Strategic play for a pool-playing robot

Thomas Nierhoff, Kerstin Heunisch, Sandra Hirche

***Abstract*— Playing pool is not a trivial task for an autonomous robotic system: In order to excel, the tight coupling of accurate perception, planning and highly dynamic while precise manipulation is required. To investigate scientific challenges arising from this tight coupling requirement an anthropomorphic robotic system has been developed, which is capable of playing pool. Focusing in this paper on the planning part, a well-calibrated pool simulator is developed in order to predict the outcome of possible strokes. An optimization based framework applying a discounted return term is presented for suggesting an optimized next stroke by planning several strokes ahead. Different to existing planners so far, our planner considers the player's and opponent's skill. Therefore, strategic considerations (e.g. safety shots) can now be evaluated depending on the outcome of the next strokes for both player and opponent.**

I. INTRODUCTION

People have built automata for entertainment and gaming purpose for centuries. Like the Jaquet-Droz automata, those machines performed a straightforward task execution without situation assessment as they lacked the ability to interact in a reasonable way with their environment. In contrast, recent robots are able to perform complex tasks based on situation-aware judgement of their next action, even being able to improve their skill for specific tasks through multiple execution. In this context, game and entertainment robots represent an interesting robotics research playground. Typically games are still structured in the sense that only a limited number of situations can occur while still requiring the full perception-cognition-action loop working in a tightly coupled manner. This makes the development of suitable robotic entertainment systems viable already today. As a result, this type of robotic systems certainly play a strong role on how robots are perceived by the society today and how they will be integrated in our lives in the future. Examples of successful integration of robotic systems are: Table soccer [1], soccer [2] and ping pong [3]. Another game is pool. Instead of fast reactions/decision making - an aspect where humans still excel humanoid robots - pool requires high accuracy when executing a stroke and good planning capabilities. We are confident a humanoid robot is superior to an intermediate human pool player in both aspects as it can be calibrated well and has got enough computing power and time to simulate hundreds of strokes before deciding which one to execute. The problem of developing a descent pool robot is split up into three processing steps: Perception, planning and action. In this paper the focus lies on the planning part, dealing with the question how to select the optimal next stroke depending on a set of previously simulated ones. It is shown that the most relevant aspects like player's and opponent's skill, tactical aspects like safety shots and the desired farsightedness during play are handled using a unified frameword.

Part of this work is the result of the Bachelor thesis by Kerstin Heunisch. Thomas Nierhoff and Sandra Hirche are with the Institute of Automatic Control Engineering (LSR), Faculty of Electrical Engineering, Technische Universität München, D-80290 München, Germany {tn, hirche}@tum.de.

In order to predict the outcome of a stroke, a model of the physical behaviour of pool is needed. In [4] and [5], the physical aspects occurring at pool are displayed. Aimed at the amateur pool player who wants to improve his game style by understanding the physical background behind pool, [6] is best-suited. Special aspects are covered in [7] drawing special attention on the movement of a ball on the table. The collision between two balls is discussed in [8]–[11]. So is the contact between cue tip and ball occurring at a stroke in [12] where they focus on spin. Finally, [13] only looks at the stroke of the white ball in the presence of friction with the table.

Concerning developed pool simulators, most existing programs like [14] are designed for entertainment purposes, making it difficult to fit the underlying physical model to one specific table. One exception is PoolFiz [4], an open source simulator designed for a real-world pool robot and used for yearly pool simulation tournaments. Here, the movement of a ball on the pool table under the effect of friction is solved analytically, making classical numerical integration obsolete. Even if this is advantageous from a computational point of view - computation is speeded up by several magnitudes - it becomes difficult to change the underlying physical model as the corresponding PDEs have to be solved again analytically.

Coming to planners, the most decent ones are participating at the International Computational Billiards Championships. Over the years Monte-Carlo sampling based planners like [15]–[17] are proven to be the most robust ones in the presence of noise. In contrast, a fuzzy logic planner is presented in [18] together with a pool robot. A novel idea is demonstrated in [19] and [20]: Instead of simulating shots randomly and selecting the most promising ones, they embed their pool simulator into an optimization routine, therefore being able to fulfil advanced goals like pocketing multiple balls with a single stroke. When making a choice among two or more possible shots, one has to evaluate the difficulty of each stroke. For this purpose [19] proposed two different options to measure the stroke difficulty: One is based on the allowed deviation angle of the white ball whereas the second one directly takes the distance cue ball - object ball, distance object ball - hole and cut angle into account.

Though there exist good planners for pool, they show limitations when planning several strokes ahead: So far, a

lot of important variables like the strength of the player and opponent, tactical decisions or the far-sightedness during position play are considered just insufficiently.

The contribution of this paper is therefore a framework tackling the planning problem for a robotic system when playing pool. It consists on the one hand of a physics-based pool simulator with parameters identified through a real pool table. On the other hand a sampling-based planning strategy is adopted, simulating the next most likely strokes for both, the robot and the opponent. By using a cost function based on a discounted return function, the outcome of several strokes ahead is represented as a single scalar value. This results in an optimization problem over a search tree. Under certain assumptions, we show how a fast solution can be obtained through dynamic programming. The key aspect of this paper is how an optimal next stroke is derived taking both the player's and opponent's skill as well as tactical aspects like safety shots into consideration.

The remainder of this paper is organized as follows: Sec. II provides a brief overview over the existing anthropomorphic robot used for pool. In Sec. III the underlying equations for the implemented pool simulator are constituted. Sec. IV displays how the simulator is fitted to a real table. Last, Sec. V concentrates on the question how an optimized next stroke is suggested for the player.

