Use of Mixed Reality to Accurately Model Billiard Shots Given a Table State

Student Name: Joseph Jenner-Bailey

Supervisor Name: Dr George Alex Koulieris

Submitted as part of the degree of BSc Computer Science to the

Board of Examiners in the Department of Computer Sciences, Durham University
29/04/2021

Abstract —
Context/Background: An increase in the popularity of cue sports, combined with improvements in computer vision and the prevalence of machine learning for the purpose of training in games such as chess and backgammon, make the development of a training system for cue sports both highly desirable and achievable. Previous work has explored each of these components individually with varying success, however, many are inaccessible and none combine them in one system.

Aims: This project aims to match the state of the art with respect to state observation whilst also reducing the dependency on specialist equipment and utilising a structure that makes the incorporation of more complex modelling a natural extension. This is done with the motivation of creating a system that aids in the training of a player.

Method: A stationary RGB-D camera mounted above a pool table is used in combination with a computer for the purpose of table, cue and ball localisation, movement detection and trajectory calculation. This information is transferred via HTTP to a mobile device. The latter provides a secondary camera angle and displays the calculated trajectories via mixed reality. Performance is evaluated based on the efficiency and accuracy of each component along with each players improvement.

Results: All components used for state observation are high in accuracy, resulting in trajectory predictions being correct 91% of the time. Combined with directions from a secondary user, these predictions improve player performance by 3%. Almost all components of the system are also found to be high in efficiency, allowing for play to be conducted without large delays. However, the overlay of trajectories on frames captured from the perspective of the mobile device suffered from misalignment, averaging at 73%.

Conclusions: This project demonstrates that a mobile display-based approach, via mixed reality, is both possible and more accessible than prior work. Although not as accurate in modelling, and resulting player improvement is not as high as expected, a full-sized, level table was not accessible during the current environment, having a significant impact on the results achieved. It is also the first human-operated system to successfully integrate multiple camera perspectives, enabling more complex modelling and shot recommendation to be explored in future work. The requirement of two individuals to operate the system is impractical, however, all code written is platform-independent, allowing for immediate deployment to a mixed reality headset.

Keywords — Mixed Reality, Computer Vision, Playing Surface Localisation, Ball localisation, Ball Classification, Cue Localisation, Movement Detection, Trajectory Calculation

I INTRODUCTION

Cue sports in all forms have existed for several centuries, predominantly being played in and around Europe. However, in the past decade, there has been a surge in popularity in Asia. In China alone, it is estimated that there are now 50 million people that regularly play snooker, along with over 210 million tuning in to watch the Betfred World Championship in 2016. Cue sports are complex, difficult to learn games. This is due to the interplay between multiple skills that must be learnt when beginning. These include stance, cue action, strategy and trajectory calculation. This can be demotivating for new players, with little immediate reward, leading to them quitting early on. The traditional approach is to isolate these skills and train them independently, in a way that is engaging. However, this is rarely achieved, the most common beginner exercise consisting of hitting a single ball up and down the table in a straight line for several hours. Where apparatus is used, it can be not only monotonous but also contrived and impractical, (*Nic Barrow's Cue Action Trainer* 2018). Furthermore, for intermediate and advanced players training has a different purpose, focusing on using all their skills together, in a competitive environment. This is the opposite of what traditional approaches aim for.

Within the past two decades, it has also become increasingly common for the development of training systems for games such as chess, backgammon and go. The latter commonly make use of machine learning to enable a user to play against an artificial opponent in realistic, challenging matches, whilst also providing recommendations for the actions a player makes.

Despite these developments and the increase in popularity, such systems have not been explored to the same extent in the domain of cue sports. This is due to the dependency on a physical medium in order for players to practice. As a result, without the use of unobtainable, highly specialised equipment such as that used in (Long et al. 2004) an opponents actions cannot be acted out. However, the absence of such equipment does not prevent the use of machine learning, such as that in (Leckie & Greenspan 2007), being used instead for the purpose of recommending shots to a player and providing feedback on their actions.

A Nature and Challenges

Prior work has sought to observe and accurately model the state of a table with the use of a combination of RGB and depth cameras. Despite the varying methods used, nearly all share the same fundamental framework. This consists of a sequence of table, ball and cue localisation such that each component can be modelled within a 2D coordinate system. This information is used to simulate and predict the outcome of a players actions. Relaying this information back to the user serves as a valuable educational tool, teaching players how to judge angles, whilst also reducing the complexity of the game and allowing them to assess their technique in isolation. Prediction accuracy's of 97% have been achieved in both (Sousa et al. 2016) and (Alves et al. 2013), working towards solving the first problem outlined.

While effective, finding a balance between being too contrived and impractical or having reduced accuracy and capability with respect to modelling, has proved to be a problem. Furthermore, to integrate the use of more advanced shot recommendation and feedback, such as that found in (Greenspan et al. 2008), requires the use of precise, and in turn, complex modelling, making use of multiple viewpoints to measure factors such as spin. This has not been previously implemented in a human-operated system.

Trajectories calculated by the system are largely dependent on the position of the player's cue, with small alterations in position having a large impact on the outcome. Therefore, for the information displayed to be accurate and reflect the current state of the table, the system must be as low in latency and efficient as possible. This is difficult to achieve without worsening accuracy in other respects, given the high amount of image processing required.

B Objectives and Achievements

The focus of this project is evaluating the *"Use of Mixed Reality to Accurately Model Billiard Shots Given a Table State"*, aiming to build on previous work to create a system that aids in player training, minimising the dependency on technical equipment and facilitating the collection of information required for more advanced shot recommendation and feedback. This is achieved via the following minimum, intermediate and advanced objectives.

The minimum objectives of the project are chosen with the intention of creating a basic system that is usable in a real-life environment. The first of these is the automatic localisation of the playing surface. This will enable irrelevant information to be ignored and so the rest of the system to function appropriately. With this, the balls on the table should be localised and identified, relative to the playing surface. Simple trajectories should then be calculated (irrespective of the player's cue) for the purpose of shot suggestion. An aerial view of the table should be augmented with these trajectories and displayed to the user via a computer display. The resulting system will provide a solid foundation upon which to build the rest of the objectives, ensuring that all components can be tested together at each stage.

While useful this does not observe the players actions and as a result, is not able to model their potential outcomes. To do so requires being able to localise the player's cue. Although making use of methods similar to those used for ball localisation, its movement places an emphasis on efficiency and requires other features such as effective movement detection to be implemented. Therefore, it is an intermediate objective. The display of trajectories via a computer screen is a hindrance for both the player and their coach due to its stationary nature. This means that in most cases a coach is unable to focus on what the player is doing whilst also viewing the display. To improve upon this, an augmented aerial view of the table should be viewable from a mobile device. This will require the implementation of an HTTP/TCP server and client to transfer data between the devices.

