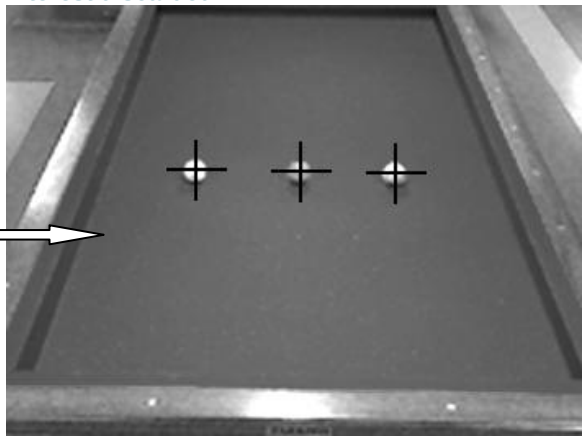


Figure 5.1.3f: The coordinates of the centers are determined, illustrated with a cross

Figure 5.1.3a..5.1.3f: The steps for acquiring a ball configuration in chronological order

5.1.3 Evaluation of Detection Method

In order to assess the accuracy of the ball position acquisition method a simple evaluation was carried out. Using the table's diamonds, 21 balls were manually placed at known real world positions. Then a snapshot was taken and acquired positions were determined using the described method. The results are shown in table 5.1.1. The left two columns represent the position at which the ball was placed, the right two show the position according to the method. The error is the distance between the real world position and the acquired position, in cm.

The average error in this case is 0,84 cm.

Y (cm) *Real*	X (cm) *Real*	Y (cm) *Acquired*	X (cm) *Acquired*	Error (cm)
28,75	201,25	28,6703	200,4476	0,806348
28,75	172,5	27,6531	171,8744	1,262761
28,75	143,75	27,8779	142,3404	1,657568
28,75	115	28,1276	114,0722	1,117226
28,75	86,25	28,7487	85,8636	0,386402
28,75	57,5	29,1366	56,7327	0,859191
28,75	28,75	28,6714	28,7471	0,078653
57,5	201,25	57,2521	200,4408	0,846321
57,5	172,5	57,2741	171,7064	0,825125
57,5	143,75	57,9968	142,6518	1,205344
57,5	115	58,1758	113,9777	1,225481
57,5	86,25	58,1225	85,8976	0,715327
57,5	57,5	57,8913	57,0496	0,596637
57,5	28,75	57,5414	28,883	0,139295
86,25	201,25	86,0857	200,8453	0,43678
86,25	172,5	86,0545	171,4992	1,019716
86,25	143,75	86,072	143,7381	0,178397
86,25	115	87,5647	114,5925	1,376406
86,25	86,25	84,7089	86,0185	1,558391
86,25	57,5	87,3112	57,0612	1,148343
86,25	28,75	86,3127	28,6202	0,14415

Table 5.1.1: Results for evaluation of acquisition method

5.1.4 Discussion

The method used is not perfect, but found suitable. The biggest disadvantages are:

1. The player should not be in the field of vision when a snap shot of the table state is taken
2. Balls positioned near each other, where one overlaps the other from the webcam's point of view, are at the moment irretrievable.

The former can be solved by warning the player, the latter by using a camera mounted above the table.

Requirement 2.1.1 dictated a maximum error of half a ball (3 cm.). Table 5.1.1 shows all results in the test are within this margin.

The webcam is firmly mounted, so that it will not move during experiments. Therefore recalibration is not necessary.

5.2 Demonstrating Solutions

The VBA will process incoming ball configurations, and output a set of solutions, which are represented by shot parameters [a, b, V, ψ]. In order to produce a valid three cushion shot, the user should reproduce these parameters on the real table as accurately as possible. This requires the VBA to visualize the shot parameters in a comprehensible way.

This section thus deals with research question 2.2: *How to present a solution to a user?*

This research question is quite broad, and must be further specified. This chapter will explore a method of presentation, which is used to explore the VBA's possibilities. Therefore we will

pose more specific questions, for which answers will be sought in a series of user tests. The questions are:

- **Does the user consider a VBA useful?**
  Although expected, it is not sure whether billiard players consider the possibility of digital assistance for improving billiard skills to be useful at all. It would be good to know whether users see an added value in a VBA.
- **How does the user use the VBA?**
  The current setup encourages the player to freely use the VBA however it suits his needs. It could be interesting to observe how individual players (would like to) use the VBA's services.
- **Does the user trust the VBA's advice?**
  Obviously trust is an important issue when it comes to advice. Whether or not users trust the opinion of a VBA should be analyzed.
- **Is the current, basic presentation method sufficient to communicate solutions?**
  As explained, users will never be able to exactly reproduce the shot parameters provided by the VBA. But we can observe and ask whether the current presentation method is clear enough.

The main requirements for the presentation layer are: *the shot parameters should be presented in such a way that the user is able to reproduce a shot accurately*, and *the user should be able to interact freely and intuitively with the VBA, he should not feel restricted by the interface.*

Keeping in mind these two requirements 5.2.1 discusses the method of presentation, 5.2.2 discusses how the user can interact with the VBA.

5.2.1 Visualization of Shot Parameters

The methods of presenting shot information to a user described in [3] and [2] (see literature review in section 1.3) are interesting, but considered outside the scope of the project. Billiard instruction books, like [6], usually show solutions to three cushion shots in a 2d diagram, showing a table view from above, with the three balls and the path the white should travel. It was chosen to implement this method of visualizing solutions, with some 'extra features', to communicate the shot parameters to the user.

A 3d model of a billiard table was implemented, of which a screen shot is shown in figure 5.2. The 3d model is used to show the shot parameters [a, b, V, ψ] that define a solution in such a way that they can be reproduced by the user. This section describes and discusses the choices made on how to communicate the parameters.

ψ

The aiming parameter should be immediately clear to the user by the white line originating from the white ball. A 3d model of a billiard cue could be added to indicate the aiming direction, but was considered superfluous.

a, b

The parameters a and b are visualized by using a front view of the white ball, with a blue dot to indicate where the cue should make contact upon impact (see figure 5.2, upper right

corner). It should be noted that during the tests, the solution finder was configured such that $b = 0$ at all times, only side spin was generated. Furthermore, the generated shots used either $a = 0.005$ or $a = -0.005$. Because the model 'produced' stronger side spin than a real player, these values for a should be greater on the real table. Therefore the horizontal position of the blue dot is exaggerated.

V

The cue velocity upon impact is implied by the distance the white has to travel to successfully carry out the shot indicated by the path. It was decided not to explicitly show this velocity. The reason for this is that it cannot be expected from a user to reproduce an exact quantitative indication of velocity, for example $3.5\ m/s$.

5.2.2 Interaction

Interaction with the VBA takes place using keyboard and mouse controls. When the user needs advice for a ball configuration, he can request a solution by pressing the space bar.

The ball configuration is acquired when possible, and the VBA will search for solutions according to the method discussed in chapter 4. The top solution will be presented first, visualized by a white line, that depicts the path the white ball should travel. The user can now navigate, using the mouse and keys (w,s,a and d), in the same way one navigates through a first person shooter game.

By pressing the return key, the shot will be demonstrated by animation. On the top left the amount of solutions found is displayed, as well as the current solution shown. By pressing the keys N(ext) or P(revious), the user can navigate through the solutions. The top right shows a cue ball, with a blue dot indicating the amount of English spin the user should apply. See figure 5.2 for a screenshot.

When the user requests a solution, a message is shown reporting that the VBA is searching for a solution. When a solution is found, it is shown. When searching for a solution fails, this can have two different causes:

1. The ball configuration could not be retrieved
2. The ball configuration could be retrieved but no feasible solution was found

Which of the two causes occurred is shown to the user, also via a text message.

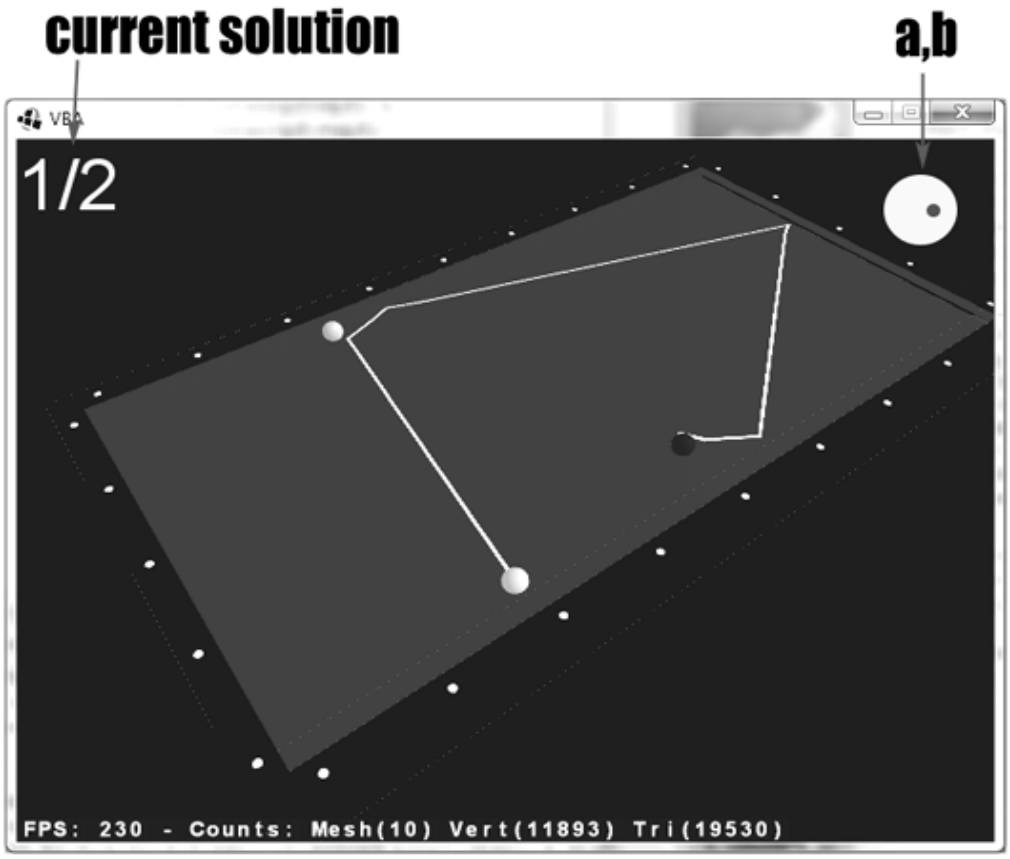


Figure 5.2: : Screenshot of application for visualization and demonstration of solutions

5.3 User Tests: System Evaluation

Now that a means of presentation and interaction between the VBA and the user has been established, the complete system can be evaluated.