II. Pool-Playing Robotic System

The entire system consists of a mobile robot with a pair of 7-DoF anthropomorphic arms, see [21]. All low level platform and arm controllers are based on Simulink and compiled using the Real-Time Workshop. The entire system is designed for real-time control at a framerate of up to 1000Hz using a RTAI real-time kernel. For fast data exchange across different computers, a self-developed shared memory called Real-Time Database (RTDB) in combination with the Ice middleware is used, see [22]. Optionally, ROS-based programs can be connected to the RTDB for any not time-critical task. Vision information comes from a ceiling-mounted camera approximately 2.5m above the table, tracking the trajectory of all balls on the table at a frame rate of 30Hz and distinguishing between the white ball, the black ball, striped and solid balls. The robot is able judge whether it can pocket a ball with respect to physical constraints and move autonomously around the table to execute the desired stroke. One challenge is a precise and fast 3D detection of cue and robot with respect to the pool table. Having mounted only one camera, fusioned data from robot-mounted laser rangefinder, arm pose data and ceiling camera is used for an accurate cue positioning behind the white ball. In order to execute a fast stroke without violating hardware constraints regarding motor current and joint velocity, the arm configuration is optimized before the stroke, see [23]. Out of 463 test strokes for random ball positions on the table, the robot succeeds in pocketing around 50% of all balls, and with a success rate of around 80% for the simplest shots (approx. 25% of all balls). Differing from existing pool robots (The Snooker Machine, Deep Green) the entire system is optimized for proper planning and execution speed, making it possible to play also against humans at reasonable speed. However, without planning at least one stroke ahead, the resulting position of the white ball is often disadvantageous for the robot. For planning ahead, an accurate billard simulator is required to model the outcome of a set of possible strokes.

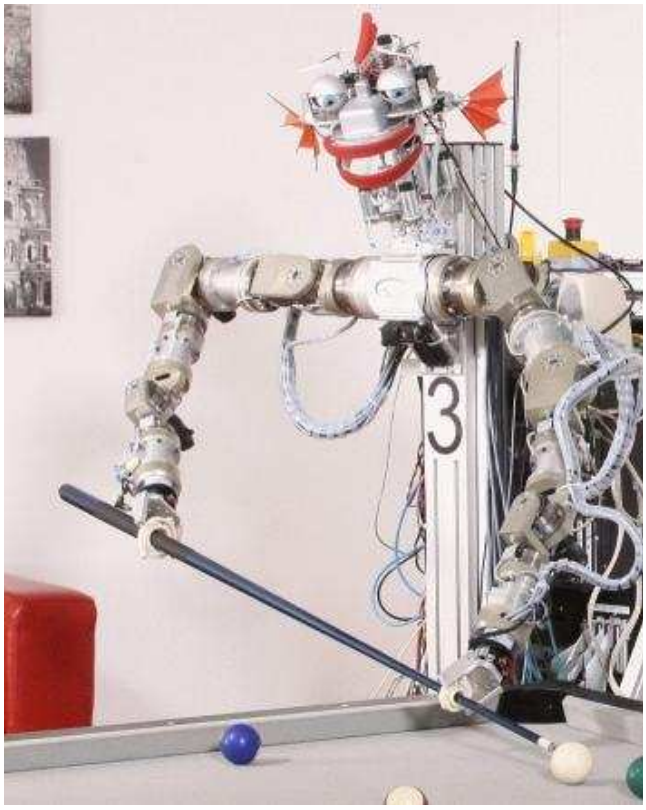

Fig. 1. Picture of the robotic platform for pool playing

III. Physical Model of the Simulator

This section covers the theoretical background for the implemented billard simulator. Most calculations are based on the results presented in [5]. In total, there are five situations one has to examine at pool: At the beginning of each move, the white ball is hit by the cue, see Sec. III-A. After that, a usual short phase of sliding on the table, see III-C, is followed by a rolling phase, see Sec. III-B, till the ball stands still. In addition, collisions with either another ball, see Sec. III-D, or a cushion, see Sec. III-E, may happen.

A. Stroke

If a ball gets hit by the cue, the cue transfers an impulse $\mathbf{p}$ in the x-z-plane on the ball in point A. Given the three parameters α for displacement of the cue along the y-axis causing side spin, β for displacement of the cue along the z-axis causing top spin and γ for the angle between z-axis and cue, the resulting velocity of the center-of-mass $\mathbf{v}_C$ and

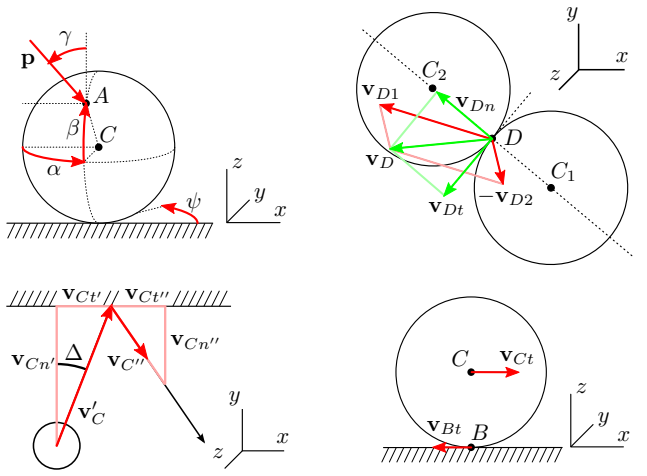

Fig. 2. Possible states of a pool game: Stroke (top left), ball-ball collision (top right), ball-cushion collision (bottom left), rolling/sliding on the table (bottom right)