The advanced objectives focus on continuing to improve the usability of the system, while also making it suitable for more advanced training. The mobile device should be used to provide a secondary camera feed. Combined with cue localisation, this will allow for factors such as where the cue ball will be struck to be measured and so more advanced modelling to be used. The latter should be integrated into the system with the use of the FastFiz pool physics library, as in (Silva et al. 2017). While an aerial view of the table is beneficial, the requirement of a secondary player or coach to frequently switch between views is still present. Mixed reality via the mobile device should, therefore, be used to display trajectories on the table. Finally, to improve players shot selection, a basic shot recommendation system should be implemented. Heuristics such as the distance and required angle for a given shot should be used to determine its probability of success. The shot with the highest probability then being displayed.

Data flow within the system is highly dependent on the current state of the table, meaning if any component is delayed or inaccurate, erroneous behaviour could occur. Therefore, each objective will be evaluated with respect to efficiency and accuracy. In addition to this, the overall system should also be evaluated with respect to a players quantitative improvement in using the system. The final system met all objectives apart from the integration of the FastFiz library for more accurate modelling. It also achieved a prediction accuracy of 91% and led to a 3% improvement in player performance.

II RELATED WORK

A RGB based approaches

Originally proposed by (Shu Sang 1994) RGB based approaches make use of only visual information to calculate the trajectories of shots. As previously stated, almost all prior systems share the components of calibration (such that a known coordinate system can be used), ball localisation and trajectory calculation. Calibration by placing a set of predefined markers on the table at known locations before the system is used, is first applied in (Long et al. 2004). Although successful in correcting for misalignment, the latter requires the dimensions of the table to be known for its geometry to be accurately modelled. This is overcome by instead using the edges of the table cushions to form a rectangle that bounds the playing surface exactly (Alves et al. 2013). Its edges describe the boundaries of play, while its points of intersection are used for calibration. Ball localisation works on the assumption that all balls in the frame are brighter than the surface of the table. This means that simple thresholding can be used to distinguish them from the table. While effective, this assumption is not likely to hold in the real world due to unpredictable lighting conditions and the many different ball and cloth colours that are commonly used. Most papers perform cue localisation followed by trajectory calculation, modelling the cue as a vector equation. Such localisation can be performed with the use of fiducial markers placed on the cue (Sánchez & Agudelo 2018). However, these alterations are arguably over-engineered. Instead, (Alves et al. 2013) first subtract a reference frame (previously collected when no motion is detected) to identify moving objects within a specified area around the cue ball. The central line of such an object is then assumed to be the line of movement of the cue. However, the use of a region of interest reduces the size of the cue to near to that of a players finger. As a result, the player's hand must always be placed far away from the cue ball, outside the region of interest. This is recognised as poor technique within cue sports. In all instances, a projector above the table is used to display information to the user.

B Head-mounted approaches

Instead of using a stationary RGB camera, one can be mounted to a users head such that its input is similar to what the user sees themselves. This removes the need for a projector and allows for more information about the scene to be collected, such as where the cue ball is struck. However, little research has focused on this area. The movement of the camera means that the depth of the table within the scene is constantly changing. With the use of a continuous video feed, thresholding according to the colour of the cloth can be applied to each frame, the most radially distant points from the centre of the table being used to define its contour (Jebara et al. 1997). Points within this contour that are seen to be different in colour to the cloth are taken to be the balls. In an attempt to improve on this process, (Hsu et al. 2017) use a google glass to collect only a small number of images from different angles to accurately reconstruct a 3d representation of the scene. Each frame is transformed such that it is similar to an aerial view of the table. Stretching as a result of these transforms gives balls an ovular shape. The centre of the area formed as a result of the intersection between all ovals for a given ball is taken to be its location. However, this stretching, along with the use of only 3 frames, is seen to lead to the centre of balls being poorly estimated and as result only localises them correctly 82 $\%$ of the time. In addition to this, it is also stated that the methodology used for table localisation is highly sensitive to lighting and so unsuccessful in some cases.

C Depth and RGB-D based approaches

Although making use of almost identical data flow as that found in RGB based approaches, the use of a depth camera leads to several benefits. Issues relating to lighting found when thresholding to localise balls are rectified by thresholding with respect to depth instead. In addition to this, the textures and patterns present on the cloth or floor can be easily ignored meaning that ball and cue localisation are less susceptible to noise (Rivera Pinzón et al. 2017). Furthermore, when in contact with another ball or table cushion, ball localisation has proven to be unreliable in RGB based approaches, seeing the objects in contact as one entity. By thresholding according to depth, only preserving features of height between 3/4B_w and 7/8B_w, where B_w is the width of a ball, (Sousa et al. 2016) overcome this. This means that points of contact occurring at 1/2B_w are removed, creating a clear separation between objects. However, the use of only depth means that balls can't be identified by colour. To solve this, (Vets et al. 2016) fit balls with unique retroreflective markers, resulting in each being distinct in the depth image. This also makes localisation much easier, allowing for balls to be tracked while in motion, displaying animations at the same time. On the other hand, (Sousa et al. 2016) make use of an RGB-D camera and use different functionality at different stages. This means that the balls and cue can be localised accurately using the depth sensor, while the balls can be identified with the use of the RGB camera. Again, all approaches use a projector to display output information.

D Advanced Trajectory Modelling

Despite the seemingly simple nature of ball movements, changes in physical conditions such as dust, the weight of the balls, the thickness of the cloth and its wear, all lead to significant changes in the trajectories followed. However, (Leckie & Greenspan 2005) and (Leckie & Greenspan 2006) have shown that the movement of balls can be accurately simulated without approximation via an analytical model. With the use of a high FPS camera to track the motion of balls, (Mathavan et al. 2009) attempt to measure required parameters for a given table. These were then applied in (Mathavan et al. 2016), using a robotic pool player to accurately specify force and spin. However, these predictions were shown to not be reliable, with some being off by as much as 25 cm. On the other hand, when used in combination with a mini-max game tree for shot suggestion (Leckie & Greenspan 2007), in the context of a match, such simulation leads to 100% pot success, only fouling 2.5% of shots (Greenspan et al. 2008). Although not the accuracy of the physics simulation itself, the latter does demonstrate that the model is accurate enough to be used to plan effective strategy. This is the ultimate aim of training. The application of this modelling in a non-robotic environment would be ineffective for the purpose of prediction. This is due to the force with which a player will strike the ball being impossible to predict but the impact it has on the outcome being significant. However, for the purpose of shot suggestion - specifying how the ball must be struck for a given outcome - this modelling could be effective for both learning strategy and the relationship between spin, power and resulting trajectories.

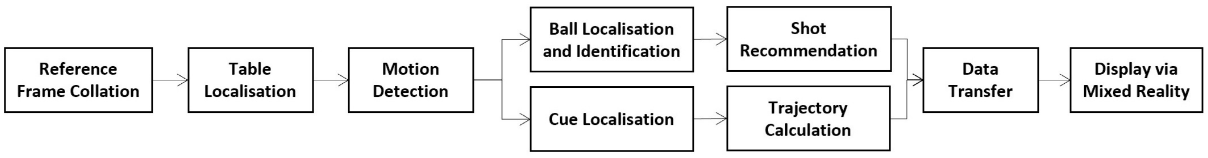


Figure 1: An overview of the system architecture.