The ultimate question that should be answered would obviously be whether a player's billiard skills would improve more and faster with the use of a VBA, compared to players that use other techniques of improving their skills, for example taking lessons from a human expert trainer or self-learning by book. However the learning effect is impossible to measure within the limited time span, because it would require following progress of a large amount of players over the duration of at least a year to have any significance. What can be explored in the evaluation instead are the potentials of the VBA as an assistance tool, by means of a user test, involving a few target users. Then an in-depth analysis on how they use the VBA can be performed.

The players participating in the user test all fall within the category of target users, discussed in 1.1.3. This implies that they have a decent amount of technical skills, and are thus able to more or less successfully reproduce a shown solution. Although the successfulness of a shot carried out by the player is observed, it cannot be taken into consideration in assessing the quality of the whole system, as it is too much a random factor.

As stated before, for the final user tests we will focus on the following questions:

- **Does the user consider a VBA useful?**
  Although expected, it is not sure whether billiard players consider the possibility of digital assistance for improving billiard skills to be useful at all. It would be good to know whether users see an added value in a VBA.
- **How does the user use the VBA?**
  The current setup encourages the player to freely use the VBA however it suits his needs. It could be interesting to observe how individual players (would like to) use the VBA's services.
- **Does the user trust the VBA's advice?**
  Obviously trust is an important issue when it comes to advice. Whether or not users trust the opinion of a VBA should be analyzed.
- **Is the current, basic presentation method sufficient to communicate solutions?**
  As explained, users will never be able to exactly reproduce the shot parameters provided by the VBA. But we can observe and ask whether the current presentation method is clear enough.

To find an answer to these questions two methods were used for evaluation. The player was asked to play three cushion billiards and meanwhile interact with the system. His actions were observed and annotated. After the player finished, a questionnaire (table 5.2) was filled in.

First a preliminary user test was performed, discussed in 5.3.1. This preliminary test was used to fine tune the system and to setup the real user test. The setup of the final user tests is discussed in 5.3.2, followed by results in 5.3.3 and discussion in 5.3.4.

5.3.1 Preliminary User Test

Before the user tests were setup and conducted, a preliminary user test was conducted. The preliminary user test was orientational, so the results were complete nor very useful for answering the research questions. The results were used however to fine tune the real user test, and to implement some improvements:

- As the process of acquiring ball positions and solving the configuration spans a couple of seconds, a sound was added (on the user's request) to indicate when the shot is solved and displayed. Instead of staring at the screen waiting for a solution, the user can now cast a look on the table, and is notified when a solution is presented.
- The initial position of the viewport in the simulation was altered, so that it matches the view from the user when he turns his head to look at the real table situation. This makes it easier and more intuitive for the user to see how the solution should be transferred from the simulation to the real situation.
- The time between a request and the webcam taking a snapshot is relatively long, around 5 seconds. This is a hardware issue and could not be improved. Therefore it was decided that, while the user already would have to wait a considerable amount of time, the extra cost of generating and simulating more shots would be relatively small. Therefore the set of values for V as input for the shot generation was extended. Shots were generated for $V = 3.0\ m/s$, $V = 3.5\ m/s$ and $V = 4.0\ m/s$, leading to more solutions, and bigger clusters.

- A technical issue caused the system to crash after a while during the test. The issue was resolved, during the real user tests the system remained stable.

The setup and results of the preliminary user test can be found in appendix F.

5.3.2 User Tests Setup

The first 25 minutes the user could freely use the VBA however he liked. During this time he was observed, and his actions were annotated. The only instructions given to the user were how to control the VBA, and not to stand in front of the webcam.

For the second session of 25 minutes the user was provided with some instructions, to simulate a more guided training methodology. The idea for the more structured session arose after the first user test. After 50 minutes the player got bored, and seemed to care less about what the VBA had to say, only casting a brief look every time. The instructions the other users got for the second session:

1. Place a ball configuration that you would like to practice
2. Mark the positions of the balls with a piece of chalk
3. Find your own solution for the ball configuration and play it accordingly, as many times as you like
4. Use the VBA to solve the shot, use the results, and keep repeating the shot until satisfied

For every BC during the first session the following properties and actions are annotated:

- Does the user ask the VBA for a solution?
- Which solution does the user choose?
- How many solutions are there for this BC?
- Does the user browse through the solutions?
- Does the user successfully carry out the shot?
- Does the user use the possibility to watch an animation of the shot?
- During the second session the player first comes up with his own solution, before using the VBA. Is the user's own solution also represented in the set of solutions offered by the VBA?

During the second session the following properties and actions were annotated:

- Does the user ask the VBA for a solution?
- Which solution does the user choose?
- How many solutions are there for this BC?
- Does the user successfully carry out the shot?
- The user first thought up a solution himself. Then he chooses one or more from the recommendations of the VBA. Is the VBA's solution he chose the same as his own?

The questionnaire (table 5.2) consists of 41 questions, of which 37 should be answered by a rating on the scale of 1..5, indicating:

1. User strongly disagrees
2. User disagrees
3. Neutral

4. User agrees
5. User strongly agrees

The user is allowed to supply arguments to his ratings.

The questionnaire is divided into 4 parts. The first two parts deal with the advice the VBA provides. First questions are asked about the advice the current version of the VBA gave. The second part questions about advice functionality that an ultimate VBA, were it available, should give. The third part deals with the representation method used. The fourth part questions about presentation methods that an ultimate VBA, were it available, should give.

5.3.3 Results

5 players took part in the user test, all members of BSV de Stoottroepen. The players were selected such that there experience level was spread along the spectrum of target users. 3 players are currently students at the University of Twente, the other two have been. This section first introduces the players. Next the data collected during the first sessions are shown. Then the results from the second session are shown, and finally the questionnaire and (aggregated) answers are shown.

Players

Player A

Age: 30
Experience level (billiards in general): Mostly libre, 6 years of competition play
Experience level (three cushion): 2 years of competition, has trained with qualified human trainer
Technical skills: Good
Moyenne: 0.3
Computer experience: Good (ICT-employee)

Player B

Age: 39
Experience level (billiards in general): Snooker, libre, three cushion.
Experience level (three cushion): 5 years of competition, has trained with qualified human trainer
Technical skills: Good
Moyenne: 0.35
Computer experience: Average

Player C

Age: 28
Experience level (billiards in general): Snooker, pool, libre, primarily three cushion. Participated in competitions for all 4 disciplines.
Experience level (three cushion): 5 years of competition, has trained with qualified human trainer
Technical skills: Excellent
Moyenne: 0.6
Computer experience: Good (math-student)

Player D

Age: 24
Experience level (billiards in general): Some libre, some pool, some three cushion
Experience level (three cushion): No competition, some recreative matches. Has trained with qualified human trainer.
Technical skills: Reasonable
Moyenne: Unkown (estimated <0.25)
Computer experience: Good (ICT-student)

Player E

Age: 25
Experience level (billiards in general): 5 years of pool experience, has trained with human pool trainer (so technical skills are present)
Experience level (three cushion): Estimated 20 hours in total in recreative play. Has not trained with human trainer.
Technical skills: Good
Moyenne: Unknown (estimated 0.25)
Computer experience: Good (study and work related)

Session 1

Instead of the tables with annotations for all 141 BC's collected, the relevant statistics are shown.

Player A

Number of BC's played: 26
Number of BC's attempted successfully: 3
Number of BC's attempted almost successfully ('just missed') : 13
Number of BC's attempted unsuccessfully: 10
Number of BC's for which the VBA's advice was used: 19
Number of BC's for which the VBA's advice was not asked: 0
Number of BC's for which no solution was found by the VBA: 6
Number of BC's for which the balls' positions could not be acquired: 1
Number of BC's for which solutions were browsed by the user: 10
Number of BC's for which the top solution was chosen by the user: 14
Number of BC's for which an animation was viewed: 14

Player B

Number of BC's played: 30
Number of BC's attempted successfully: 8
Number of BC's attempted almost successfully ('just missed') : 6
Number of BC's attempted unsuccessfully: 13
Number of BC's for which the VBA's advice was used: 9
Number of BC's for which the VBA's advice was not asked: 18
Number of BC's for which no solution was found by the VBA: 1
Number of BC's for which the balls' positions could not be acquired: 2
Number of BC's for which solutions were browsed by the user: 3

Number of BC's for which the top solution was chosen by the user: 6
Number of BC's for which an animation was viewed: 0

Player C

Number of BC's played: 21
Number of BC's attempted successfully: 5
Number of BC's attempted almost successfully ('just missed') : 9
Number of BC's attempted unsuccessfully: 5
Number of BC's for which the VBA's advice was used: 18
Number of BC's for which the VBA's advice was not asked: 1
Number of BC's for which no solution was found by the VBA: 0
Number of BC's for which the balls' positions could not be acquired: 2
Number of BC's for which solutions were browsed by the user: 11
Number of BC's for which the top solution was chosen by the user: 10
Number of BC's for which an animation was viewed: 1

Player D

Number of BC's played: 23
Number of BC's attempted successfully: 2
Number of BC's attempted almost successfully ('just missed') : 10
Number of BC's attempted unsuccessfully: 11
Number of BC's for which the VBA's advice was used: 22
Number of BC's for which the VBA's advice was not asked: 0
Number of BC's for which no solution was found by the VBA: 0
Number of BC's for which the balls' positions could not be acquired: 1
Number of BC's for which solutions were browsed by the user: 17
Number of BC's for which the top solution was chosen by the user: 17
Number of BC's for which an animation was viewed: 3

Player E

Number of BC's played: 43
Number of BC's attempted successfully: 5
Number of BC's attempted almost successfully ('just missed') : 7
Number of BC's attempted unsuccessfully: 31
Number of BC's for which the VBA's advice was used: 33
Number of BC's for which the VBA's advice was not asked: 6
Number of BC's for which no solution was found by the VBA: 2
Number of BC's for which the balls' positions could not be acquired: 2
Number of BC's for which solutions were browsed by the user: 4
Number of BC's for which the top solution was chosen by the user: 31
Number of BC's for which an animation was viewed: 3

Session 2

The idea to split up the user test in two sessions came after the first user test, in which player E was involved. Therefore this player only played freely (this also explains the big amount of BC's played in the first session), and is not included here.