the rotational velocity $\boldsymbol{\omega}$ is calculated as

$$\mathbf{v}_C = \frac{|\mathbf{p}|}{m}\begin{pmatrix}\sin\gamma\\0\\0\end{pmatrix}, \tag{1}$$

$$\boldsymbol{\omega} = \frac{|\mathbf{p}||\mathbf{r}_{CA}|}{m}\begin{pmatrix}\sin\alpha\cos\beta\cos\gamma\\-\cos\alpha\cos\beta\cos\gamma+\sin\beta\sin\gamma\\\sin\alpha\cos\beta\sin\gamma\end{pmatrix} \tag{2}$$

m denotes the mass of the ball, $\mathbf{r}_{CA}$ the vector from C to A. For arbitrary stroke directions on the table, $\mathbf{v}_C$ and $\boldsymbol{\omega}$ need to be rotated by an angle ψ around the z-axis.

B. Rolling on the Table

A ball rolling on the table is characterized by a tangential speed of the contact point B with the table equal to zero $\mathbf{v}_{Bt} = 0$. In this case

$$\boldsymbol{\omega}_t = -\frac{1}{|\mathbf{r}_{CB}|^2}\,\mathbf{r}_{CB}\times\mathbf{v}_C, \tag{3}$$

for the tangential rotational velocity $\boldsymbol{\omega}_t$ in the x-y-plane and $\mathbf{v}_C$ holds. The corresponding rolling friction force $\mathbf{f}_{roll}$ is calculated as

$$\mathbf{f}_{roll} = -\frac{\mathbf{v}_C}{|\mathbf{v}_C|}mg\lambda_{ra} - \mathbf{v}_C mg\lambda_{rr}, \tag{4}$$

with g for the gravitational constant, λ_{ra} for the velocity-independent rolling friction coefficient and λ_{rr} for the velocity-dependent rolling friction coefficient.

If the ball has a normal rotational velocity $\boldsymbol{\omega}_n$ along the z-axis, this velocity will be unaffected by the rolling friction force $\mathbf{f}_{roll}$. Therefore, [5] suggests to calculate the normal angular acceleration $\boldsymbol{\alpha}_n$ depending on the radius of the contact area between ball and table ρ as

$$\boldsymbol{\alpha}_n = -\frac{\boldsymbol{\omega}_n}{|\boldsymbol{\omega}_n|}\frac{2}{3\Theta}mg\rho\lambda_{sa}. \tag{5}$$

The inertia Θ of a ball depending on the radius of a ball r_B is calculated as $\Theta = 0.4mr_B$.

C. Sliding on the Table

A ball is sliding on the table if the translational velocity of the contact point with the table $\mathbf{v}_{Bt} \neq 0$. In this case, the sliding friction force $\mathbf{f}_{slide}$ is

$$\mathbf{f}_{slide} = -\frac{\mathbf{v}_{Bt}}{|\mathbf{v}_{Bt}|}mg\lambda_{sa} - \mathbf{v}_{Bt}mg\lambda_{sr}, \tag{6}$$

with λ_{sa} denoting the velocity-independent sliding friction coefficient and λ_{sr} the velocity-dependent rolling friction coefficient. Similar to the previous case the ball is rolling, (5) holds.

D. Ball-Ball Collision

The collision between two balls is approximated by a partially elastic hit in normal direction, see [10]. In addition, some impulse in tangential direction is transferred due to friction between the two ball in the moment of contact. The transferred impulse $\mathbf{p}_n$ in normal direction on ball one is

$$\mathbf{p}_n = \frac{1+e_{bb}}{2}m\mathbf{v}_{Dn}, \tag{7}$$

and the transferred impulse in tangential direction $\mathbf{p}_t$ is

$$\mathbf{p}_t = |\mathbf{p}_n|\lambda_{bb}\frac{\mathbf{v}_{Dt}}{|\mathbf{v}_{Dt}|} \tag{8}$$

For (7) and (8), $\mathbf{v}_{D1}$ denotes the contact point velocity of ball one in a reference coordinate frame. The same accounts for $\mathbf{v}_{D2}$ and ball two, see Fig. 2. The relative velocity between the two balls can then be calculated as $\mathbf{v}_D = \mathbf{v}_{D2} - \mathbf{v}_{D1}$. It consists of a normal component $\mathbf{v}_{Dn}$ in $\mathbf{r}_{C1C2}$ direction and a tangential component $\mathbf{v}_{Dt}$ orthogonal to it. The resulting velocity $\mathbf{v}_{C_1'}$ after the collision is

$$\mathbf{v}_{C_1'} = \frac{\mathbf{p}_n+\mathbf{p}_t}{m} + \mathbf{v}_{C_1} \tag{9}$$

If the tangential inpulse $\mathbf{p}_t$ is not zero, the transferred angular momentum $\mathbf{L}_t$ on ball one is calculated as

$$\mathbf{L}_t = \mathbf{r}_{C_1D}\times\mathbf{p}_t, \tag{10}$$

and the resulting angular velocity $\boldsymbol{\omega}'$ after collision as

$$\boldsymbol{\omega}' = \frac{\mathbf{L}_t}{\Theta} + \boldsymbol{\omega} \tag{11}$$

E. Ball-Cushion Collision

In [5] an in-depth analysis of the ball-cushion collision is presented. In this paper a simplified model is used, assuming the ball is rolling on the table immediately before and after cushion contact. Under the assumption of no sidespin $\boldsymbol{\omega}_n = 0$, the contact with a cushion is modeled by an partially inelastic hit along the normal direction as

$$\mathbf{v}_{Cn''} = -\mathbf{v}_{Cn'}e_{bc}, \tag{12}$$

$$\mathbf{v}_{Ct''} = \mathbf{v}_{Ct'}, \tag{13}$$

$$\mathbf{v}_{C''} = \frac{\mathbf{v}_{Cn''}+\mathbf{v}_{Ct''}}{|\mathbf{v}_{Cn''}+\mathbf{v}_{Ct''}|}|\mathbf{v}_{C'}|\,\delta_v \tag{14}$$

where (14), e_{bc} denotes the coefficient of restitution depending on the velocity before cushion contact $|\mathbf{v}_{C'}|$ and the input angle Δ. In addition, the velocity after cushion contact $|\mathbf{v}_{C''}|$ is reduced by a factor δ_v relative to the velocity before cushion contact.