III SOLUTION

A Overall Architecture

The project itself can be broken down into multiple sub-components. Each of these is discussed in detail within the following sections. Together, these components form the architecture seen in Figure 1, giving a system similar to that found in (Sousa et al. 2016) and (Rivera Pinzón et al. 2017) but taking inspiration from (Jebara et al. 1997) and (Hsu et al. 2017) to improve both accessibility and the suitability for it to be used in further research. Table, ball and cue localisation are all used in combination to predict the outcome of players shots based on the state of the table and their current positioning. In addition to this, shots are suggested to the player based on their perceived complexity. Motion detection is used to classify the state of play and so determine which processes should take place, and how information should be used. All of this is conveyed to the user with the use of mixed reality via a mobile device after being transferred over an HTTP network.

B Tools Used

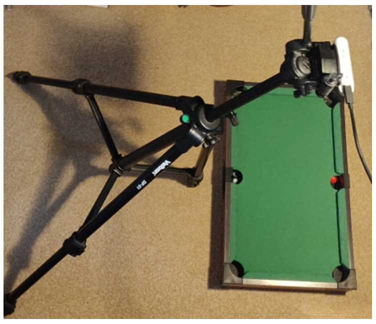


Figure 2: Table and depth camera.

All code has been written in either C++ or C# for both the purpose of efficiency and increased compatibility between libraries used. All tests and measurements were conducted using a Microsoft Surface Book with an Intel Core i7-6600U 2.6 GHz processor and 8 GB of RAM.

The stationary nature of a billiards table meant that the use of a stationary camera would be both discrete and not over-engineered. To get the benefits of both RGB and depth based approaches, an Intel D415 RGB-D camera was used. This was mounted above the table with the use of a tripod (Figure 2) and allowed for both highly accurate depth images (up to 2% in error), along with RGB frames sufficient for the application, to be collected at 100 and 30 fps respectively. To access this information required using the Intel Realsense SDK for C++. This enabled all parameters of the camera, such as depth units and disparity shift, to be altered and tuned at run time to provide the most precise frames possible. The SDK also offered a series of post-processing filters for purposes such as decimation and hole filling. Those chosen were applied to every frame, making use of threading to prevent delays within the main application itself. OpenCV was also used to derive information from these frames, with the use of image processing algorithms such as Canny edge detection, Hough transform and Morphological erosion. Although not the simplest library to use, OpenCV is highly tailored for the purpose of efficiency whilst also providing a great amount of control over each algorithm and its parameters.

A OnePlus 6T with a Qualcomm Snapdragon 845 2.80 GHz processor and 6 GB of RAM was used as the mobile device in the system, making use of both Unity and Vuforia. The extensive amount of development and testing carried out for Vuforia made it both more reliable (than an approach developed in OpenCV) and faster to deploy.

Due to current restrictions, frequent access to a full-size table was not possible. Therefore, a miniature, toy table measuring 0.3m x 0.5m (Figure 2) was used, with both the cue and balls being approximately to scale. However, the player's hand was not. In addition to this, while almost all components such as the cue and balls interacted correctly, the surface of the table was very uneven. This made the overall accuracy of predictions difficult to evaluate, requiring a player to strike the ball harder to compensate for the roll of the table but this then causing them to cue across the shot, resulting in a miss.

C Table Localisation

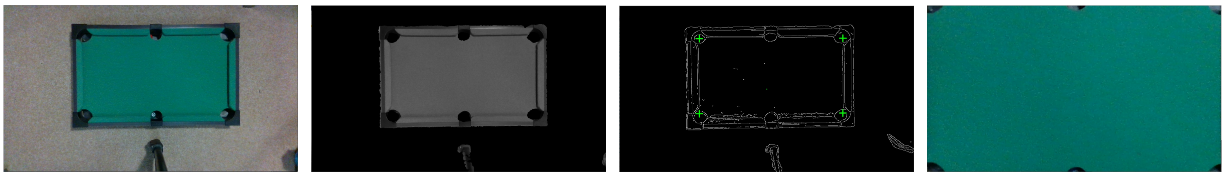

(a) Raw RGB frame. (b) Masked frame. (c) Table corners. (d) Transformed frame.

Figure 3: Frames captured within table localisation.

Given the field of view of the camera and the amount of activity and information within the scene, it was crucial to restrict what was processed. Specifically, only actions directly above the playing surface, bounded by the cushions of the table. To localise this area, before play began the system started by collecting and averaging 100 depth and 100 RGB frames. This gave two frames that were low in noise and that could be used as references throughout the program, providing a baseline to compare against. In the context of table localisation, it was first intended that the average depth frame would be used as in (Rivera Pinzón et al. 2017), starting by thresholding to extract only the cushions of the table. However, the uneven nature of the table meant this was not possible, leading to only parts of the cushions being present in the output. Other papers, such as (Alves et al. 2013), make use of an RGB camera for the purpose of table localisation. However, again this was not practical due to other features in the scene (such as the texture of the carpet the table was placed on) having more distinct edges, making the parameters of following algorithms extremely sensitive. As a result of this, it was chosen to first threshold the reference depth frame such that the entirety of the table was extracted. The output of this was then used as a mask, being AND'd with the RGB frame to give Figure 3b. The latter removed all background patterns from the scene and meant that an RGB based approach could be applied successfully. This entailed applying Canny edge detection followed by a Hough transform. The result was an array of all distinct straight lines found within the frame. Of these, those at a near 90 or 0 degree angle were deemed most likely to be the edges of the table. Taking those most central to the image as the lines between the playing surface and the cushions, the corner pockets of the table could be localised using their intersections (Figure 3c). This served two purposes: the lines identified enabled the detection of collisions with the table cushions, while the locations of the pockets could be used to calculate a homography matrix that would map the playing surface to the full frame size (Sousa et al. 2016). Transforming all future frames using this matrix, as in Figure 3d, meant that other localisation procedures could then be performed within a 2D coordinate system, simplifying the calculation of trajectories. This transform also meant that movement away from the table surface could be ignored, allowing for the use of motion detection to identify when a player was taking a shot.

D Ball Localisation and Identification

(a) Depth frame of balls. (b) Filled ball contours. (c) Balls identified.

Figure 4: Frames captured within ball localisation.

For trajectories to be calculated correctly it was required that the balls were localised with respect to the transformed frames generated during table localisation. This was achieved following the methodology laid out in (Sousa et al. 2016). To start subtraction between the observed and reference depth frames was performed giving the height of objects relative to the surface of the table. Thresholding was then applied to remove all features below $3/4B_w$. This meant that all points of contact between balls or between a ball and a cushion were no longer in the frame. As a result, a contour finder (Suzuki & Abe 1985) could be used to detect each ball individually. The centre of each contour was used as the location of the ball. Before applying the latter, morphological erosion was first used to remove any noise still present after thresholding.