Table 5.3.1 shows the results for the second session. Column A shows the BC numbering. Column B Shows whether an attempt was with (+) or without (-) consulting the VBA. Column C indicates which solution was chosen by the player. Column D shows the total number of solutions. Column E shows whether the attempt was successful (a), almost successful (close miss, b), or unsuccessful (c). Column F indicates whether the solution chosen from the set of solution is the same as the solution the user thought up himself at first.

Player A						Player B						Player C						Player D					
A. Ball configuration	B. VBA asked for advice	C. Chosen solution	D. Total number of solutions	E. Attempt successful	F. Solution same as user's	A. Ball configuration	B. VBA asked for advice	C. Chosen solution	D. Total number of solutions	E. Attempt successful	F. Solution same as user's	A. Ball configuration	B. VBA asked for advice	C. Chosen solution	D. Total number of solutions	E. Attempt successful	F. Solution same as user's	A. Ball configuration	B. VBA asked for advice	C. Chosen solution	D. Total number of solutions	E. Attempt successful	F. Solution same as user's
1	-			a		**1**	-			b		**1**	-			a		**1**	-			a	
	+	1	7	a	+		-			a			+	2	3	a	+		-			c	
2	-			c			+	1	10	a	+	**2**	-			c			+	2	3	b	
	-			a		**2**	-			c			-			c		**2**	-			c	
	+	1	6	c			+	3	6	a	+		+	2	3	c			-			c	
	+	1	6	c			+	2	6	b		**3**	-			b			+	2	6	c	
	+	1	6	c			+	1	6	a			+	1	4	b	+		+	1	6	c	+
	+	1	6	c		**3**	-			a		**4**	-			a			+	1	6	c	
	+	2	6	c	+		+	2	5	c	+		+	2	4	a			+	1	6	c	
3	-			c			+	4	5	c		**5**	-			a			+	1	6	c	
	-			c			+	4	5	c			+	3	11	a	+		-			c	
	-			c			+	4	5	c		**6**	-			a			-			c	
	+	2	2	b			+	1	5	b			+	8	12	a	+		-			a	
	+	2	2	b			+	1	5	b		**7**	-			c		**3**	-			a	
	+	2	2	a			+	1	5	b			+	2	3	c			+	3	5	b	
	+	2	2	b			+	1	5	b		**8**	-			c			+	5	5	b	+
	+	1	2	c			+	1	5	b			+	2	4	a	+	**4**	-			b	
	+	1	2	c								**9**	-			a			+	1	3	b	+
	+	1	2	c									+	2	2	a			+	1	3	b	
4	-			b															+	1	3	b	
	-			a															+	1	3	b	
	+	1	1	c	+													**5**	-			a	
	+	1	1	c															+	6	7	b	
	+	1	1	c															+	6	7	a	
																		6	-			c	
																			+	1	4	b	
																			+	2	4	b	+

Table 5.3.1: Second session results for players A,B,C and D

Questionnaires

After the two sessions all users filled in a questionnaire. The results are shown below. No averages are shown, only the individual ratings. The additional remarks and explanations by the test users are mentioned in the discussion section, when relevant.

User	A	B	C	D	E
1.Advice (current version)					
1.1. I trusted the VBA's advice	5	4	4	4	4
1.2. The top solution was always the best	4	4	3	4	4
1.3. Offering multiple solutions was useful	5	5	5	5	5
1.4. Sometimes solutions were missing	4	1	4	4	3
1.5. There were too many solutions that did not make sense	2	2	2	2	2
1.6. The VBA's advice was useful for BC's I *could not* solve myself	5	5	4	5	5
1.7. The VBA's advice was useful for BC's I *could* solve myself	5	4	2	2	4
1.8. Using the VBA's sevices provided more insight in the game	5	4	4	2	4
1.9. The VBA's advice made me feel more secure	4	3	4	4	1
1.10. When I failed a shot, it mostly was due to myself	4	5	5	5	4
1.11. When I failed a shot, it mostly was due to the VBA's advice	1	1	2	2	1
1.12. The VBA in combination with a training scheme would be more useful than one of both alone	4	4	4	4	5
1.13. I'd trust the VBA's advice more than my own insight	3	2	3	4	3
1.14. I'd trust the VBA's advice more than from a qualified human trainer	2	1	2	1	1
1.15. I think my skills would improve more using a VBA's aide	5	5	4	4	4
1.16. I'd rather train with a VBA than with a player as good as me	4	4	3	4	3
1.17. I'd rather train with a VBA than with a player better than me	3	2	2	1	1
1.18. I'd rather train with a VBA than with a player less good as me	5	4	4	3	5
Advice (ultimate version)					
2.1. I believe feedback on my errors would be useful	5	5	5	5	5
2.2. I believe it would be useful if the VBA would prescripe BC's to play	5	5	5	4	5
2.3. I believe it would be useful to have a players profile, keeping track on my performance on different types of shots	5	5	5	4	2
2.4. I believe it would be useful that the VBA prescripes BC's based on this profile	5	5	5	4	4
2.5. I believe it would be useful to receive advice on technical issues	5	5	5	5	1
2.6. I believe it would be useful to receive feedback on technical issues	5	5	5	5	5
Presentation (current version)					
3.1. The current presentation of a solution is clear	4	5	4	5	4
3.2. The aiming direction is clear	4	4	4	4	5
3.3. The thickness with which ob1 should be hit is clear	3	4	4	2	4
3.4. The amount of velocity that should be imparted on the ball is clear	4	5	1	3	2
3.5. The amount and type of spin that should be inflicted is clear	2	5	3	5	4
3.6. The current method of presentation is too simplistic	2	1	2	1	4
3.7. The possibility to watch the shot animated is useful	5	4	5	3	4
3.8. The possibility to navigate freely through the 3d representation is useful	3	3	3	4	3
3.9. The time the VBA needs to solve a shot is too long	2	2	2	2	2
Presentation (ultimate version)					
4.1. The use of augmented reality by projecting lines on the table	4	4	5	5	5

using beamers would be useful					
4.2. The use of augmented reality by projecting lines on the table using a wearable head display would be useful	3	3	3	1	4
4.3. An ECA would be a welcome addition to interaction with the VBA	3	4	4	2	1
4.4. The lack of interaction with a real human would prevent me to replace a human trainer with a VBA	4	4	4	4	5

Table 5.2: Questionnaire. Answers are on a 5 point likert scale, ranging from 1 (user completely disagrees) to 5 (user completely agrees).

5.3.4 Discussion

The questions posed in 5.3:

1. **Does the user consider a VBA useful?**
2. **How does the user use the VBA?**
3. **Does the user trust the VBA's advice?**
4. **Is the current, basic presentation method sufficient to communicate solutions?**

As the results for the individual users were quite different, an attempt to answer these questions will be done for each player individually, based on the session and questionnaire data. After that an analysis will be performed on the results of the questionnaire. Finally a conclusion is built upon the available results.

Player A

1. Player A asked the VBA for advice for all BC's he encountered during the first session. Judging from his answers to the questionnaire, he does find a VBA useful, but would still prefer a human trainer.
2. Player A used the VBA to solve every ball. This can partially be attributed to his enthusiasm in testing the system. But it was observed that he did need it sometimes for finding solutions, or alternatives to his own solutions. In contrast with other testers, he frequently (14/19) viewed animations of the shot. He did this mostly to get a feel of the shot and observe the danger of a kiss.
3. Because player A asked the VBA for advice in all cases, and chose the top solution (14/19), It can be stated that he trusted the VBA. His answers in the questionnaire do not contradict this.
4. During the sessions, no problems were observed concerning misinterpretation of the solution. Player A did indicate in the questionnaire, that a more detailed indication of the amount of spin, as well as an explicit indication of the thickness at which OB1 should be hit, would be preferred.

Player B

1. Player B used the VBA much less than the other players, however the questionnaire answers point out that he would find the VBA's assistance useful. In one instance he said "thank you" out loud.
2. Player B, as opposed to the other players, almost only used the VBA during the first session when he ran into trouble, and needed a solution. During the second session, when he was forced to use the VBA, he indicated that it was useful to see alternatives for BC's he could solve himself. Sometimes seeing the alternatives made him reconsider his own solution.

3. The player indicated in the questionnaire, that he did not always trust the top solution the VBA advised to be the best. In these cases he often assessed the second or third as the best solution. This is reflected in that he chose the top solution 6/9 of the time during the first session, and 7/13 during the second. That this player seem to trust (and use) the VBA less than the others, might be attributed to that he was, especially in the beginning, running very well on his own.
4. The player had no problem at all at interpreting the shown solution. He did not use animation, nor did he use free navigation.

Player C

1. Player C is the most experienced user, with an average placing him above the target user group. Surprisingly, he did find using the VBA's advice quite useful. Only once he chose to not follow the VBA's advice. Player C did complain about the absence of solutions involving contra spin on the first cushion. As these are considered advanced, these are the ones he would like to practice more often. He used the second session more to explore the VBA, and found that the solutions the VBA came up with were correct.
2. Player C was very interested and used the VBA's service extensively. He browsed the different solutions most of the time, to find one he was most comfortable with. Because of the player's experience, he recognized solutions immediately. Therefore he did not need the information about which spin had to be imparted.
3. Player C did trust the VBA more than expected, as he is the player with the most expertise and experience. Due to player C's expertise, one would expect him to be cautious about the advice. However his answers to the questionnaire make clear that he attributed misses to himself, not to the advice of the VBA, i.e. he thought the VBA was right in these cases. Although he did add a remark that sometimes the VBA was clearly wrong. Player C did browse a lot through the solutions. Often he did not consider the top solution most suitable: 9/19 in the first session, 2/10 in the second session, but it should be noticed that the second session was also used to find alternative solutions to his own.
4. Player C was not interested in freely navigating through the solutions. He used animation only twice, to see how big the risk for a kiss was. He added the remark that he would like to see the path of object ball 1 indicated also.