IV. Parameter Determination

For the model presented in Sec. III, the following parameters need to be determined: λ_{ra}, λ_{rr}, λ_{sa}, λ_{sr}, λ_{bb}, e_{bb}, e_{bc}, δ_v, ρ, m, r_B. Out of the 11 unknown physical parameters, the following 8 are evaluated:

1) Ball mass m and radius r_B
2) Rolling friction coefficients λ_{ra} and λ_{rr}
3) Sliding friction coefficients λ_{sa} and λ_{sr}
4) Cushion parameters e_{bc} and δ_v

The other ones - namely ρ, λ_{bb} and e_{bb} - cannot be determined due to too imprecise equipment and other superimposed, more dominant effects. Here, values based on the results of [5] and [10] are used.

The rolling friction coefficients λ_{ra} and λ_{rr} are measured by tracking the trajectory of a single ball rolling on the table with the ceiling-mounted camera. Exemplary results through piecewise linear regression are shown in Fig. 3. For velocities greater $0.05\frac{m}{s}$ the rolling friction is nearly independent of the ball velocity. For velocities smaller $0.05\frac{m}{s}$ it decreases almost linearly. The calculated friction coefficients are shown in (16). To reduce the influence of outliers due to measurement noise like inaccurate camera trigger timings, only data points within a certain distance to the x-axis are evaluated, see the cone of blue dots in Fig. 3. The measured data is still noisy as the measured friction coefficient is close to zero and therefore hard to determine. The exact parameters

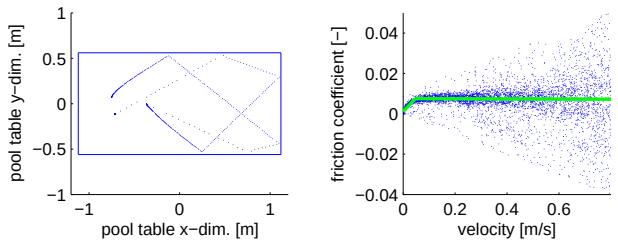

Fig. 3. Determining the rolling friction coefficient. Shown on the left side is some part of the trajectory obtained by the ceiling-mounted camera. The right side displays the calculated rolling friction coefficient with the corresponding piecewise linear regression result. For better readability, only every tenth sampling point has been plotted

of the regression line are

$$\lambda_{ra}(\mathbf{v}_C) = \begin{cases} 0.0011 & \text{for } |\mathbf{v}_C| \leq 0.0473, \\ 0.0075 & \text{for } |\mathbf{v}_C| > 0.0473, \end{cases} \quad (15)$$

$$\lambda_{rr}(\mathbf{v}_C) = \begin{cases} 0.1350 & \text{for } |\mathbf{v}_C| \leq 0.0473, \\ -0.0004 & \text{for } |\mathbf{v}_C| > 0.0473 \end{cases} \quad (16)$$

To determine λ_{sa} and λ_{sr}, a small rack with defined weight standing on three fixed pool balls is designed as shown in Fig. 4. The entire rack is pulled by a linear axis with constant velocity. The force exerted on the linear axis due to the sliding friction of the rack is measured using a JR3 force/torque sensor. Different speeds between $0.01\frac{m}{s}$ and $1\frac{m}{s}$ are evaluated, see Fig. 4. The noise of the data results from small vibrations during pulling caused by the elastic string connecting vehicle and linear axis. Similar to rolling friction,

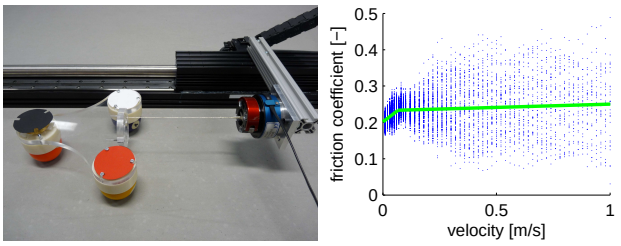

Fig. 4. Determining the sliding friction coefficient. A rack standing on three fixed pool balls is pulled over the pool table by a linear axis at various predefined speeds. The calculated sliding friction coefficient and the piecewise linear regression lines are shown on the left

sliding friction seems to be constant over a wide area of velocities and almost linear dependent of the velocity for velocities smaller $0.05\frac{m}{s}$. Results obtained through piecewise linear regression for the two cases are

$$\lambda_{sa}(\mathbf{v}_{Bt}) = \begin{cases} 0.2014 & \text{for } |\mathbf{v}_{Bt}| \leq 0.672, \\ 0.2322 & \text{for } |\mathbf{v}_{Bt}| > 0.672, \end{cases} \quad (17)$$

$$\lambda_{sr}(\mathbf{v}_{Bt}) = \begin{cases} 0.4763 & \text{for } |\mathbf{v}_{Bt}| \leq 0.672, \\ 0.0180 & \text{for } |\mathbf{v}_{Bt}| > 0.672 \end{cases} \quad (18)$$