With the balls localised (Figure 4b) it was then necessary to identify the balls detected. This was done by calculating a histogram of RGB values for each ball, using specific masks to identify the region that had to be measured. The combination of channel values that was of maximum frequency was taken to be the colour of the ball itself. The reflective surface of the balls meant that the variance in perceived colour as a result of lighting was significant. However, all ball colours were fundamentally distinct, not shades of the same colour. Therefore, wide bin sizes were used, the middle value of a bin being the colour output if it were the most frequent (Figure 4c). This increased consistency and reduced the effects of lighting. With the colour of the balls identified, this information could be used for the purpose of shot suggestion. Most importantly, irrespective of the game being played was identifying the cue ball. To do so all ball colours were compared and that of maximum value was said to be the white ball.

E Cue Localisation

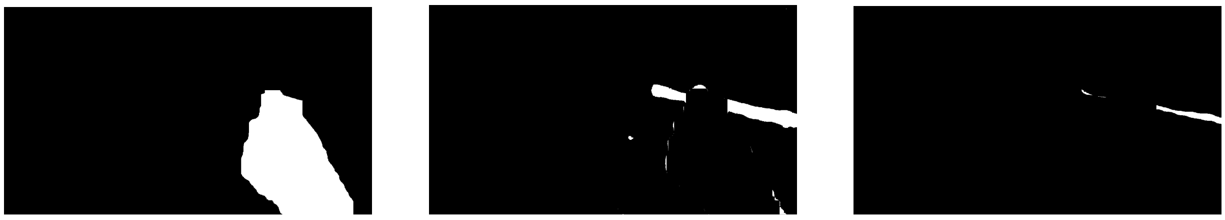

(a) Players hand. (b) Cue with noise. (c) Cue without noise.

Figure 5: Frames captured within cue localisation.

Cue localisation, similar to ball localisation, made use of thresholding, followed by the use of a contour finder to identify the outline of the cue. Balls were removed by subtracting a reference frame of the table (obtained immediately before the cue entered the scene) from that most recently captured. Binary thresholding according to depth left only the cue and the players hand in the frame. With this (Sousa et al. 2016) apply another Hough transform to identify the line of the cue. However, this was not successful, the cue occupying much less space relative to the size of the hand when compared to that in (Sousa et al. 2016). This resulted in inconsistent and random lines. Therefore, to be effective all other known objects needed to be removed from the frame. To do so the method of cue localisation laid out by (Rivera Pinzón et al. 2017) was extended upon. The small width of the cue, relative to that of the player's arm, meant that it could be removed from the scene with the application of morphological erosion. Applying morphological dilation to this output gave a frame with no cue and only the player's hand, approximately the same size as that in the original (Figure 5a). Subtracting this from the original gave the cue, along with some noise around the hand (Figure 5b). The latter was removed by again applying morphological erosion, this time to a much smaller degree (Figure 5c). Another Hough transform could then be used to generate a line that represented the cue. However, even after morphological erosion, the width of the cue meant that the line with most intersections in the parameter space was the diagonal of the cue. As a result, the angle was always inaccurate. Instead, a contour finder was utilised, followed by fitting a rectangle to enclose the contour found. This gave four lines, two of which shared the direction vector v_{de} and were approximately parallel to the line of movement of the cue. The latter were identified using the assumptions that: the cue was always partially located outside of the frame, the cue was always longer than it was wide. The centre of the bounding box then defined the centre of the cue (x_c, y_c), while the vector v_{de} specified the cues direction v_{dc}. For the examples seen in figure 5, this resulted in the trajectories seen in Figure 7c.

F Motion Detection

As previously stated, the process of cue localisation was dependent on image subtraction involving a frame of the table captured before the cue had entered the scene. This could also be defined as the most recent stationary frame obtained, given the system was in motion. To obtain such a frame required the use of movement detection. This was achieved by applying subtraction between consecutive frames and thresholding such that any differing pixels were set to 255. If the total number of pixels set to 255 was above 0.003% of the frame, then the scene was said to be in motion (Alves et al. 2013). This approach was highly sensitive and found to be vulnerable to

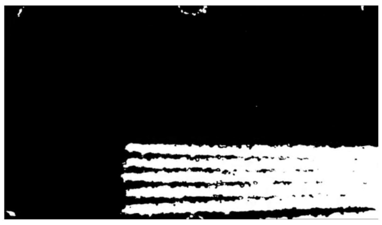


Figure 6: Difference frames for motion detection.

noise. In addition to this, the abrupt changes in state meant that in some instances the system was said to be non-moving while the cue was still in the frame, momentarily paused. This resulted in a reference frame being sampled, with the cue still present and so it not being detected for some localisation's when it did move again. Several improvements were made by (Sousa et al. 2016) in an attempt to solve this. Instead of comparing with only the previous frame, a buffer of the previous N_p = 40 frames were stored. The difference between each of these and the current frame was calculated and summed to give $M_t(x, y)$ (Figure 6).

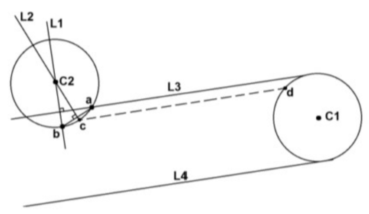
$$M_t(x, y) = \sum_{i=0}^{N_p} |D_t(x, y) - D_{t-i}(x, y)| \tag{1}$$

$M_t(x, y)$ was then thresholded to give $Mb_t(x, y)$, such that $Mb_t(x, y)$ was set to 255 if $M_t(x, y) \geq 0.0005 N_D$ and 0 otherwise, where N_D=255. If more than 0.05$\%$ of pixels in $Mb_t(x, y)$ were equal to 255 then movement was said to have been detected in the frame. In combination with this a counter K was used along with multiple, more descriptive states instead of just two. These included:

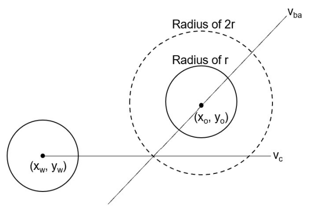
1. **In-Motion:** $K < K_t$ (where $K_t = 25$) and the system was in motion in the previous frame.
2. **Not In-Motion:** $K < K_t$ and the system was not in motion in the previous frame, or $K < K_t$ and the current frame is the first frame to be processed.
3. **Started:** The system was not in motion in the previous frame but is in the current frame.
4. **Stopped:** The system was in motion in the previous frame but is not in the current frame.

K initially started with a value of zero (the minimum possible value), however, different combinations of states in the previous and current frames led to its value being altered. If the system was not in motion and movement was detected then K was incremented, if not then it was decremented. If K reached a value of K_T then it changed from not in-motion to in-motion and K was set to 0. The opposite procedure then applied. If the system was in motion and no movement was detected then K was incremented, if not then it was decremented. If K reached a value of K_T then it changed from in-motion to not in-motion. Ball localisation was then only performed when in the stopped state and cue localisation in the in-motion state. This was found to be much more resistive to noise. However, the use of a buffer of size N_p = 40 was computationally intensive and with little benefit. As a result, a value of N_p = 5 was used which was much more efficient and equally effective. Furthermore, the maximum counter value of $K_t = 25$ reduced sensitivity so much so that cue localisation only began several seconds after the cue had entered the frame and required waiting a significant period before the next shot could be taken, ball localisation needing to occur first. Therefore, a value of K_T = 15 was used.