Player D

1. Player D was the most inexperienced test user, and had a hard time reproducing the shot parameters as displayed. However, he did find the VBA's advice useful, as in many instances he had no idea how to solve a particular BC.
2. Where the more experienced three cushion players could immediately transfer the solution displayed on the screen to the BC on the real table mentally, player D found this somewhat more difficult, and therefore used the free navigation a couple of times. He browsed the solution set almost every time, to find a shot that he liked. While watching him, I did notice that he often chose the wrong solution, i.e. not the easiest. This indicates that for a less experienced player, the solution sorting should be improved so that only one solution will be recommended.
3. Player D seemed to trust the VBA, and chose the top solution 16/22 times during the first session, 9/16 during the second.

4. Player D would like to have an exact visualization indicating the thickness with which the cue ball should hit the first object ball. He also indicated that it was difficult to estimate the velocity at which the cue ball should be struck.

Player E

1. Player E noticed an interesting phenomenon. He claimed that being advised by the VBA made him lazy, and less careful than when he would think up a solution on his own. This effect would totally undermine the goal of the VBA to improve a user's skill. A more guided training methodology would improve this. He indicated that he would like a version where users could train, get better, and interactively compare their statistics with others, introducing a game element. Furthermore the player did not find the VBA very useful in his case, as he is a pool player and not so much interested in three cushion play.
2. Player E had not been playing frequently for a while. Therefore he had great difficulties with aiming and playing fluently. This, together with the fact that he only did session one, seemed to result in him getting bored and less interested in carefully reproducing the VBA's advice.
3. Judging from the amount of time he used the VBA's advice (33/39), as well as the amount of times he used the top solution (31!), he did trust the VBA. However, as he had very little experience in three cushion billiards, he was more dependent on the VBA's advice than the players A,B and C.
4. Player E did find the way of communication intuitive enough. The only problem he had was figuring out at what velocity to strike the ball.

Questionnaire

The answers given by the users on the questionnaire (table 5.2) speak for themselves. We will discuss the questions where the answers differed.

1.4. Sometimes solutions were missing	**4**	**1**	**4**	**4**	**3**
Player B only used the VBA in case he was not able to figure out his own solution. Therefore there were no solutions that he missed from the set of solutions.					
1.7. The VBA's advice was useful for BC's I *could* solve myself	**5**	**4**	**2**	**2**	**4**
Both player C and D did not need the VBA as a reference for their own solutions, only for the BC's they could not solve themselves.					
1.8. Using the VBA's services provided more insight in the game	**5**	**4**	**4**	**2**	**4**
Player D felt he did not gain more insight in the game, because he felt he was more following up orders, without really understanding how and why a solution was recommended.					
1.18. I'd rather train with a VBA than with a player less good as me	**5**	**4**	**4**	**3**	**5**
Player D argued that when he would train with a player less good, roles would be reversed and he would be the teacher, which could help him gaining insight.					
2.3. I believe it would be useful to have a players profile, keeping track on my performance on different types of shots	**5**	**5**	**5**	**4**	**2**
Player E argued that he knows his weaknesses himself, and does not need the VBA for that.					
2.5. I believe it would be useful to receive advice on technical issues	**5**	**5**	**5**	**5**	**1**

Player E argued that, as a pool player, he was not interested in learning technical skills in three cushion billiards, as it might harm his pool game.

3.3. The thickness with which ob1 should be hit is clear	**3**	**4**	**4**	**2**	**4**
Both A and D could not easily deduce the thickness from the line indicating the path of the white.					
3.4. The amount of velocity that should be imparted on the ball is clear	**4**	**5**	**1**	**3**	**2**
B,C and D could not easily deduce the required impact velocity from the line indicating the path of the white. C suggested a bar on the screen as an indicator.					
3.5. The amount and type of spin that should be inflicted is clear	**2**	**5**	**3**	**5**	**4**
A and C would like to see more detail in the amount of spin required.					
3.7. The possibility to watch the shot animated is useful	**5**	**4**	**5**	**3**	**4**
It was expected at first that the player with the least experience would benefit most from animation. However, player D barely cares about it, whereas the more experienced players found it useful to estimate the risk for a kiss.					
4.2. The use of augmented reality by projecting lines on the table using a wearable head display would be useful	**3**	**3**	**3**	**1**	**4**
Player D argued this would be impractical.					
4.3. An ECA would be a welcome addition to interaction with the VBA	**3**	**4**	**4**	**2**	**1**
D and E argued that it would be distracting.					

Conclusions

1.Does the user consider a VBA useful?

Based on the sessions as well as the questionnaire, it can be concluded that the players did show interest in the possibility of a Virtual Billiards Assistant, and would find it a useful addition.

2.How does the user use the VBA?

During the sessions it became apparent that there are some differences in the way users might use the VBA's services. One was only interested when he could not solve a BC himself. The more experienced player liked to use the animation as a reference for assessing the danger of a kiss. Some looked at the screen just briefly, to have a coarse idea on how to play, others looked more detailed to the path of the white, and where it should hit the cushions. Some seem to trust the advice for the best solution, and base their attempt on it, others extensively use the solution browsing option, looking for attractive alternatives.

Obviously, the tests were very short and some users were naturally more interested in the VBA itself, testing what it could do, so it probably is not a real good reflection of how a billiards player would use the VBA's services when he grows accustomed to it. It does however show that different users find different uses, even in this basic version. The improvement of a players' skill would probably benefit from a more complete training methodology, instead of just using the VBA as a shot solving tool.

3.Does the user trust the VBA's advice?

The users trusted the VBA more than expected beforehand. Although they would still trust a real human trainer more at the moment. All players were able to notice flaws in the solution sets, i.e. shots that were (near) impossible to reproduce. However, they did not question the accuracy of the shown paths when a solution was found that they found suitable. The imperfection in the model thus did not seem to be noticed, or something the players were bothered with. This implies that requirement 1.5 is not violated.

4.Is the current, basic presentation method sufficient to communicate solutions?

The current presentation method was found sufficient, although a more precise indication of thickness and initial velocity (which are at the moment implicitly given, by the path of the white) were requested by some. More sophisticated methods of presentation by projecting the path on the table itself using beamers was found desirable by some, though one player made the remark that it might make the game too easy for training purposes. Opinions on whether an ECA would be useful varied too much to conclude on.

The user test shows that the requirement that *the shot parameters should be presented in such a way that the user is able to reproduce a shot accurately* is met, the only issues encountered were that some users would like an explicit indicator for the velocity at which a ball should be struck, and for the thickness at which OB1 should be hit. The requirement that *the user should be able to interact freely and intuitively with the VBA, he should not feel restricted by the interface*, also can be considered to be met, indicated by the fact that different users found different ways to use the VBA's service. This method of presenting a solution thus can be seen as an answer to the research question: *How to present a solution to a user*?

6. Conclusions and Future Work

Although the virtual billiards assistant developed during this project is not an ultimate one, in the sense that it could render human expert trainers obsolete, significant steps were made in exploring the possibilities of aiding a billiards player using technology. The current version of the VBA is the first known digital tool that is able to assist a three cushion player by solving the tactical part of the game.

The user tests in chapter 5.3 point out that the current version of the VBA is found useful by players of different experience levels, and would be considered a welcome addition to their training repertoire. It cannot be concluded however whether using the VBA would result in greater improvement of skill when compared to traditional training methods. Some of the users argued that the availability of a VBA during practice might be 'misused', by simply relying blindly on the VBA, making the player lazy. To solve this more guided methods could be introduced, demanding a more pro-active attitude of the user. However by using the VBA's services, a player will not find himself in a situation anymore where he has no clue how to solve a ball configuration, or comes up with a solution that is considered unfeasible.

The main research questions *How can the VBA calculate a solution to recommend to a player for any given ball configuration?* and *How to communicate a solution with the user? w*ere answered in this thesis by choosing methods that were implemented, tested and analyzed.

The general requirements that were posed are fulfilled. The VBA is reasonably fast. Furthermore the VBA is little obtrusive and allows the player to move freely. The VBA is able to provide an advice for almost all ball configurations, although it might not always be the solution considered most feasible by a human expert. The VBA runs robustly in the testing environment, during the user tests and demonstrations no failures occurred. The user tests indicated that users trust, and are willing to trust the VBA to a high degree.

In the remainder of this chapter the different parts of the VBA will be treated separately, discussing the results and what should be done to improve the methods.

6.1 The Physical Model

The physics involved in billiards were modeled accurately, and suitable for implementation, drawing heavily upon the models in [4] and [7]. The model and its implementation were briefly analyzed, by comparing simulated shots to shots carried out on a real table. This comparative analysis was not very in-depth, and should be carried out more thoroughly, because the usefulness of the VBA depends heavily on how accurate it can simulate shots. A user should trust the VBA, and the VBA should not violate this trust.

That said, the model and its implementation are observed to be accurate enough when looking at the final result. The user test, as well as the evaluations in chapter 4 and own observations all indicate that the solutions found are reproducible on a real table, the margin of error introduced by the difference between the real table and the simulated one was subjectively found acceptable.

Improvements can be made however. First of all, the issue of contra-spin in resolving a ball-cushion impact should be analyzed. This problem was simply ignored during the project by excluding solutions involving contra-spin on the first cushion. Although this did not result in a great loss of viable solutions, in some cases it would have been a better solution. Other recommended improvements are introducing friction in ball-ball impact and a more realistic model of cue-ball impact.

Another important issue of future implementation involves the fact that no two tables in real life are exactly the same. The more accurate the model, the bigger the importance of calibrating this model to the specific characteristics of an individual table. To make matters even more complicated, a table itself, especially older tables, can have great local differences. An example of this is caused by the fact that a billiard match always starts with the same ball configuration, which almost every billiards player plays using the same solution. The point of impact on the first cushion thus is hit more frequent, relative to other parts of the cushion. After a considerable amount of time this causes the rubber at that point to wear out more than other parts of the cushion, which can greatly influence the coefficient of restitution.

6.2 Finding Solutions for Ball Configurations

The possibility to simulate many shots in a short time period enables the VBA to find solutions using a more or less brute-force method. It was decided this was preferred over the use of existing methods like the systems in [VER87]. The main reason for this is that using the brute force method almost all ball configurations can be solved. Furthermore solutions are accurate, as the method makes use of a detailed physical model.