Overall, 1217 recorded cushion contacts are used to evaluate the physical behaviour of the cushion. Two parameters are determined: The coefficient of restitution of the cushion and the velocity of the ball after cushion contact. Both parameters are evaluated depending on the velocity of the ball before cushion contact and the input angle. As Fig. 5 shows, e_{bc} depends heavily on both parameters: The velocity of the ball and the input angle. The regression planes for e_{bc} and δ_v are

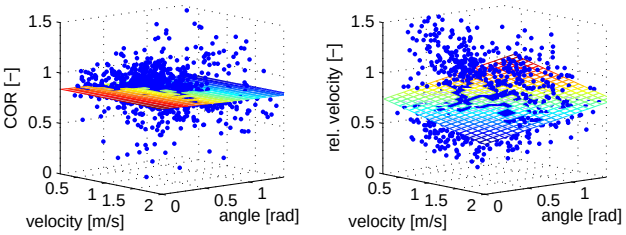

Fig. 5. Measuring the cushion parameters. On the left side, the coefficient of restitution e_{bc} of the cushion with corresponding regression plane for 1217 cushion contacts is shown. On the right side, the relative velocity δ_v with calculated regression plane is displayed. Note that the obtained results for the relative velocity are based on very noisy data and should therefore been taken with a pinch of salt

described by

$$e_{bc}(\mathbf{v}_C, \Delta) = 0.9331 - 0.0278|\mathbf{v}_C| - 0.1051\Delta, \quad (19)$$

$$\delta_v(\mathbf{v}_C, \Delta) = 0.7366 - 0.1094|\mathbf{v}_C| + 0.1698\Delta \quad (20)$$

V. Stroke Planning

Concerning planning, the question is how a recommendation for the next stroke is stated depending on the analyzed outcome of a set of simulated strokes. The previous sentence leads to the two major topics covered in this section: First, which stroke to simulate? And second, how to analyze the outcome of a stroke systematically?

A. Stroke Selection

When executing a stroke, one can vary α, β, γ, ψ and $\mathbf{p}$. In order to reduce complexity and being able to make predictions that can be transferred on a real pool table, only two parameters are varied, $\mathbf{p}$ and ψ. Implicitly it is assumed the ball is hit centrally, i.e. $\alpha = 0$, $\beta = 0$ and $\gamma = \frac{\pi}{2}$. For this case, an optimal stroke angle ψ_o is determined for each object ball - hole combination such that the object ball is pocketed in the mid of each pocket, see [19]. In addition, there are two angles $\psi_{\pm h}$ marking the maximal allowed deviation to the left/right of ψ_o the ball is just pocketed, as shown in Fig. 6. If the angular deviation is bigger, one is unable to pocket the object ball. Two other angles $\psi_{\pm e}$ denote the angular deviation the object ball is just hit without making a foul. Similar limits exist for the stroke intensity: An lower limit p_{-e} the white ball is just fast enough to pocket the object ball and an upper limit p_{+e} depending on the robot's maximal achievable velocity. As a result, one has only to consider parameter tuples in a 2D search space between $\psi_{\pm e}$ and $p_{\pm e}$ assuming the robot is at least able to hit the object ball and programmed well enough to approximate p_{-e}. All calculations can be extended to cover also possible obstacles (other balls) on the table.

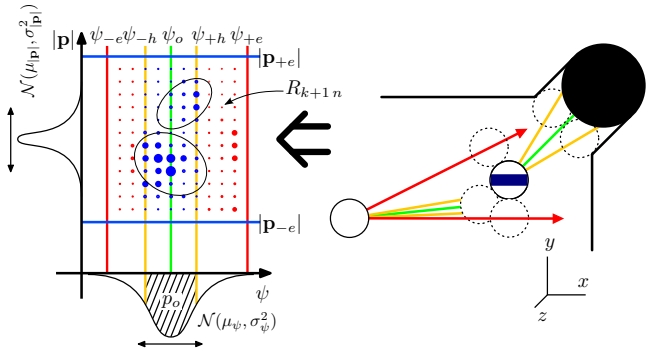

Fig. 6. Calculation of $R_{k\,n}$. Assuming the robot is "skilled" enough, only sampling points within $\psi_{\pm e}$ and $|\mathbf{p}_{\pm e}|$ are considered. Each sampling point has got an assigned value $R_{k+1\,n}$ denoting the expected return for the robot from the next stroke on. Big blue dots indicate a high return for the player (positive) whereas big red dots mark a high return for the opponent (negative). Then $R_{k\,n}$ is calculated based on an optimization over $\mu_{|\mathbf{p}|}$ and μ_ψ. The optimization cost function is the average of all sampling points weighted with $\mathcal{N}(\mu, \sigma^2_{|\mathbf{p}|})$ and $\mathcal{N}(\mu_\psi, \sigma^2_\psi)$. Under certain assumptions, μ_ψ can be set to a fixed value and $\mu_{|\mathbf{p}|}$ can be approximated with the center of clustered $R_{k+1\,n}$ values (black ellipse). In addition, p_o based upon the ruled area marks the percentage to pocket the current ball if the outcome of a stroke depends only on the angle

The robot's (and opponent's) skill is determined by a set of recorded sampling shots with measured angular deviation relative to ψ_o and measured intensity deviation relative to a predefined value. Both values are approximated by normal distributions $\mathcal{N}(\mu_\psi, \sigma^2_\psi)$ for the angular deviation and $\mathcal{N}(\mu_{|\mathbf{p}|}, \sigma^2_{|\mathbf{p}|})$ for the impulse deviation. Neglecting the stroke intensity, the percentage p_o to pocket an object ball is calculated based on the approximated normal distribution of the angular deviation and $\psi_{\pm e}$.