G Trajectory Calculation



(a) Old calculation - taken from (Alves et al. 2013).



(b) Proposed calculation.

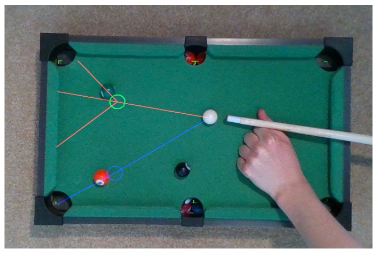

(c) Predicted trajectory of shot.

Figure 7: Visualisation of trajectory calculation.

With all balls localised and the cue modelled by the vector $v_c = (x_c, y_c) + k_c v_{dc}$, ball movements could be predicted. This first required determining if a legal shot was being played and that the cue would make contact with the white ball. Each ball was modelled by the vector equation of a circle $(p - ba) \cdot (p - ba) = r^2$ where p was an arbitrary point on its circumference, ba the location of its centre and r its radius. Substituting the vector equation v_l of a straight line into the equation for a given ball gave the following.

$$\begin{aligned} r^2 &= (v_l - ba) \cdot (v_l - ba) && (2) \\ &= (x_l - x_{ba})^2 + (y_l - y_{ba})^2 && (3) \\ 0 &= (x_{dl}^2 + y_{dl}^2)k_l^2 + 2(x_{dl}(x_l - x_{ba}) + y_d(y_l - y_{ba}))k_l + (x_l - x_{ba})^2 + (y_l - y_{ba})^2 - r^2 && (4) \end{aligned}$$

If positive, the discriminant $dis = b^2 - 4ac$ indicated that the two lines intercepted and that the quadratic formula could be applied

$$k_l = \frac{-b \pm \sqrt{dis}}{2a} \tag{5}$$

where

$$\begin{aligned} a &= x_{dl}^2 + y_{dl}^2 && (6) \\ b &= 2(x_{dl}(x_l - x_{ba}) + y_d(y_l - y_{ba})) && (7) \\ c &= (x_l - x_{ba})^2 + (y_l - y_{ba})^2 - r^2 && (8) \end{aligned}$$

For the case of calculating the point of contact between the cue and the cue ball, $ba = (x_w, y_w)$ and $v_l = v_c$. Conditional on $k_c > 0$, the point of contact was given by the minimum possible value for k_c. Given the absence of more advanced modelling and that the line of the cue had greater influence over the line of cue ball movement than where it is struck, all movement was projected from the centre of the cue ball via the line $v_w = (x_w, y_w) + k_w v_{dc}$. It was then required to check whether the cue ball would collide with any other balls on the table. To do so, (Alves et al. 2013) project from both sides of the cue ball, check for a collision with every object ball oba for each line and then calculate any resulting trajectories as in figure 7a (the line L2 being that of resulting object ball movement). However, it was found that it was possible to half the number of calculations required, only projecting from the centre of the cue ball (Figure 7b). At a point of collision, the distance between the two ball centres was exactly $2r$. Therefore, in calculating the intersection of the line projected from the cue ball with the circle of radius $2r$, centred at the point oba, it was possible to determine whether the balls would collide. This was achieved by setting $v_l = v_w$, substituting $2r$ in for r and using $ba = oba$. Feeding the minimum output value for k_w back into the equation v_w gave the location of the cue ball at the point of collision. The direction vector of the object ball movement was then given by $v_{doba} = oba - ((x_w, y_w) + k_w v_{dc})$ and in turn its overall line of movement as $v_{oba} = oba + k_{oba} v_{doba}$. To reflect the expected cue ball movement as a result of this collision the normal vector of v_{oba}, v_{cn} was also calculated. The direction of v_{cn} was set so as to ensure $cos(v_{cn} \cdot v_{dc}) > 0$. All lines were then scaled for the mobile display and eventually drawn to both screens (Figure 7c).

H Shot Recommendation

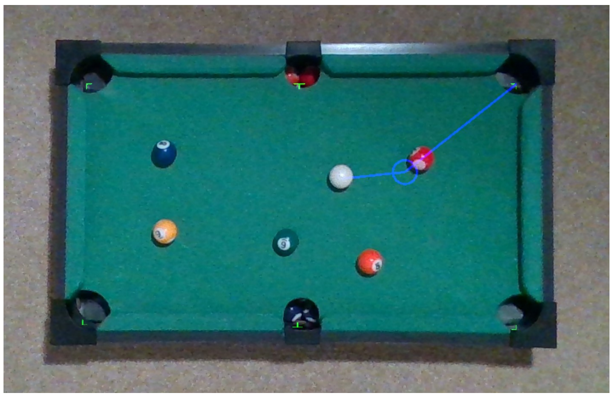


Figure 8: Recommended shot by the system.

For players new to cue sports, the difficulty of a shot can be hard to judge. This can be problematic as when beginning it is desirable to train on simple shots specifically. These make identifying errors in a players technique easier to observe and analyse. Therefore, it was chosen to focus on implementing a greedy algorithm for shot recommendation, suggesting shots that were measured to be most likely to result in a pot. With all balls localised and taking pockets to be in the standard configuration relative to the transformed frames, every pairing of an object ball oba and a pocket po was iterated over. For each pair, the vector $v_{doba} = po - oba$ was calculated. Using methods outlined in ‘Trajectory Calculation’, working in reverse order, the required cue ball movement v_{dw} to result in the object ball movement v_{doba} was calculated. With these vectors 3 parameters were calculated: the distance between the cue ball and object ball $||v_{dw}||$, the distance between the object ball and the pocket $||v_{doba}||$ and the angle α between v_{dw} and v_{doba}. Feeding these into the equation outlined in (Chua et al. 2002)

$$\Delta = \frac{||v_{dw}|| \cdot ||v_{doba}||}{\cos^2\alpha} \tag{9}$$

gave an estimate of the complexity of the shot. Intuitively, this defined simple shots as those where the required distance for balls to travel, and the angle between trajectories were both small. Having calculated Δ for all possible pots (only those with $90^\circ < \alpha \leq 180^\circ$ being considered), the shot with minimum value was chosen. It’s corresponding vectors v_{dw} and v_{doba} were later displayed to the user (Figure 8).