The requirements associated with generating possible solutions state that the set of solution for a ball configuration should be complete (i.e. all reasonable solutions should be present), as well as that the resulting solutions should be clustered in a natural way, resulting in a set of one or more solutions that are perceived as of different type.

The expert evaluation showed that not all reasonable solutions were found for every ball configuration. This could be improved by simulating more shots, which requires a greater range of the input shot parameters. Even solutions involving $\theta > 0$ could be considered in some cases. During the user tests, it was observed the cases where the users indicated that the set of solutions contained zero solutions they considered feasible were rare.

The clustering of shots, based on the first 4 cushions, was found adequate enough by the expert, as well as own observations.

The next step in finding solutions to recommend to the user is sorting the shots, in such a way that the top solution is also the solution a human expert would recommend. A heuristic based sorting method was explored. During a first evaluation, although most of the times the solution surfacing as best was an acceptable solution, it often was not the one the expert would recommend. It is unclear whether the heuristics based approach is suitable at all, although it seems to use the same intuitive approach a human expert trainer uses. During a second evaluation, after the shot generation method was extended by adding more possible values for V, the sorting of shots scored better. 17 out of 44 sets had the solution that was recommended by the expert ranked as best.

To improve the sorting algorithm an attempt should be made to use more heuristics. For example, a possible heuristic might be whether the player can view both the aiming point on the first cushion as well as the aiming point on the second cushion while lining up for a shot, without having to turn his head. Also, some balls are positioned in a way that it is difficult for the player to reach, depending on whether he is right- or left-handed.

As stated before, the described method to solve ball configurations is brute force. An advantage of this method, which was not stated as a requirement, is that it is built as a layer on the implementation of the physical model. If a solution found with this method is incorrect, the error is caused by inaccuracies in the physical model, not in the generation method.

6.4 Interaction

One of the requirements for the VBA was that it should be unobtrusive. In order to achieve this, the interaction with the user is kept simple. Only one input action is required by the user to request solutions for an encountered ball configuration: pressing the space bar on a laptop. The only constraint is that the user must not stand in front of the camera. Upon request the user has to wait a few seconds, before a sound indicates that a solution is found (or not).

The user test indicated that this method of interaction was found pleasant and preferred by the users.

The acquisition of balls using a webcam is sufficiently precise, no cases were encountered where the error was noticeable. Only in very few cases the VBA was unable to resolve the balls' positions.

The interaction layer can be improved in numerous ways. It could serve as a platform for more research in the domain of human media interaction, especially on the topic of virtual trainers. Three subjects that should be very interesting to look into are:

- Tracking the movement of balls during a shot, and afterwards deducing what the differences are between the shot as recommended by the VBA and the shot carried out by the player. These differences should yield information about what a player did wrong if he misses a shot, typically resulting in the VBA giving tips much like a real trainer would.
- Trying out more complicated presentation methods, for example using augmented reality like in [2] or [3]. Also the use of avatars might be a welcome addition. Another technology that would be very interesting to look at in the future is electronic ink. This could make it possible to turn the table's cloth into a display itself, so that the paths could be drawn directly on it.
- Helping players improve technical skills by providing tips and feedback. This could be achieved for example by using computer vision to analyze a players stance, or using cue's with built in gyroscopes to analyze in great detail the striking movement of the cue before and upon cue-ball impact. This would be helpful for players of every experience level, even the highest.

Bibliography

1 de Beer, R. Mozartjaar 2006: De intrede van de kitsch. *de Volkskrant* (januari 2006).

2 Larsen, Lars Bo. The Automatic Pool Trainer - a Platform for Experiments with Multi Modal User Interaction. *Proceedings of the Fourth Danish HCI Research Symposium, November* (November 2004).

3 Jebara, Tony, Eyster, Cyrus, Weaver, Josh, Starner, Thad, and Pentland, Alex. *Stochasticks: Augmenting the Billiards Experience with Probabilistic Vision and Wearable Computers*. MIT Media Lab, October 1997.

4 Leckie, Will and Greenspan, Michael. An event-based pool physics simulator. *Lecture Notes on Computer Science: 11th Advances in Computer Games*, 2006, 4250 (December 2006), 247-262.

5 Smith, Michael. PickPocket: A Computer Billiards Shark. *Artificial Intelligence*, 2007, Volume 171, Issues 16-17 (November 2007), 1069-1091.

6 Verworst, Jean. *Berekend biljarten*. Jean Verworst, Lier, Belgium, 1987.

7 Han, Inhwan. Dynamics in carom and three cushion billiards. *Journal of Mechanical Science and Technology*, 2005, Volume 19, Number 4 (April 2005), 976-984.

8 Marlow, Wayland C. *The Physics of Pocket Billiards*. American Institute of Physics, 1996.

9 Alciatore, Dave. Throw - Part I: introduction. *Billiards Digest* (August 2006). http://billiards.colostate.edu/bd_articles/2006/aug06.pdf.

Appendices

Appendix A: Three Cushion Billiard Game Rules

Introduction

Three Cushion billiard is a form of billiards, with simple rules but conceived as quite difficult to master. In this appendix the rules and terminology will be explained.

Material

Billiards is played on a billiards table (Figure 1). The table surface is covered with a (usually green or blue) cloth, surrounded by four rubber cushions, and a wooden frame. The dimensions of the cloth within the cushions are typically 230x115cm, or 285x142,5cm. The latter is used for professional 3 cushion billiards, most billiard locations will have the smaller sized table. Because the smaller sized table is far more suitable for beginners and slightly advanced players, this size is used throughout the project.

On the wooden frame surrounding the cushions are white dots called diamonds. These diamonds can be used as visual aids, for determining ball positions or aiming points. The diamonds are key in three cushion solution systems like Verworsts' [6].

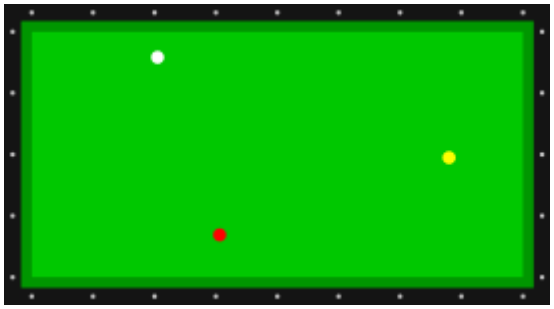


Figure 1: Billiard table

On the table are three balls, a white, a yellow and a red one. Balls have a standardized radius of 31mm. The cushions usually have a height of 40mm. A billiard match is always between two opponents. Each one of the players is assigned a different ball as theirs, either the white or the yellow. That ball is called the players' cue ball.

The cue is a wooden stick, with a piece of leather on top called pomerans. The pomerans is covered with blue chalk, to enhance the application of spin on the cue ball.

Throughout the project, the positions of the three balls within the playing field are called a ball configuration (BC).

The Rules

The objective of a three cushion game is to score as much points as needed to complete a match. In order to win the match, a player must score the amount of points needed before his opponent does. The number of points needed often differs due to the fact that most billiard competitions have introduced a handicap system. As an example we introduce player A, needing 25 points to complete a match, and player B needing 30 points to complete.

At the start of a match it is decided which player will start. This player gets assigned the white ball as cue ball. The match starts with a standard ball configuration called Acquit (Figure 2).

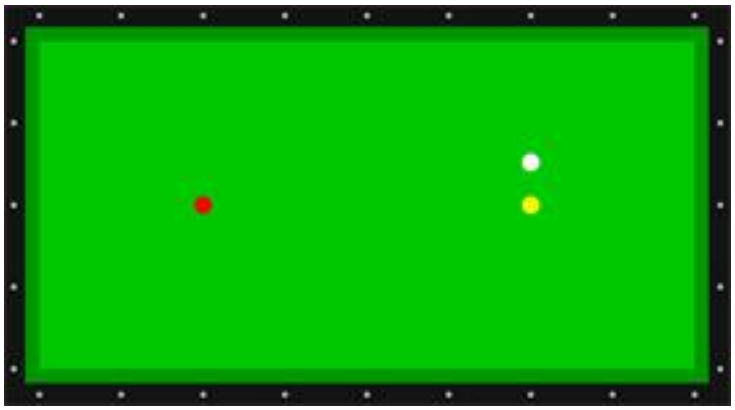


Figure 2: Acquit

To score a point, the player has to strike his cue ball in such a way that:

1. The cue ball collides once or more with the other two balls
2. Before the first collision with the second ball occurs, the cue ball has collided three or more times with a cushion

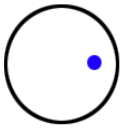
In this case the player is said to have produced a carombole, rewarded with a point. The player now faces a new ball configuration and can have another attempt to gain a point. The player can play on until he fails to produce a carombole, after which the turn is given to the other player to attempt to gain points.

Throughout the project the ball with which the cue ball (or CB) collides first is called object ball 1 (or OB1), the other object ball 2 (or OB2).

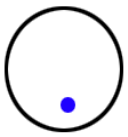
A variety of possible three cushion shots can be watched for example at:
http://nl.youtube.com/watch?v=BF81cWqklS4

Techniques

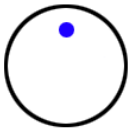
Different techniques can be used to force a cue ball to travel along a path, depending on where the ball is struck with a cue. In figure 3a the application of side spin –also called English- is illustrated. Side spin is applied to influence the outgoing angle after colliding with a cushion. This opens up a range of possibilities when defining a path the cue ball should travel. Figure 3b and 3c illustrate bottom and top spin. This type of spin is used to influence the path of the cue ball after it collided with the second ball.



Figuur 3a: English



Figuur 3b: Bottom Spin



Figuur 3c: Top Spin

Appendix B: Terminology

In this appendix several billiard-related terms are explained, to enhance the readability of this document.

Cue: The cue is a wooden stick, used by the player to strike the cue ball in order to set it in motion.

Cue Ball (CB): The ball that is struck with the cue.

Object ball 1 (OB1): The ball first to collide with the cue ball.

Object ball 2 (OB2): The ball second to collide with the cue ball.

Ball Configuration (BC): The CB, OB1 and OB2 when at rest, take up different positions in the continuous 2d space within the table frame. The coordinates of these make up the ball configuration. Each configuration requires its own unique solution(s).

