COVER FEATURE

Toward a Competitive Pool-Playing Robot

Michael Greenspan, Joseph Lam, Marc Godard, Imran Zaidi, and Sam Jordan, Queen's University
Will Leckie, Nortel
Ken Anderson, Larus Technologies
Donna Dupuis, University of British Columbia

Deep Green is a vision-based, intelligent robotic system that currently shoots pool at a better-than-amateur level, with the ultimate goal of challenging a proficient human opponent at a championship level.

From behind a closed door in a university campus hallway comes the distinctive clacking sound of a pool game in progress. This isn't a student lounge, but rather a laboratory where we're developing a vision-based, intelligent robotic system to play competitive pool. Named Deep Green, the system currently shoots at a better-than-amateur level, and our goal is to advance the system to be able to challenge a proficient human opponent, ultimately at a championship level.

Pool—by which we loosely refer to all cue sports, including billiards, carom, and snooker—is somewhat misunderstood, more likely to evoke images of shifty characters in smoky bars than advanced robotics. It evolved in the royal courts of medieval Europe as an indoor version of croquet. Today, pool is enjoying a resurgence of popularity worldwide. Variations are played in almost every country, and pool was recognized as a demonstration sport in the 1998 Nagano Olympics. According to a 2005 survey,[1] more than 35 million people played pool that year in the US alone, and pool ranked as the eighth most popular participation sport, just after cycling and fishing.

The first attempt to automate pool was the Snooker Machine developed at the University of Bristol in the late 1980s,[2] which culminated with a televised game on BBC's science program *QED*.[3] Since then, researchers have developed a number of pool-playing robotic systems[4-6] as well as a training system that has a computer vision component but doesn't involve robotic actuation.[7,8]

DEEP GREEN

As Figure 1 shows, Deep Green is centered on a 3-degree-of-freedom (DOF) industrial gantry robot, which is mounted to the ceiling to avoid impeding human access to the table. A digital camera, the *global vision system* (GVS), is attached to the ceiling aiming down toward the table, accompanied by an array of directional lights. Attached to the gantry's vertical post is a 3-DOF spherical robotic wrist that, combined with the gantry's linear motion, affords the robot complete reachability over the workspace.

The *end-effector,* illustrated in Figure 2, includes two distinct cue devices, one based on a linear electromagnetic motor and the other actuated pneumatically. The electromagnetic cue can be finely controlled to strike up to a velocity of 3 meters per second, which is sufficient for normal play, whereas the pneumatic cue is used solely for power breaks and strikes at 12 m/s. A small eye-in-hand camera, the *local vision system* (LVS), is also attached to the end-effector, as is a pick-and-place vacuum tool for ball-in-hand conditions and automatic racking.

The table itself is a standard 4-foot × 8-foot coin-operated pool table, and all devices are connected to a single

Published by the IEEE Computer Society

PC. While the system has been designed to play the popular game of 8 Ball, with slight modifications it could play any other variation of pool. Figure 3 shows a number of example shots.

ROBOTICS

Rather than build our own hardware, we based Deep Green on standard commercially available, albeit customized, components. This makes the system relatively inexpensive and quick to deploy, and it allowed us to focus our effort on the computational challenges.

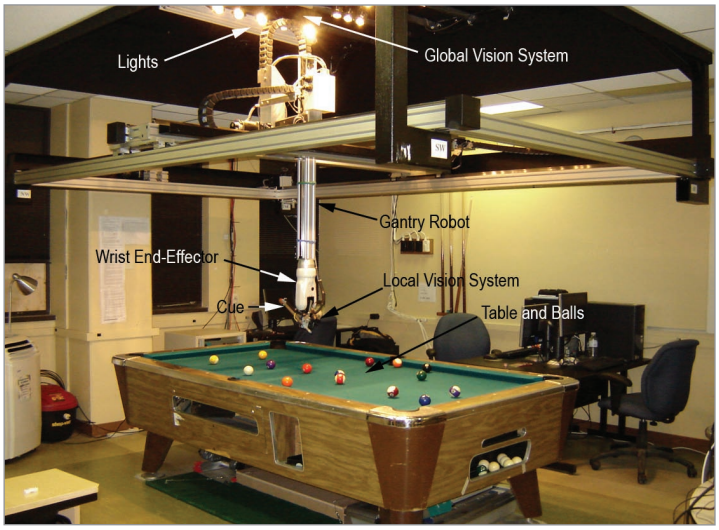

Figure 1. Deep Green robotic pool-playing system. The system is centered on a 3-degree-of-freedom gantry robot mounted to the ceiling to avoid impeding human access to the table.