I Data Transfer

The large differences in the outcome as a result of small changes made by the player meant it was essential that all information displayed was as recent as possible. All previous approaches made use of a wired connection and proved to be successful. However, to be effectively used for mixed reality in a headset or mobile device, required being able to move freely around the table. A wired connection was, therefore, not suitable and a wireless approach was taken instead. An HTTP server, written in C# was implemented on the user's mobile device, while an HTTP client was implemented in C++ on the computer being used for processing data from the RGB-D camera. This data could then be dynamically sent from the computer to the mobile device while play was taking place. To ensure that this transfer was both low in latency and reliable, a persistent TCP/IP protocol was used in combination with threading on both the client and server sides. In the event that a connection was lost the client immediately attempted to reconnect. Steps were also taken to reduce the quantity of data that was required to be transferred over the network for each frame. While sending an augmented aerial view of the table would have allowed for accurate localisation of the mobile device in 3d space and so arguably more precise overlay of trajectories, this was found to be too intensive to keep latency low. Instead, only the coordinates of line endpoints were transferred over the network in a json data structure. These were then used for drawing the trajectories.

J Mixed Reality

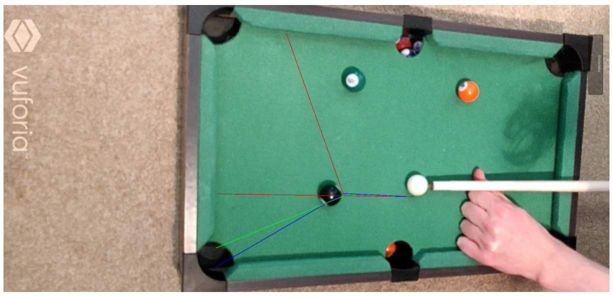


Figure 9: Trajectory displayed via mobile.

For trajectories to be overlayed required knowing the transform from the aerial view of the table to that of from the perspective of the mobile. This allowed for trajectories to be mapped appropriately such that they appeared on the table surface to the user. Determining this transform could be done in several ways. The most obvious was to identify points of correspondence between the mobile and aerial views, using feature points detected via an algorithm such as SURF, determining which of those corresponded using RANSAC and then calculating the homography matrix that maximised the number of corresponding points. However, this would require one device to have access to both perspectives and so the transfer of frames over HTTP. As previously discussed, this was too intensive to keep latency low. An alternative was to make use of fiducial markers, such as QR codes, placed on the cushions of the table. Localising these from the perspective of the mobile would then be sufficient to calculate the homography matrix. However, the use of QR codes required physically altering the scene, in a way not commonplace in the real world. Instead, Vuforia's object scanner was used to pre-scan the table. This enabled the table itself to act as a fiducial marker within the scene and could be identified regardless of the state of the balls on its surface. Furthermore, the use of Vuforia in combination with Unity meant that in making the lines drawn a child of the scan, all necessary transforms were performed automatically. Red lines were used to show the predicted initial and resulting cue ball movements. On the other hand, a green line was used to show the object ball movement and blue lines were used for the recommended trajectory. An example can be seen in Figure 9.

IV RESULTS

This section focuses on the results of the system and how they were collected to facilitate the evaluation of player improvement, prediction accuracy and efficiency. All tests were conducted within the same environment, with only minor changes in lighting conditions.

A Player Improvement

With respect to player improvement, the success of the system was measured by its impact on players potting and shot selection. The former was easy to measure, counting the number of successful pots with and without the use of the system. However, a players shot selection was conditional on their opponent's performance. Therefore, despite a player having good shot selection, their opponent may have performed better. As a result, they would still have poor pot success. 8 ball pool is a cue sport renowned for being heavily focused on potting, with frequent table clearances, as opposed to tactical play. Therefore, to control for the actions taken by an opponent, they are removed from the game. Performance was then measured by the number of shots taken in order to pot all balls on the table. Three different configurations were tested: no predictions or shot suggestion, trajectory prediction but no shot suggestion and trajectory prediction along with shot suggestion.

Table 1: Player improvement with use of the system.

Configuration	**Number of Balls Potted**	**Number of Shots**	**Accuracy (%)**	**Improvement (%)**
Unassisted	15	52.4	29	-
Prediction	15	47.2	32	3
Prediction and Suggestion	15	48.5	31	2

All tests were carried out for four different players, along with another individual acting as the coach, using the mobile device to give directions as to the angle and positioning. All fowl shots were included, the cue ball being handled according to standard rules. However, in the instance where a ball was also potted during a foul shot, the potted ball was returned to the table (as near to the black spot as possible) and the pot was not counted. Table 1 shows that player performance was low in all configurations. Despite this, the number of shots that resulted in a pot is also seen to increase in both instances where the system was used. In the case, where only predictions were used to augment the scene this increase is 3%. However, with the addition of shot recommendation this improvement is seen to be less, only increasing by 2%.

B Prediction Accuracy

B.1 Modelling

The accuracy of the system can be measured with respect to both its overall outcome (the accuracy of its predictions) and the performance of each of its individual components. All measurements - except for those of table localisation - were made using samples from recordings taken of participants during the collection of player improvement data. For measurements with binary outcomes, data from the first 30 shots of each player was used. On the other hand, for continuous data, measurements from the first 5 shots of each player was used. In the case of table localisation, 20 separate runs of the system were repeated. Conducting measurements in this way ensured that changes in the accuracy of components for each player would be reflected in the overall accuracy of the predictions made.

The predictions made by the system specify, based on the initial cue ball trajectory, the lines the object and cue balls will follow after they collide. The error present was, therefore, calculated as the difference in the angle and positioning between the prediction made and the resulting trajectory. All shots counted were played with no spin as modelling did not account for this. Furthermore, due to both player inaccuracy and the uneven nature of the table, only shots in which the cue ball followed the correct initial trajectory and no roll was seen, were used.

Table 2: Accuracy of each system component.

Component	**Number of Tests**	**Accuracy (%)**
Cue Ball Trajectory	20	90
Object Ball Trajectory	20	93
Table Localisation	20	94
Ball Localisation	20	74
Cue Ball Identification	120	98
Cue Localisation	20	65
Movement Detection	120	87

Predictions made by the system had a low amount of variance in inaccuracy between measurements for each player, instead displaying a consistent amount for each shot. Error averaged at 10% for the trajectory of the cue ball and only 7% for the trajectory of the object ball.

Table 2 also captures the accuracy of each solution component. The error in table localisation was measured using the distance each corner was off by in both the horizontal and vertical directions, relative to the width and length of the table, giving 94% accuracy. Ball localisation required first identifying the contour of a ball - giving its radius - and then the centre of the contour itself. The error in both of these was reflected by the number of green pixels (the colour of the cloth) present within and the number of non-green pixels located outside, of the circle drawn at the identified centre with the calculated radius. Much like the inaccuracy in predictions, the degree to which components were inaccurate was never extremely significant but consistently present throughout, almost constant for a given player. For trajectories to be calculated it was also essential that the cue ball was correctly identified. Counting the number of cases where the cue ball was incorrectly identified yielded an accuracy of 98%, making it the most accurate component. The error in cue localisation required the manual identification of 3 midpoints for each sample:

one at the tip of the cue, one at the butt and one at the centre. With these, the error in terms of positioning and angle was calculated, achieving 65% accuracy. It is important to note here that although measurements of cue localisation were not taken at necessarily the same stage in each shot, in all instances the cue was effectively stationary. This was to ensure that any inaccuracy seen was as a result of the localisation method used, rather than a delay due to inefficiency in the system. The accuracy in the classification of motion at each stage was deemed to be a product of the algorithm used itself, not hindered by its efficiency. Therefore, it has been included as a measurement of the systems ability to model the state of the table. The latter required measuring the fraction of each shot in which either the cue was present within the frame but no trajectory was displayed, or the cue was not present but the reference RGB frame was seen not to have been updated. Averaging over all shots gave an accuracy of 87%.