Billiards: Billiards is a general term for cue sports like pool, snooker, libre, three cushion and others. In this document billiards is often used, unless explicitly stated, as short for three cushion billiards.

Carombole: When a shot is carried out successfully, the term carombole is used, and a point is awarded. Carombole is also used as a general term for billiard games without pockets (holes in the table).

Diamonds: Markers on the wooden frame around the table, 7 on each long rail, 3 on each short rail, dividing a table into 8x4 grid.

Kiss: A kiss is an unintended (early) collision between two balls, typically preventing the shot from resulting in a carombole.

Libre: Libre is played with exactly the same materials as three cushion billiards. The difference is that the cue ball only has to impact with the two object balls, without the obligation of hitting cushions.

Moyenne/Average: At the end of, or during a match, a player's moyenne can be calculated by dividing his amount of points by the amount of turns he has taken. When accumulated over a longer period (typically a season), this is an indication of a player's level. This average is used in a handicapping system, to determine the amount of points a player has to make to finish his match.

Appendix C: Brief Evaluation of Physical Model

In order to tune the parameters of the physical model, and evaluate it, some shots were carried out using only the cue ball, playing it to hit 4 or 5 cushions. The shots were conducted on the real table, from fixed starting positions aiming at a particular diamond, at such a pace that the cue ball would end its path shortly after impact with the last cushion. Impact points of the cue ball with the cushions were estimated. The figures below show the path of the white as estimated on the real table as a straight line (from impact point to impact point). The simulated shot is depicted with a dotted line. The pictures clearly indicate that the behavior of the cue ball is not predicted well if the first impact point is near the corner. It was not determined nor researched why this is the case.