B. Planning Ahead

In order to analyze the strokes, skilled human players also consider the most likely situations for the next n strokes beside the current situation. A situation on the table at time k in the future is described as a state X containing the positions of the N balls on the table, where X is the Cartesian product $X = x_1 \times x_2 \times \ldots \times x_N$, where the sets x_i are subsets of $\mathbb{R}^2$ giving the space of possible positions of the i-th ball on the table. Then w_k is the cost associated with the state X_k determining how good or bad the situation is for the player. Similar to reinforcement learning problems as described in [15] and [24] we model this effect with an discounted finite-horizon return function with discount factor δ:

$$R_{0n} = \sum_{k=0}^{n} \delta^k w_k, \quad 0 \leq \delta < 1 \tag{21}$$

Let us assume an arbitrary finite sequence of strokes resulting in a sequence $(X_0, \ldots, X_n)$ and consequently a sequence of states returns $(w_0, \ldots, w_n)$. The return w_k is defined as +1 if the player pockets a ball and 0 otherwise. For the opponent, the reward is -1 if he pockets a ball and 0 otherwise. Because the outcome of each stroke is probabilistic, one has to weight every possible stroke outcome with its corresponding probability depending on the impulse $\mathbf{p}$ and angle ψ and integrate over all probabilities $p(\mathbf{p}, \psi)$:

$$R_{0n} = \sum_{k=0}^{n} \left[\delta^k \iint w_k \, p(\mathbf{p}, \psi) \, d\psi d\mathbf{p} \right], \quad \delta \in \mathbb{R}_+ \tag{22}$$

For $\delta = 1$, there is a proper physical understanding of R_{0n}, as a R_{0n} value greater zero indicates the player is going to pocket more balls over the next n rounds than the opponent from a statistical point of view. The opposite applies for R_{0n} values smaller than zero. Consequently, the resulting goal for this planning problem is formulated as

$$\underset{\mathbf{p}, \psi, \text{stroke sequence}}{\text{maximize}} \; (R_{0n}) \tag{23}$$

The first step in solving (23) is to discretize the problem and to set up a search tree of depth n where each node of depth $k = 1 \ldots n$ represents the situation on the table after k strokes. When sampling the 2D search space of each node, one also has to consider the possible outcome for the opponent. When simulating a robot stroke of depth k in the tree and the shot parameter are within $\psi_{\pm h}$ and $|\mathbf{p}_{\pm e}|$, the robot can continue playing the next round. Iteratively, one gets new search spaces of depth $k + 1$. The same accounts for the opponent of depth $k + 1$ if the shot parameters are within $\psi_{\pm h}$ and $\psi_{\pm e}$ and $|\mathbf{p}_{\pm e}|$. If the maximum depth n is reached, we do not plan another step ahead. In this case, $R_{n\,n}$ is a single scalar number representing the maximal percentage to pocket a ball p_{maxP} for the robot respectively p_{maxO} for the opponent.

Subsequently, an optimization algorithm is used varying the stroke intensity $\mu_{|\mathbf{p}|}$ and stroke angle μ_ψ variables of each node of the tree in order to solve (23). This problem is intractable within reasonable for large search trees as there

are $\mathcal{O}((n_s)^n)$ variables to optimize for n_s sampling points for each table and a search tree of depth n.

However, experiments show that under certain assumptions a good approximate solution is obtained. Except for extremely easy shots or very skilled pool players and $\delta \lesssim 1$, μ_ψ is close to ψ_o after optimization. This abandons the need to optimize for μ_ψ. The physical interpretation is that except for those situations the primary goal is to pocket the object ball in the most simple manner, thus by aiming at the center of the pocket. In addition, optimization for $\mu_{|\mathbf{p}|}$ is bypassed by assigning a value to each sampling point in the search space of a node of depth k equal to $R_{k+1\,n}$, where $R_{k+1\,n}$ represents the return function for the subtree starting from the specific sampling point on. The resulting search space with sampling points and corresponding values $R_{k+1\,n}$ is then clustered using the OPTICS algorithm [25], taking only a fraction of the best $R_{k+1\,n}$ values into account. Every found cluster with $\mu_{|\mathbf{p}|}$ set to the center of the cluster is compared with each other, making it possible to calculate $R_{k\,n}$ as the average return based on $\mathcal{N}(\mu_\psi, \sigma_\psi^2)$, $\mathcal{N}(\mu_{|\mathbf{p}|}, \sigma_{|\mathbf{p}|}^2)$ and $R_{k+1\,n}$ via backward induction. After having iterated to the root of the search tree, $R_{0\,n}$ will return the optimal values μ_ψ and $\mu_{|\mathbf{p}|}$ for the next shot.

C. Safety Shots

Pool offers another element of strategic gameplay called safety shots. After having announced a safety shot, the opponent will continue with the next shot in any case. Looking at the search tree, because we now have to consider two different possibilities for each situation, the number of nodes in the tree grows from $\mathcal{O}((n_s)^n)$ to $\mathcal{O}((n_s)^{2n})$. This is the brute-force approach. A more elegant solution first analyses each situation and then decides dynamically whether one also wants to consider a safety shot. In general, there are two reasons for announcing a safety shot in round k:

1) There is no chance for the robot of executing a legal stroke after the next stroke. We reduce this situation to the case the robot can't hit any object ball.
2) The situation after the next shot will be bad for both robot and opponent.