B.2 Display

Table 3: Accuracy of trajectory display.

Component	Number of Tests	Accuracy (%)
Overlay Alignment	20	73

Despite the accuracy in modelling, this information was useless if its display to the user was inaccurate. Given that all calculations were performed relative to the playing surface, providing that this region was correctly aligned, all other errors could be said to be a result of modelling. The latter was measured similarly to the error in table localisation. Frames from the users mobile were overlaid with a rectangle, calculated to have the same size and positioning as the playing surface. The relative error in the positioning of each corner was then measured and averaged to give the result seen in Table 3. While the size and orientation appeared to be consistently correct, alignment itself averaged an accuracy of only 73%. Furthermore, although once identified table tracking was successful, initial localisation on the users mobile could be dependent on lighting conditions and as a result took up to 10 seconds.

C Efficiency

Table 4: Average processing time for crucial components.

Component	Number of Tests	Average Time (μs)
Initial Processing	1000	58.664
Cue Localisation	1000	2.173
Data Transfer	1000	91.381

The dynamic nature of play placed importance on the efficiency of the system, delays ultimately leading to the display of inaccurate information to the user. Cue localisation and the transfer of data from the computer to the mobile device both had the potential to have severe impacts on the system if inefficient. Therefore, both of their respective run times were measured (in microseconds) and averaged. The former was achieved by calculating the difference between a timestamp taken before the process began and another taken after it finished. The initial processing in the system consisted of all components that took place prior to cue localisation. Therefore, it is unsurprising to see in Table 4 that its average time is relatively high at 58.664 microseconds. Cue localisation, on the other hand, was faster than expected at only 2.173 microseconds. However, the largest contributor to any delay within the system, taking 91.381 microseconds on average, was the transfer of data from the computer to the mobile device.

V EVALUATION

A Strengths and Limitations

The results in Table 1 show that, although not by a considerable amount, the system does consistently improve players capabilities. Baseline performance, without the use of the system, was low. This was due not only to most participants being beginners but also the difference in conditions and scale making the difficulty of all shots harder. In particular, the roll of the table made a significant number shots impossible to execute consistently. However, the improvement seen was most prominent in those shots that could be potted consistently. This combined with the high accuracy of predictions, 90% and 93%, (when a roll was not present) would suggest that were a level, to scale table used, a much greater variety of shots would be possible, increasing both baseline performance and the relative improvement seen as a result of the use of the system. Although it has not been possible to measure the long term impact of shot suggestion on player performance, the reduced improvement seen with its use (relative to when only trajectory predictions were displayed) would suggest that a greedy approach is not suitable. Instead a more sophisticated algorithm is required, such as that in (Greenspan et al. 2008), focusing on the use of long term planning to reduce the average difficulty of shots instead.

Error present in both cue and object ball trajectory predictions can be largely attributed to inaccuracy in ball localisation, specifically the radius of balls being measured to be too small. Reduction in the threshold depth value did not resolve this, instead, introducing large amounts of noise. This high sensitivity was a result of the balls used being 2x smaller in radius than those used in standard play. Therefore, the difference in depth between a balls midpoint and the table surface was half. This will have led to a significant increase in the relative error as a result of inaccuracy in the depth sensor. Again using to scale equipment would have reduced these effects. Despite the error in cue localisation being significant, this was largely due to the bounding box calculated being accurate in rotation but not width. As a result, while the position of the cue centre was inaccurate, it's direction vector was valid. Given that the system was designed not to model factors such as spin, the projection of the direction vector from the centre of the cue ball naturally corrected for this. Hence its small impact on the overall prediction accuracy. Cue ball identification was also found to be high in accuracy, successful in 98% of cases. All errors were a result of a combination of overly bright lighting being used and stripped balls laying white region upwards. A potential improvement to this process would be to also consider the range of values present in the histogram of colours for each ball. That of the white being smaller when compared to most other balls. In addition to this, in most cue sports such as snooker and English 8 ball pool, the causes of these errors would not be present, since only solid coloured balls are used. Lighting conditions were also responsible for errors in table localisation, uneven lighting leading to some cushions/edges being more defined than others and shadows being misinterpreted. This was prevented with the use of a secondary light source mounted above the table. However, due to conditions, it was not possible to mount this directly above the table, resulting in the errors mentioned. Almost all tables in a professional and club environment are fitted with lighting directly above the table. As a result, it is likely that in such an environment table localisation would be more consistent. The high accuracy in movement detection at 87% is another strength. However, the error present cannot be ignored, with cases where movement detection did fail having a large impact on the rest of the system. Such cases were a result of little motion occurring for a prolonged period, with only a small portion of the cue present in the frame. Therefore, the difference between frames was almost equivalent to that of naturally occurring noise. This meant that increasing sensitivity to motion did not improve the accuracy in detection, only leading to stationary states being deemed to be moving. To solve this problem would require being able to identify the cue within the frame, independently of its motion and the area that it occupies. The highly regular shape of both the cue and all patterns on its surface, may make this difficult. The display of trajectories on the mobile device achieved an accuracy of 73%, lower than that when viewed from the computer. However, the nature of this inaccuracy meant it could be easily identified and corrected for by the user, all line points being shifted by an equal amount, as can be seen in Figure 9. The cause of misalignment was due to the low number of feature points on the table. The simplest approach to resolve this would be with the use of QR codes placed on the table cushions. However, as previously discussed this would be best avoided. An alternative would be the use of OpenCV to develop a more tailored algorithm.

The efficiency in cue localisation was high only taking 2.173 microseconds on average. As a result, overlays displayed on the computer screen were accurate and suffered little delay. Data displayed on the mobile device was also accurate with respect to time, allowing the user to provide accurate guidance throughout. However, the delay in transmission, 91.381 microseconds on average, combined with the time taken for initial processing, 58.664 microseconds on average, did mean that some patience was required on the users part. Network conditions, although not outstanding, were more than suffice suggesting both the data structures and algorithms used for transmission require optimisation.