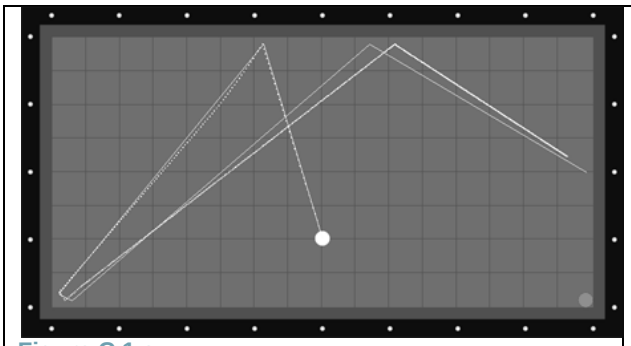


Figure C.1.a

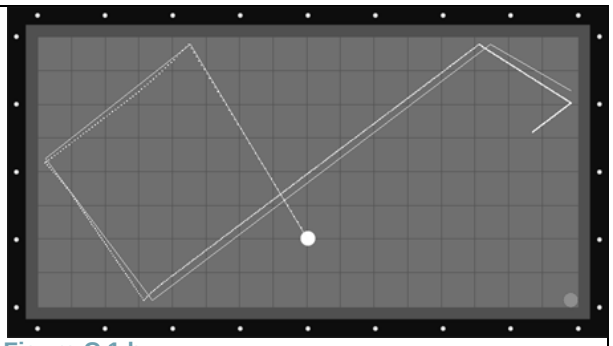


Figure C.1.b

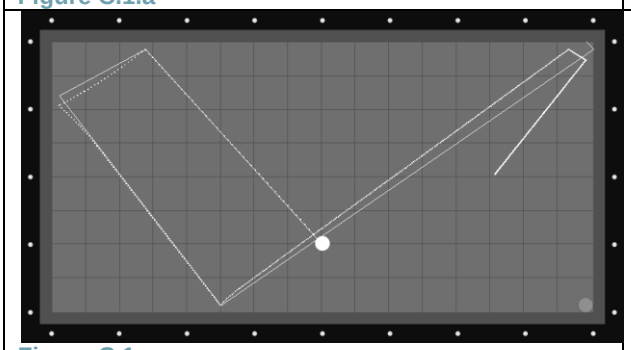


Figure C.1.c

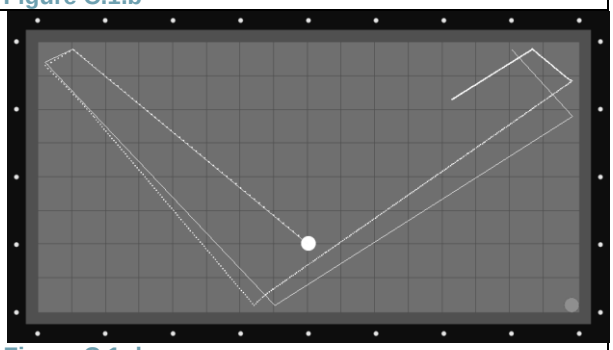


Figure C.1.d

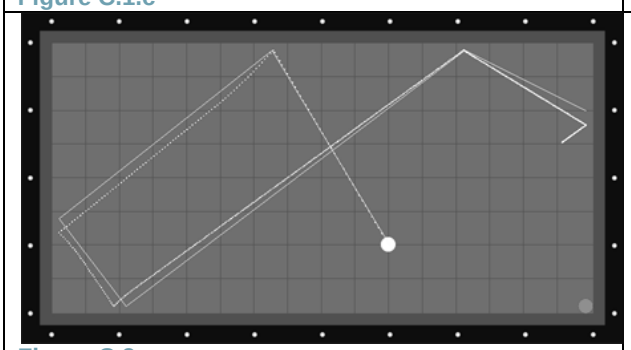


Figure C.2.a

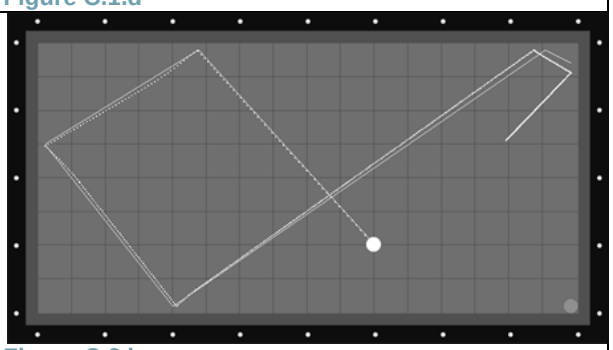


Figure C.2.b

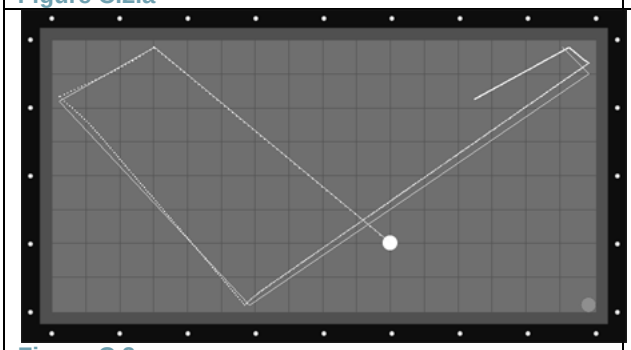


Figure C.2.c

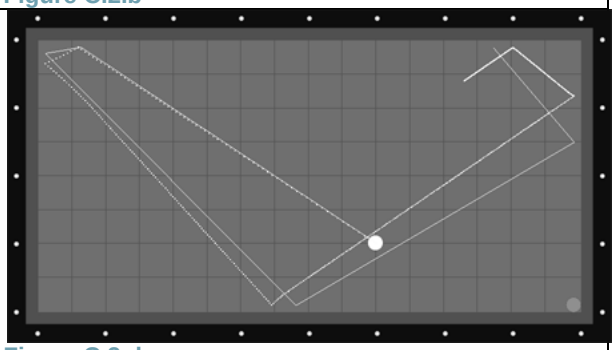


Figure C.2.d

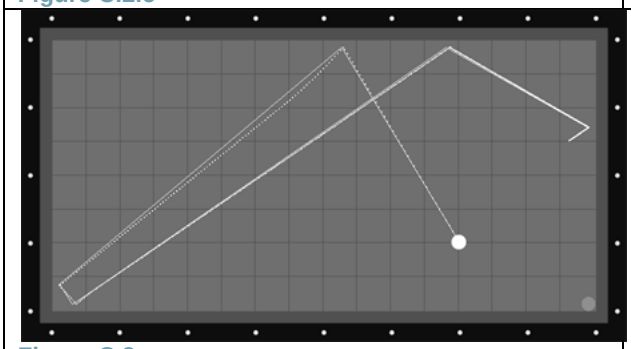


Figure C.3.a

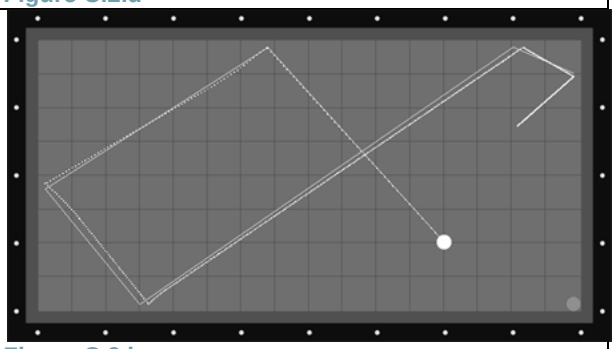


Figure C.3.b

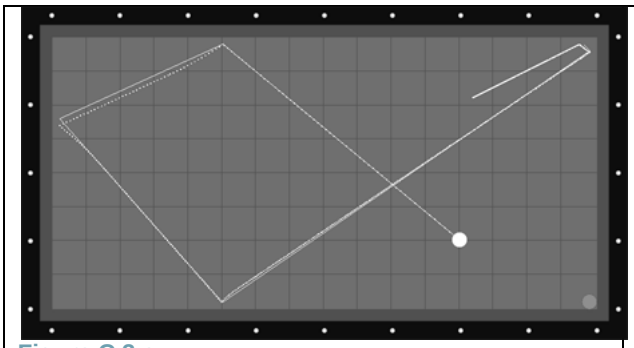


Figure C.3.c

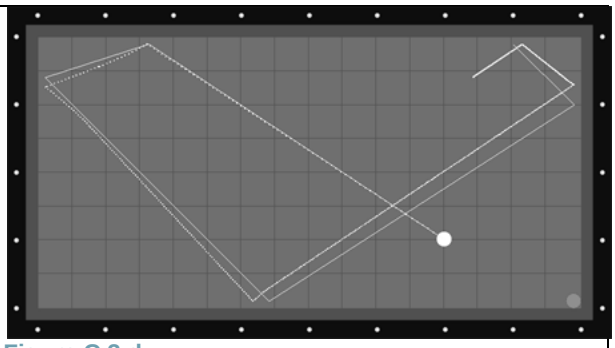


Figure C.3.d

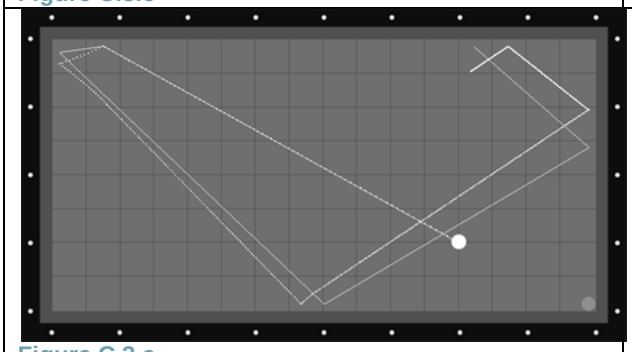


Figure C.3.e

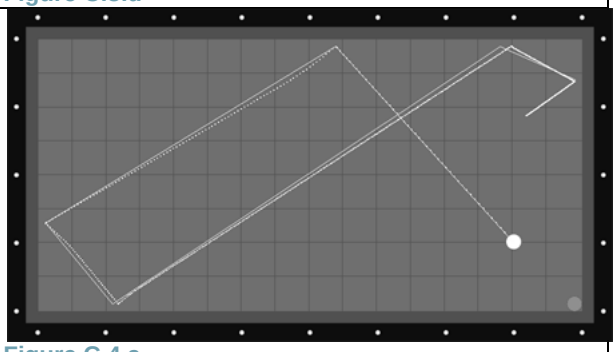


Figure C.4.a

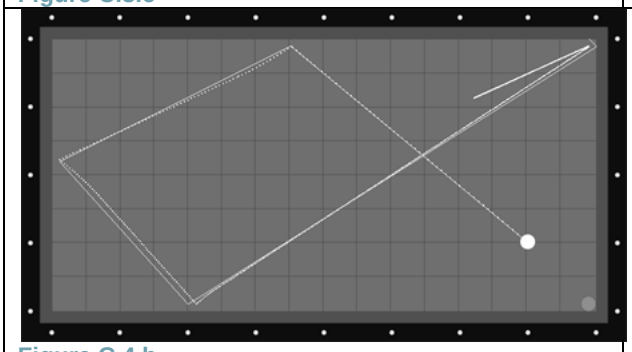


Figure C.4.b

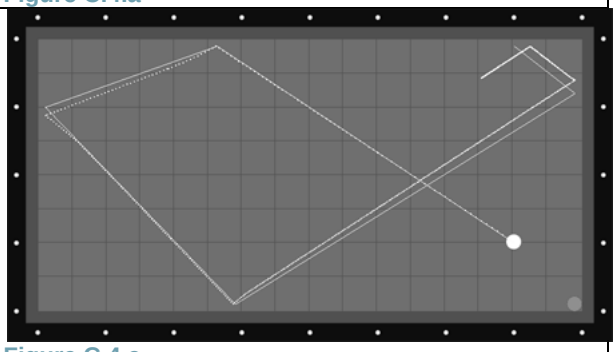


Figure C.4.c

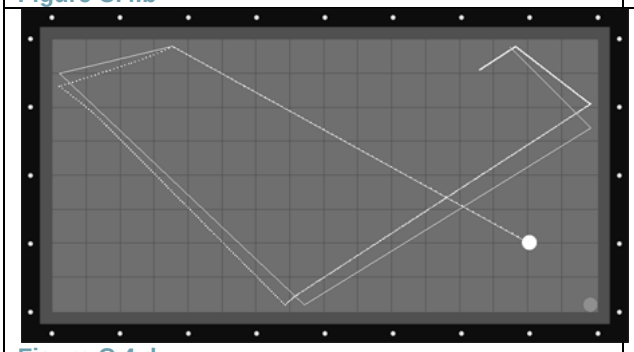


Figure C.4.d

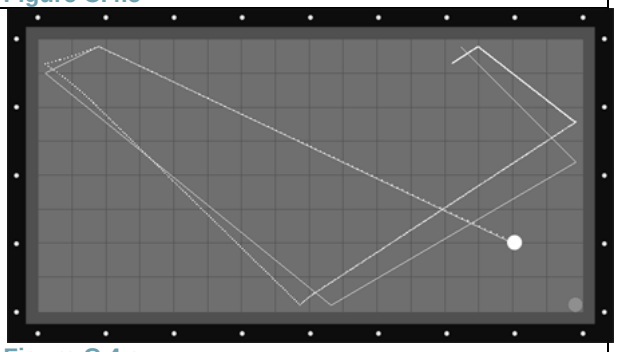
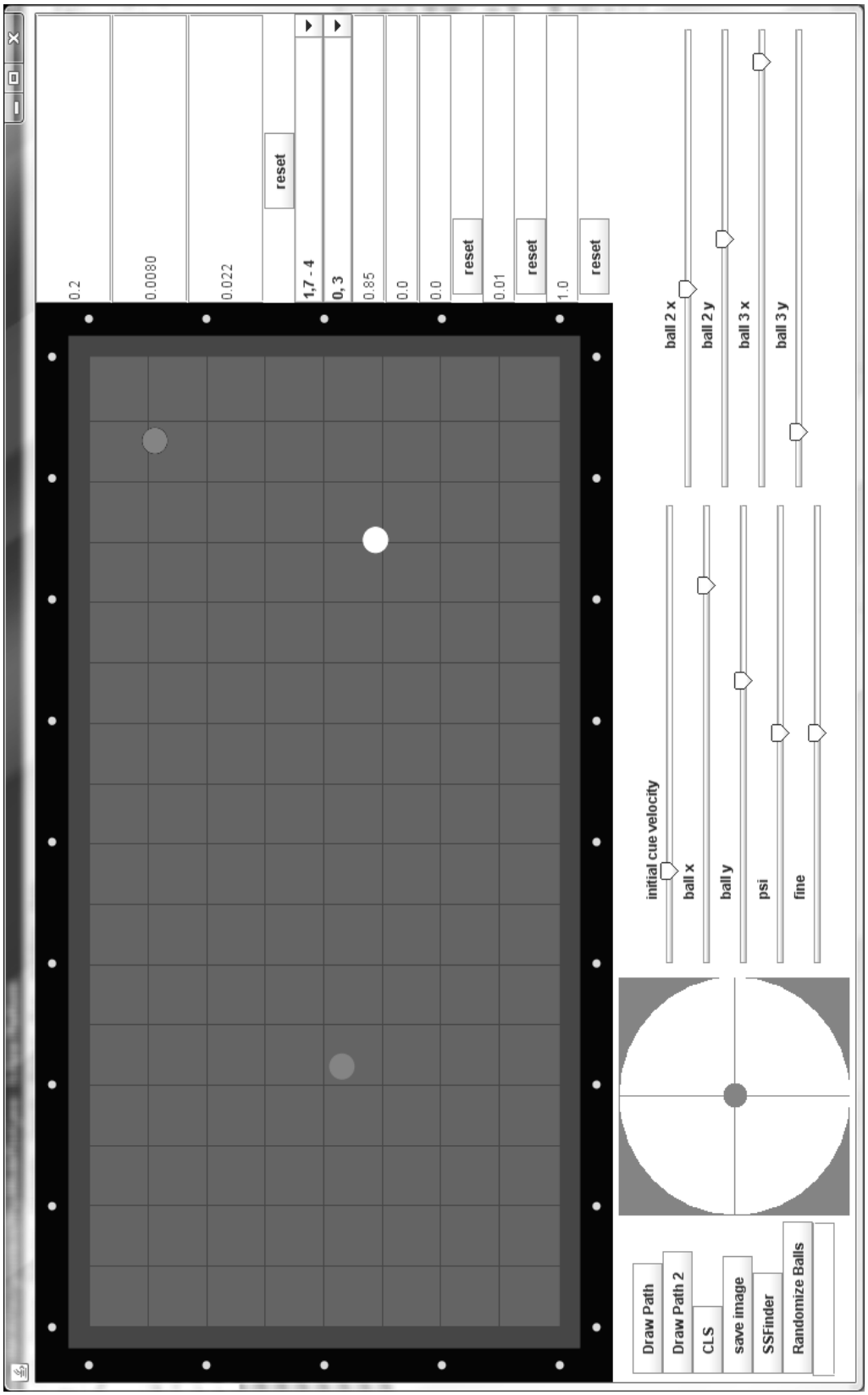


Figure C.4.e

Appendix D: Test Application



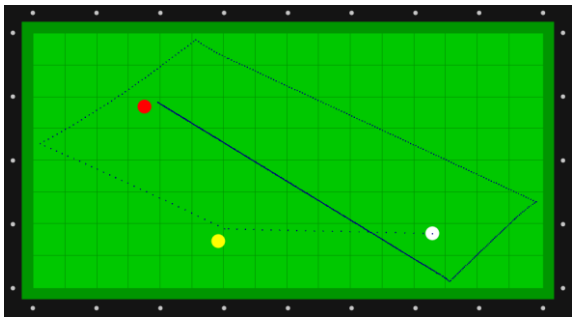
Test application, used throughout different stages of the project, mainly development and evaluation of the physical model and shot recommendation method

Appendix E: Reproducibility Results

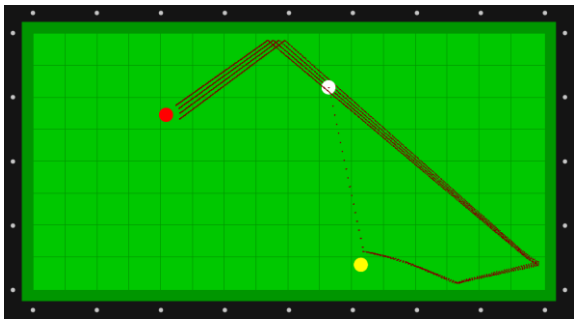
Table D.1 shows the results of the evaluation carried out in 4.1.4.1. The first column shows the number of the solution cluster. For example the first cluster of the second ball configuration is numbered 2.1. The second column shows the amount of solutions in the cluster, the third shows whether the shot was reproducible (1), approximately reproducible(2) or not reproducible(3). The bold faced solutions are shown on the next page.