With respect to R_{0n} and the game logic, not announcing a safety shot in the first case will result in comitting a foul the next round, thus letting the opponent place the white ball two rounds ahead whereever he wants. As this is assumed to be optimal for the opponent, it is always better to announce a safety shot. For the second case, announcing a safety shot means expecting a high return in the long run with the disadvantage of letting intentionally execute the opponent the shot after the next round. Here, the expected return $R_{k+1\,n}$ for any subtree starting from depth $k+1$ without announcing a safety shot is close to zero and a safety shot may lead to better results.

VI. Results

Measuring μ_ψ, σ_ψ, $\mu_{|\mathbf{p}|}$ and $\sigma_{|\mathbf{p}|}$ for any player (human or robot) is achieved by a set of recorded sample strokes. Regarding the stroke intensity, this can be adjusted and measured precisely for the robot. On the other hand, human players are told to pocket a set of balls with "light intensity", "normal intensity" and "high intensity". As the standard deviations for all three cases were quite similar, a single value has been used instead. However, this fact has to be considered if the planner is ought to be used for humans.

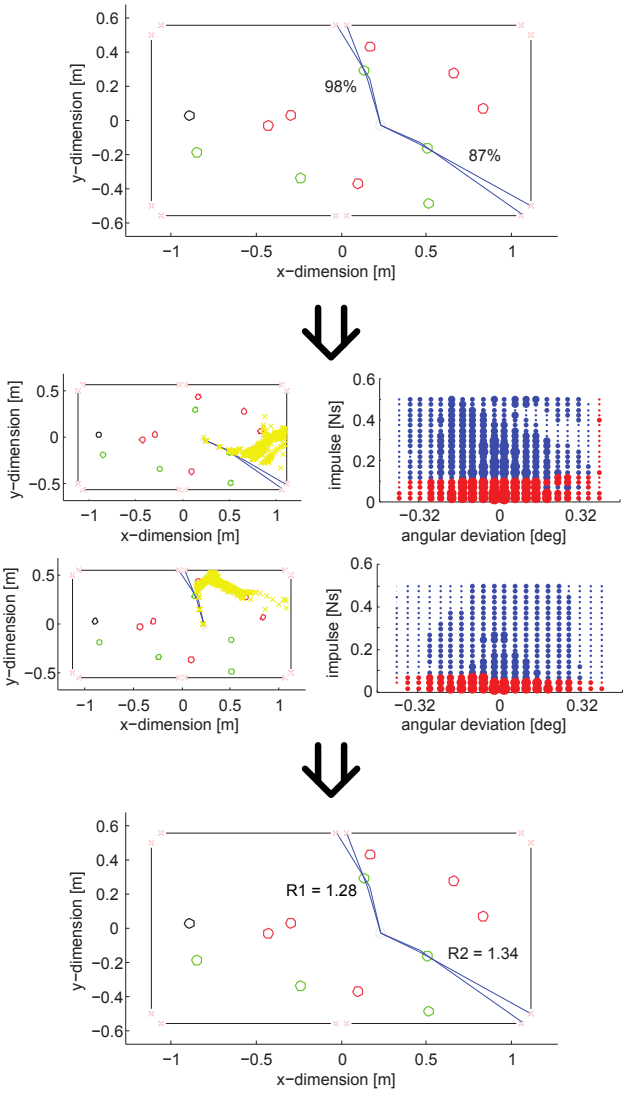

Fig. 7. Planning results for a sample situation. Shown on top is the current situation with the success rates for a greedy approach only considering the current situation on the table. The next four images illustrate the calculations the final return function $R_{0\,1}$ is based on. Yellow crosses mark the position of the white ball after each sample stroke. Last, the lowermost image shows the expected return for both strokes when planning one stroke ahead

Planning results are shown in figure 7 for a planning depth of 1 and $\delta = 1$. Shown is the effectiveness of the proposed planner over a greedy approach considering only the current situation. The two easiest shots have success rates of 98% and 87% as shown in the topmost image. Thus, a greedy approach will select the shot with 98% success rate. However, when simulating sample shots for the two possibilities as shown in the next four images, the situation changes: On the left side, the final positions of the white ball

after having executed a set of sample shots are marked with a yellow cross. The right side displays the search space that is used to calculate $R_{0\,1}$. Because the tree depth equals 1, every $R_{1\,1}$ value represents the maximal percentage to pocket a ball in the next round for the player (blue dots) if he pockets the object ball without a foul or for the opponent (red dots). The lowermost image shows the resulting $R_{0\,1}$ values for both strokes, indicating that it is advantageous to pocket the ball with the lower success rate as it has the higher expected return.

VII. Discussion

As many aspects of pool are already analyzed in detail, most topics of this paper can be compared in a broader context. To a large extent our pool simulator model is founded on well established physical principles. For the ball-cushion collision, the resulting equations are extended to model the physical effects for the given pool table better. It is not our intention to model every physical effect as accurate as possible (e.g. spin), rather we concentrate on the dominant, measurable and best understood effects. Differing from other pool simulator implementations, this planner uses numerical integration instead of an analytical solution for better modeling of discontinuous friction effects (as shown in Sec. IV) and easier model improvement whenever new measurements are available.

With respect to the planner presented in this paper, the discounted return approach in combination with modeled stroke angle and impulse deviation is promising as parameter tuning is reduced to choose a suitable discount factor δ that determines the look-ahead horizon of the planning strategy. One still existing problem covers end game situations where one or two players have only few balls left. Because the planner does not depend explicitly on the number of balls as shown in (22), a potentially desired shift in the playing strategy cannot be accomodated adequatly.