B A Comparison with the State Of The Art

Little data exists for comparison with key papers such as (Rivera Pinzón et al. 2017) and (Jebara et al. 1997), providing no quantitative results. However, both (Alves et al. 2013) and (Sousa et al. 2016) do provide data and achieve shot prediction accuracy's of 97%. While higher than the 91% achieved in this paper, both were conducted on full-size tables with (Sousa et al. 2016) not modelling the interactions between balls, only the trajectory of the white ball assuming no collisions. In addition to this, (Sousa et al. 2016) achieves only 91% with respect to white ball classification, lower than the 98% achieved in this paper, suffering from the same problems with respect to lighting and ball orientation. Neither paper gives the specific accuracy of processes such as table and ball localisation, only their success rate in execution. However, (Hsu et al. 2017) classify ball localisation as correct if it is within 2.5cm of the true ball location and achieve a an accuracy of 82%. This is significantly worse than what is presented here, the system achieving 100% ball localisation when quantified in this way. This difference in accuracy is a result of only a non-stationary viewpoint being used, emphasising the importance of a stationary, aerial view of the table. In addition to this, the system presented reduces vulnerability to highly textured flooring during table localisation and improves the systems ability to distinguish between a players hand and the pool cue. These are both problems that were found to be present in the methodology of (Sousa et al. 2016). Furthermore, the use of a mobile device to display calculated trajectories to the user has also reduced the on dependency specialist equipment found in previous papers and makes it the first human operated system of its kind to integrate multiple viewpoints.

C Approach

This project was structured so as to first build a basic system in which all functionality was present to at least some extent. This meant that it could be fully tested throughout, as improvements and extensions were integrated. Such an approach also meant that in the event of an objective not being met, the system could still be successfully tested and evaluated. While these benefits were found to be true, the latter also meant that development was potentially slower and that the quantity of work present in the final project may have been lessened as a result. In addition to this, the originally rigid structure chosen did not account adequately for unexpected problems. Most notably were those as a result of the Intel Realsense D415 camera used, with little documentation and sometimes unpredictable behaviour. Although a more Agile approach was later adopted, if done again using such a framework from the beginning would likely prove beneficial.

VI CONCLUSIONS

This project has resulted in a system that matches the state of the art in most capacities including ball and table localisation and white ball identification, also achieving 91% with respect to prediction accuracy and improving player performance by as much as 3%. Unfortunately due to current restrictions, these findings cannot be validated on a tournament standard table. However, we are confident that in such conditions, performance would only be improved.

The system created has also succeeded in removing the need for components such as a projector for the purpose of displaying trajectories. As a result, it is both less cumbersome and more accessible, proving that the *"Use of Mixed Reality to Accurately Model Billiard Shots Given a Table State"* is both viable and more favourable in some respects. Furthermore, the integration of a mobile device makes it the first human-operated system of its kind to utilise multiple camera feeds for the purpose of localisation.

Although the use of the FastFiz library was not achieved, it is hoped that the additional data available as a result of this added perspective will be used for more complex modelling in future work. This combined with methods outlined in (Leckie & Greenspan 2007) could be used to produce recommendations that account for future shots and improve players decision making. The need for cue localisation independent of its movement has also been highlighted and is arguably the most important problem currently faced. All code written is also platform-independent, allowing for its deployment on headsets such as the Microsoft Hololens. This would remove the need for a secondary user and allow the player to see trajectories directly on the table itself. This would likely lead to a further improvement in performance.

References

Alves, R., Sousa, L. & Rodrigues, J. (2013), Poolliveaid: Augmented reality pool table to assist inexperienced players, *in* '21st Int. Conf. on Computer Graphics, Visualization and Computer Vision, Plzen, Czech Republic, 2013.'.

Chua, S. C., Tan, A. W. C., Wong, E. K. & Koo, V. C. (2002), Decision algorithm for pool using fuzzy system, *in* 'iCAiET 2002: Intl. Conf. AI in Eng. Tech', pp. 370–375.

Greenspan, M., Lam, J., Godard, M., Zaidi, I., Jordan, S., Leckie, W., Anderson, K. & Dupuis, D. (2008), 'Toward a competitive pool-playing robot', *Computer* **41**(1), 46–53.

Hsu, C.-C., Tsai, H.-C., Chen, H.-T., Tsai, W.-J. & Lee, S.-Y. (2017), Computer-assisted billiard self-training using intelligent glasses, *in* '2017 14th International Symposium on Pervasive Systems, Algorithms and Networks 2017 11th International Conference on Frontier of Computer Science and Technology 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC)', pp. 119–126.

Jebara, T., Eyster, C., Weaver, J., Starner, T. & Pentland, A. (1997), Stochasticks: Augmenting the billiards experience with probabilistic vision and wearable computers, *in* 'Digest of Papers. First International Symposium on Wearable Computers', pp. 138–145.

Leckie, W. & Greenspan, M. (2005), 'Pool physics simulation by event prediction 1: Motion transitions', *J. Int. Comput. Games Assoc.* **28**, 214–222.

Leckie, W. & Greenspan, M. (2006), 'Pool physics simulation by event prediction 2: Collisions.', *ICGA Journal* **29**, 24–31.

Leckie, W. & Greenspan, M. (2007), Monte-carlo methods in pool strategy game trees, *in* H. J. van den Herik, P. Ciancarini & H. H. L. M. J. Donkers, eds, 'Computers and Games', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 244–255.

Long, F., Herland, J., Tessier, M.-C., Naulls, D., Roth, A., Roth, G. & Greenspan, M. (2004), Robotic pool: an experiment in automatic potting, *in* '2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)', Vol. 3, pp. 2520–2525 vol.3.

Mathavan, S., Jackson, M. & Parkin, R. (2009), 'Application of high-speed imaging to determine the dynamics of billiards', *American Journal of Physics - AMER J PHYS* **77**, 788–794.

Mathavan, S., Jackson, M. & Parkin, R. (2016), 'Ball positioning in robotic billiards: A nonprehensile manipulation-based solution', *IEEE/ASME Transactions on Mechatronics* **21**(1), 184–195.

Nic Barrow's Cue Action Trainer (2018). [Last Accessed: 2021-04-15] Available from: https://www.thesnookergym.com/cue-action-trainer.

Rivera Pinzón, D. M., Gonzalez, E. & Gomez, E. (2017), 'Billiard game parameters calculation using a depth camera for augmented reality applications', *Contemporary Engineering Sciences* **10**, 1171–1180.

Sánchez, H. & Agudelo, F. E. (2018), 'Sistema de realidad aumentada aplicado al juego de billar', *Siembra CBA Magazine* **2**(1), 93–100.

Shu Sang, W. (1994), Automating Skills Using a Robot Snooker Player, PhD thesis, Bristol University.

Silva, A., Malhadas, D. & Ramires Fernandes, A. (2017), Poolimmersion: A realistic physically-based billiards simulation, *in* '2017 24º Encontro Português de Computação Gráfica e Interação (EPCGI)', pp. 1–8.

Sousa, L., Alves, R. & Rodrigues, J. (2016), 'Augmented reality system to assist inexperienced pool players', *Computational Visual Media* **2**, 183–193.

Suzuki, S. & Abe, K. (1985), 'Topological structural analysis of digitized binary images by border following', *Comput. Vis. Graph. Image Process.* **30**, 32–46.

Vets, T., Nijs, L., Lesaffre, M., Moens, B., Bressan, F., Colpaert, P., Lambert, P., Van de Walle, R. & Leman, M. (2016), 'Gamified music improvisation with billiart: a multimodal installation with balls', *Journal on Multimodal User Interfaces* **11**, 25–38.