Camera calibration

The system's robotic aspects rely primarily on computer vision. Before using the cameras, we had to calibrate them so that they could accurately determine the ball locations within the table's metric coordinate reference frame. Using standard techniques, we determined the cameras' intrinsic parameters, including factors to correct for the radial distortion inherent to optical systems. It was also necessary to rectify the table plane to compensate for perspective distortions that result from the GVS retinal plane not being aligned exactly parallel to the table surface, which is difficult to achieve manually to the desired accuracy.

The retinal plane and the table are related by a transformation known as a *homography,* a mapping between two planes. The standard technique for determining a homography involves extracting a minimum of four corresponding point locations between a planar pattern and its image. This technique is awkward to apply in Deep Green as the pattern must be large (the table's size) as well as very flat and accurate.

Alternatively, we exploit an invariant property of the projective space that uses a simple target comprising perpendicular lines, such as a large carpenter's square. This technique lets us integrate measurements taken at various positions on the table into a single homography, which we estimate up to an affinity. With a few additional simple measurements, we can then recover the remaining rotation and scale parameters that map the image pixels to metric locations on the table surface.

Ball localization and identification

At runtime, Deep Green acquires a GVS image when the balls come to rest and unwarps it to remove the radial and perspective distortions. It then compares this image with a set of statistics—pixel means and variances—acquired from a set of approximately 30 background images of the table, without any balls present. For each pixel, if the difference between the foreground and background pixel values exceeds some threshold value of the background standard deviation, the system judges that pixel to be foreground, that is, possibly a ball.

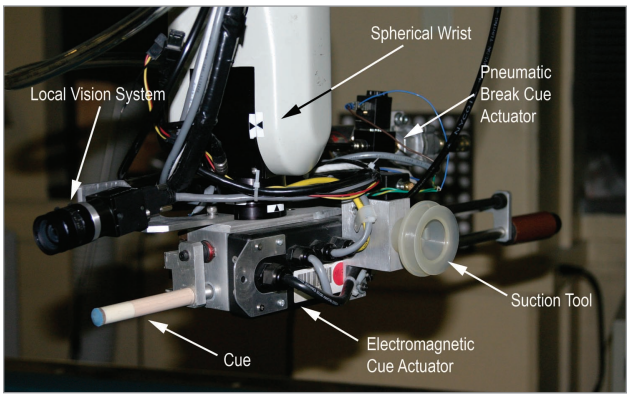

Figure 2. End-effector components.

Because this filter passes significant noise, the system applies a connected-components algorithm and only admits those regions large enough to be valid balls. It then processes these ball regions using circle-extraction and best-fit routines, leading to an accurate estimate of each ball's center location.

Once Deep Green has accurately identified the ball locations, it sends the circular subregions defining each ball to a color-indexing routine to determine the ball identities. It must know the exact identity (number) of each ball, as the formal rules for 8 Ball require nominating a ball and pocket for each shot. Offline, the system forms a 2D histogram in normalized RGB space for each of the 16 ball types from a collection of images of each ball, taken at different aspects and at various locations on the table. At runtime, it compares the color space histogram of each ball region with this database and uses a histogram similarity metric to classify the ball.

Despite strong similarities between the colors of different ball types, and reuse of colors among the stripes and