VIII. Conclusion

This paper presents a framework for a robotic pool playing system with advanced planning capabilities. Particular emphasis is on the interactive playing capability with a human opponent requiring for a tight coupling and real-time capability of perception, planning, and stroke execution. Based on a unified approach, we are able to handle tactical decisions during a pool game including the consideration of different player strengths. Whereas the current implementation is tied to the pool game, the general idea can be extended to various episodic games in order to include human capabilities into the robot's tactical considerations.

Acknowledgement

This work is supported in part within the DFG excellence research cluster *Cognition for Technical Systems - CoTeSys* (`www.cotesys.org`).

References

[1] T. Weigel and B. Nebel, "Kiro - an autonomous table soccer player," in *In Proc. Int. RoboCup Symposium 02*. Springer-Verlag, 2002, pp. 119–127.
[2] http://www.robocup.org.
[3] Y. H. Zhang, W. Wei, D. Yu, and C. W. Zhong, "A tracking and predicting scheme for ping pong robot," *Journal of Zhejiang University - Science C*, vol. 12,no.2, pp. 110–115, 2011.
[4] W. Leckie and M. Greenspan, "An event-based pool physics simulator," *ACG 2005, Taipei, Taiwan, September 6-9*, pp. 247–262, 2006.
[5] I. Han, "Dynamics in carom and three cushion billiards," *Journal of Mechanical Science and Technology*, vol. 19, no.4, pp. 976–984, 2005.
[6] R. Shepard. (1997) Amateur physics for the amateur pool player. [Online]. Available: http://www.tcbilliards.com/articles/physics.shtml
[7] A. Salazar and A. Sanchez-Lavega, "Motion of a ball on a rough horizontal surface after being struck by a tapering rod," *European Journal of Physics*, vol. 11, pp. 228–232, 1990.
[8] S. C. Crown, "Modeling the effects of velocity, spin, frictional coefficient, and impact angle on deflection angle in near-elastic collisions of phenolic resin spheres," 2004.
[9] J. H. Bayes and W. T. Scott, "Billard-ball collision experiment," *American Journal of Physics*, vol. 31, no.3, pp. 197–200, 1963.
[10] R. E. Wallace and M. C. Schroeder, "Analysis of billiard ball collisions in two dimensions," *American Journal of Physics*, vol. 56, no.9, pp. 815–819, 1988.
[11] D. Gugan, "Inelastic collision and the hertz theory of impact," *American Journal of Physics*, vol. 68, no.10, pp. 920–924, 1999.
[12] R. Cross, "Cue and ball deflection in billiards," *American Journal of Physics*, vol. 76, no.3, pp. 205–212, 2007.
[13] M. de la Torre Juarez, "The effect of impulsive forces on a system with friction: the example of the billiard game," *European Journal of Physics*, vol. 15, pp. 184–190, 1994.
[14] Carom3d. [Online]. Available: http://www.carom3d.com
[15] M. Smith, "Pickpocket: A computer billiards shark," *Artif. Intell.*, vol. 171, pp. 1069–1091, 2007.
[16] C. Archibald, A. Altman, and Y. Shoham, "Analysis of a winning computational billiards player," in *Proceedings of IJCAI*, 2009, pp. 1377–1382.
[17] M. Smith, "Running the table: An ai for computer billiards," *International Computer Games Association Journal*, vol. 56, no.9, pp. 191–193, 2004.
[18] Z. M. Lin, J.-S. Yang, and C. Y. Yang, "Grey decision-making for a billiard robot," in *Proceedings of SMC (6)*, 2004.
[19] J.-F. Landry and J.-P. Dussault, "Ai optimization of a billiard player," *Journal of Intelligent and Robotic Systems*, vol. 50, pp. 399–417, 2007.
[20] J.-P. Dussault and J.-F. Landry, "Optimization of a billiard player - tactical play," in *Computers and Games'06*, 2006.
[21] D. Brščić, M. Eggers, F. Rohrmüller, O. Kourakos, S. Sosnowski, D. Althoff, M. Lawitzky, A. Mörtl, M. Rambow, V. Koropouli, J. M. Hernández, X. Zang, W. Wang, D. Wollherr, K. Kühnlenz, C. Mayer, T. Kruse, A. Kirsch, J. Blume, A. Bannat, T. Rehrl, F. Wallhoff, T. Lorenz, P. Basili, C. Lenz, T. Röder, G. Panin, W. Maier, S. Hirche, M. Buss, M. Beetz, B. Radig, A. Schubö, S. Glasauer, A. Knoll, and E. Steinbach, "Multi joint action in cotesys - setup and challenges," CoTeSys Cluster of Excellence: Technische Universität München & Ludwig-Maximilians-Universität München, Munich, Germany, Tech. Rep. CoTeSys-TR-10-01, June 2010.
[22] D. Althoff, O. Kourakos, M. Lawitzky, A. Mörtl, M. Rambow, F. Rohrmüller, D. Brščić, D. Wollherr, S. Hirche, and M. Buss, "An architecture for real-time control in multi-robot systems," in *Human Centered Robot Systems*, ser. Cognitive Systems Monographs. Springer Berlin Heidelberg, 2009, vol. 6, pp. 43–52.
[23] T. Nierhoff, O. Kourakos, and S. Hirche, "Playing pool with a dual-armed robot," in *International IEEE Conference on Robotics and Automation*, 2010, pp. 3445 – 3446.
[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.
[25] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure." ACM Press, 1999, pp. 49–60.