Solution	SIC	Repr.	Solution	SIC	Repr.
1.1	5	1	6.1	9	1
1.2	4	1	**6.2**	**3**	**1**
1.3	3	1	6.3	2	3
1.4	**1**	**1**	6.4	1	1
1.5	1	2			
			7.1	1	3
2.1	**4**	**1**	**7.2**	**1**	**3**
2.2	4	3	7.3	1	1
2.3	3	1	7.4	1	3
2.4	3	1	7.5	1	3
2.5	2	3	7.6	1	3
2.6	1	3			
2.7	1	1	8.1	3	3
2.8	1	3	**8.2**	**3**	**1**
			8.3	1	3
3.1	2	3	8.4	1	2
3.2	**1**	**3**	8.5	1	3
3.3	1	3	8.6	1	3
4.1	4	3	**9.1**	**3**	**1**
4.2	**4**	**1**	9.2	2	3
4.3	3	2	9.3	1	3
4.4	3	1	9.4	1	3
4.5	2	3	9.5	1	1
4.6	2	3			
4.7	1	3	**10.1**	**5**	**1**
4.8	1	3	10.2	4	3
4.9	1	1	10.3	2	3
4.10	1	3	10.4	2	1
			10.5	2	1
5.1	5	3	10.6	1	3
5.2	2	3	10.7	1	3
5.3	**2**	**1**	10.8	1	3
5.4	2	3	10.9	1	3
5.5	2	3	10.10	1	3
5.6	2	3			
5.7	1	3			
5.8	1	3			
5.9	1	3			
5.10	1	3			
5.11	1	1			
5.12	1	3			
5.13	1	3			
5.14	1	3			
5.15	1	3			

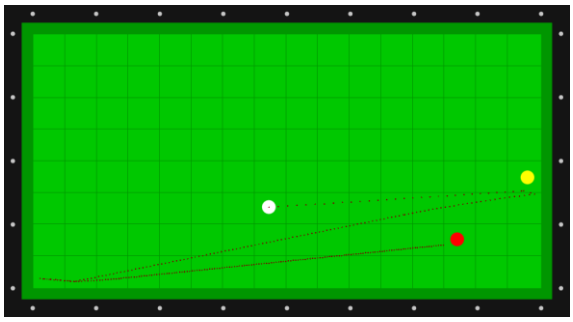
For all the different ball configurations one solution cluster was picked, for illustrative purposes.



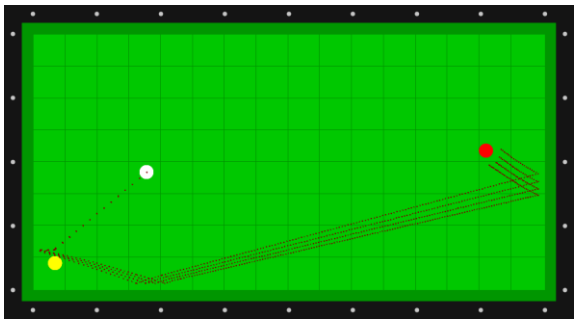
1.4



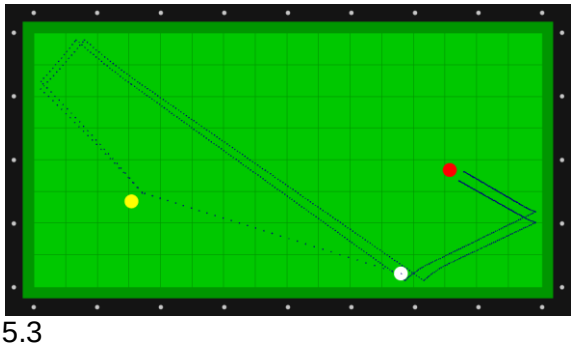
2.1



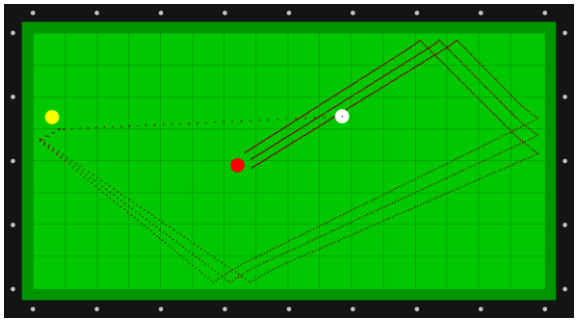
3.2



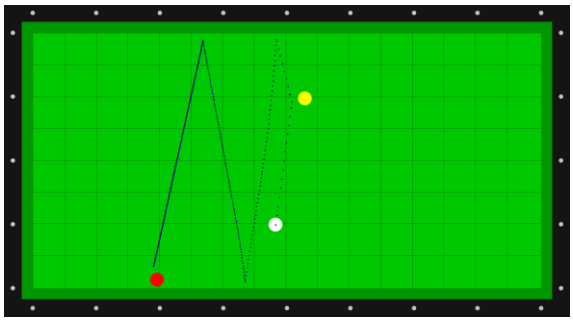
4.2



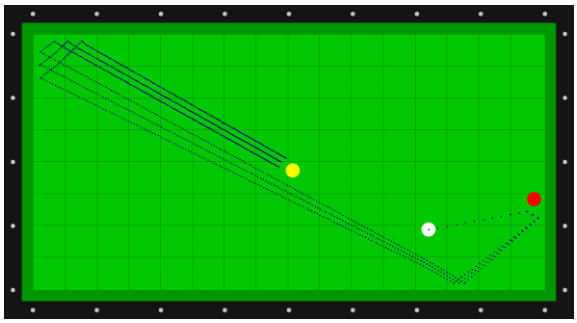
5.3



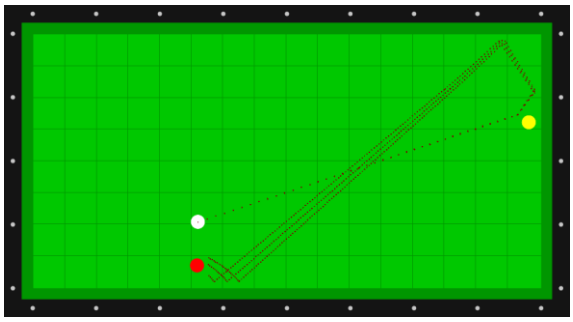
6.2



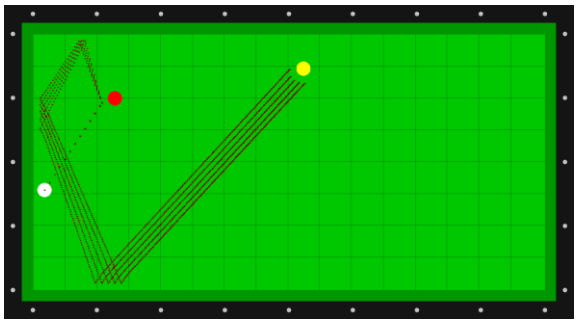
7.2



8.2



9.1



10.1

Appendix F: Preliminary Usertest

The preliminary test was conducted informally, to observe whether the system was stable enough, and as inspiration for the final user test. The results are shown here for the sake of completeness.

During the session the player played 21 different ball configurations. Table 1 shows for each ball configuration the total amount of solutions found, which was chosen, what the outcome of the shot was and optionally further remarks.

BC	Solution/Total	Outcome	Remarks
1	1 / 3	?	
2	4 / 8	?	
3	1 / 18	Just missed	
4	- / 1	-	Own solution
5	- / -	-	No solution found
6	6 / 12	Just missed	
7	1 / 1	?	
8	- / -	-	Standing in front of webcam
9	1 / 1	-	Not a feasible solution, user did not know this and played it anyway
10	2 / 4	Succesful	
11	1 / 2	?	
12	1 / 5	?	
13	- / -	-	No solution found
14	1 / 2	Miss	
15	- / -	-	No solution found
16	1 / 6	?	
17	3 / 3	Miss	
18	1 / 6	Miss	The top solution was not feasible, user played it however
19	1 / 3	Miss	
20	2 / 8	Miss	Missed because white was situated near the cushion, enforcing $\theta > 0$
21	1 / 4	Miss	Kiss prevented success

Table 1: Results for 21 ball configurations

General observations:

The user only twice replaced the balls at their original positions. The first time he marked the ball's shadows and replaced them after a first attempt, and attempted the same solution a second time. The second time he did not mark the balls in front, but replaced the balls approximately at the same location. He then pressed the spacebar again to re-acquire the exact ball positions, and again solve the shot. The same solutions were presented, this time he chose another.

For every ball configuration, the user first navigated through all solutions, and chose one that fitted what he thought was the best one, instead of trusting the first solution to be the best. It was observed that sometimes this resulted in that the solution preferred was not the best solution (according to me).

The player had no problems with figuring out where to hit the cue ball in terms of shot parameters a and b. Also the velocity was not an issue. The player said that he was aiming in most cases with only the impact point on the first cushion in mind.

The player viewed the animation of a shot for about half of the BC's, mostly for BC's he asserted as difficult. Free navigation was used often, to view the shot from the a player's point of view.

Questionnaire:

1. **What is your level of experience?** Few years of experience playing pool billiards, some experience regarding three cushion billiards playing for fun.
2. **Were the recommendations useful for BC's you already knew how to solve?** Yes, while the visualization of the path and the impact points on the cushion were determined more precise.
3. **Were the recommendations useful for BC's did not know how to solve?** For some the VBA gave solutions that I did not figure out myself, and that were successful. In other cases the recommended solution was not feasible.
4. **Would you prefer a training session with the VBA over a training session alone or with another player?** Yes, it provides a more goal-oriented and tentative approach.
5. **Would you prefer a more sophisticated interaction with the VBA, for example by means of an ECA?** No, however some things could improve (see 7).
6. **Was the time needed to solve a configuration too long?** Although a bit long it was not annoying, but when a solution is found, it would be comfortable if the user is alarmed with a beep.
7. **Any general remarks?** A better indication could be given about the thickness at which the cue ball should hit the object ball. Also, when changing view from the screen to the real table, I was often disoriented by the difference of ball positions from my point of view. Also the free navigation, especially the mouse movement should be 'slower'.