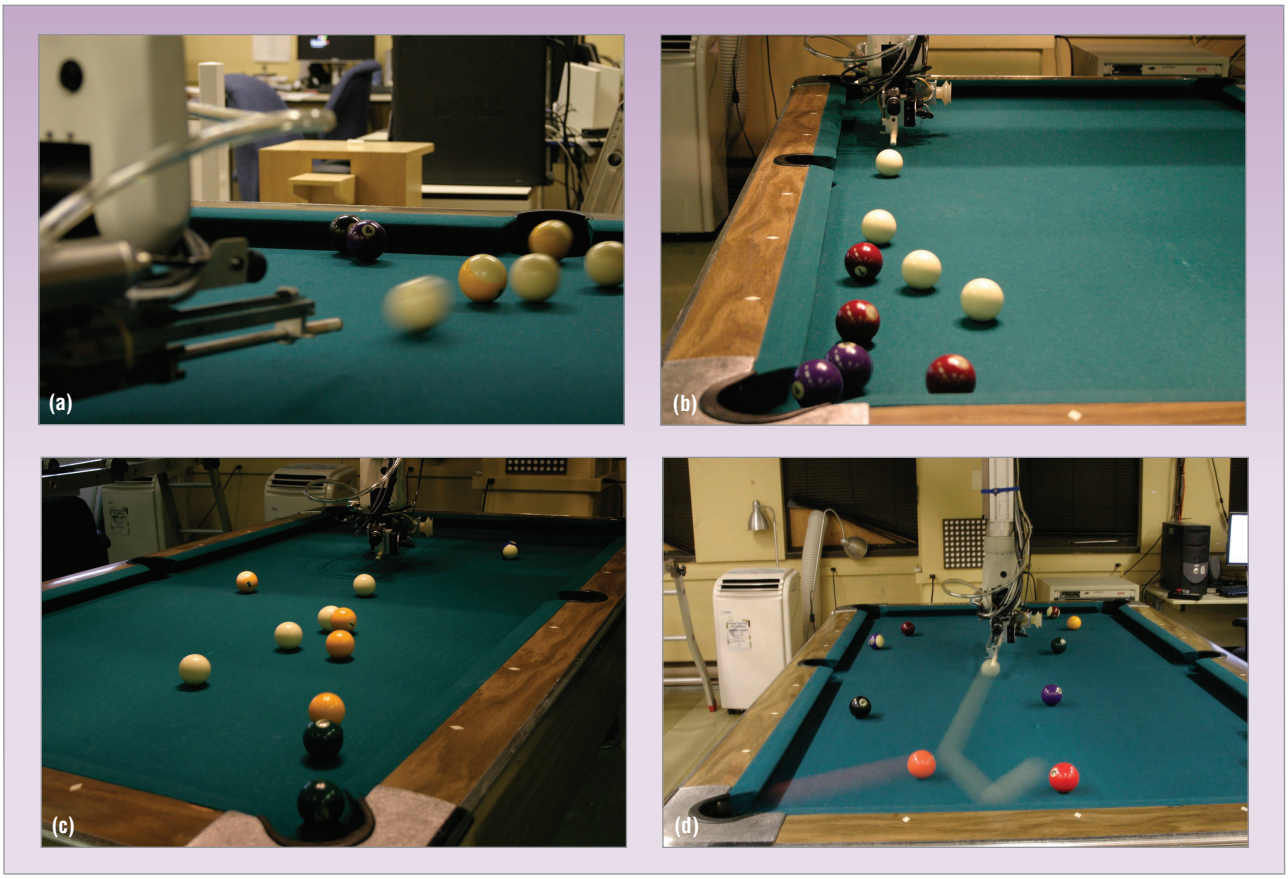

Figure 3. Example shots. (a) 9 ball in the side pocket—composite of three images. (b) Combination shot: 4 ball in the corner pocket, off of the 7 ball—composite of four images. (c) Combination shot: 6 ball in the corner pocket, off of the 1 ball—composite of four images. (d) 5 ball in the corner pocket—time-exposure image.

solids, the color-indexing method can reliably determine each ball's identity. Once the system has accurately localized and identified each ball, it can simulate the table state for shot planning.

Robot calibration

The challenge in using a standard gantry platform is its limited accuracy, as industrial robotics tend to be highly precise and repeatable but not terribly accurate. While it's possible to design a gantry robot with fine-grained accuracy, such a device would be expensive, delicate, and unlikely to maintain its accuracy while absorbing the impacts required to place shots. A more reasonable approach is to demand less accuracy from the primary positioning device and rely upon the vision system for calibration and correction.

One calibration technique involved both the LVS and GVS cameras.[9] We repeatedly positioned the robot over a series of circular patterns placed on the table surface. We then used the correspondence between the robot joint encoder values and the centers of the extracted circles within the GVS image to determine the functional relationship between the robot coordinate frame and the table plane. This technique reduced robot positioning error from the order of centimeters to within 0.6 mm on average, with a standard deviation of 0.3 mm.

Eye-in-hand visual servoing

While robot calibration rendered an improvement, a positioning accuracy of 0.6 mm is insufficient to successfully pot many long shots. It may be possible to further refine our calibration technique, successively unraveling the robot's many mysterious nonlinearities. However, the likely result of such an effort would be a very brittle system—any change in the system parameters, due to aging or other extrinsic conditions such as vibrations or temperature, would require a tedious recalibration.

To improve positioning accuracy, we have developed an eye-in-hand visual-servoing system in which the LVS camera is mounted on the end-effector with its optical axis pointing roughly along the direction of the cue. The LVS uses the known ball locations determined by the GVS as visual landmarks to detect and compensate for positioning errors accumulated during the gantry's coarse motion.

LVS correction. Consider the nearly perfect straight shot illustrated in Figure 4. In this GVS image, the inscribed line is defined by the extracted center locations of the

cue and object balls prior to placing the shot. The rendered circles are a sequence of three extracted positions of the object ball, at times t_0 to t_2, once the shot has been placed. The centers of these circles fall on or close to the line, indicating that the robot was positioned to make a very accurate straight shot. The final resting positions of the cue and object balls at time t_f also fall on this line, further supporting the shot's quality.

From the LVS's vantage, this is the *ideal line*. When the robot is servoed to its shot position, as determined by the GVS, it accumulates error. By analyzing the LVS image, and comparing the line connecting the current cue and object ball centers with the ideal line, the system can calculate transformations that correct for the robot positioning error.[10]

Figure 5a shows an LVS image acquired after the robot has been servoed to its shot position, using only the information from the GVS. The current (red) and ideal (green) lines aren't aligned, indicating positioning error. After the system executes the automatic alignment procedure, the current line overlaps almost exactly with the ideal line, as shown in Figure 5b, and the shot will therefore be very close to a perfect straight shot.

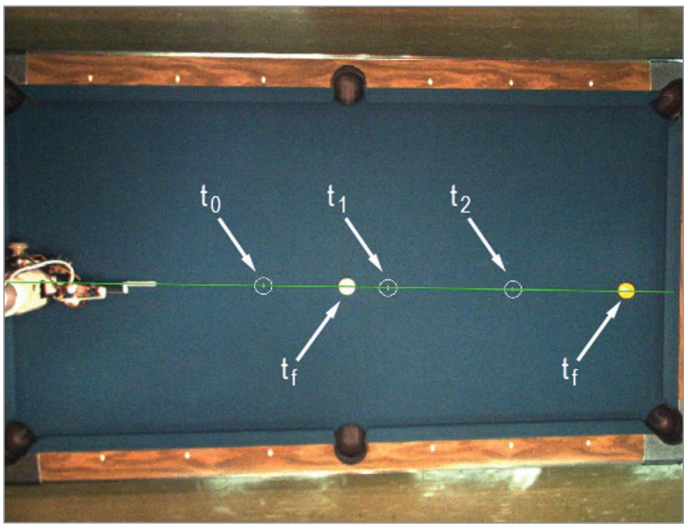

Figure 4. Straight shot. Intermediate object ball locations fall on a line defined by initial cue and object ball locations.

Alignment methods. We have developed two different methods to align the robot position with the LVS ideal line.[10] The simpler one is iterative and based entirely on 2D LVS image data. The other method uses knowledge of the 3D rigid transformation between the robot wrist coordinate reference frame and the LVS optical frame. This transformation, known as the *tool control frame* (TCF) matrix, is determined offline in a calibration stage.

Figure 6 plots the result of an experiment designed to characterize the performance of these two methods. A total of 90 straight shots were executed. Thirty of these shots used only information from the GVS and robot calibration, 30 more applied alignment using the image-based method, and the final 30 used the position-based method. We calculated the angular error of each shot by extracting the object-ball center locations at a number of (at least two) positions along their trajectories using the GVS and comparing the angle of this line with the line defined by the cue and object balls prior to placing the shot (similar to Figure 4).

We plotted the angular errors for each of the 3 sets of 30 shots in ascending order. Alignment using either method significantly reduced the angular error. Without alignment, the mean absolute error was 1.8 degrees. With alignment, the error was reduced by more than two thirds, to 0.51 degrees and 0.56 degrees for the image- and position-based methods, respectively. While the accuracy is similar for both alignment methods, the position-based method is approximately 40 percent faster. Once the straight shot is aligned accurately, the TCF matrix can be used to further rotate and translate the cue around the cue-ball center to execute a cut shot of any desired angle and spin.

GAMING

For those who play pool only casually, skill is the limiting factor, and sinking the current ball is usually the sole concern. For more advanced players, however, strategy

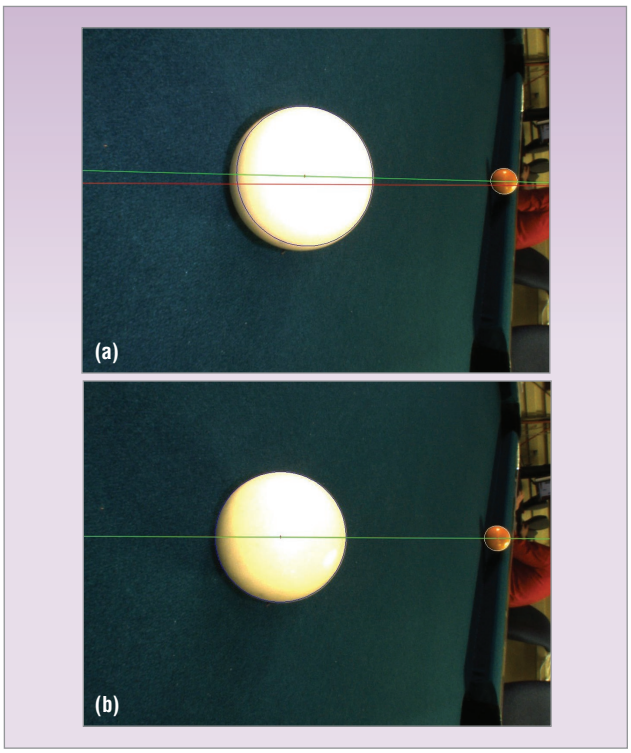

Figure 5. LVS correction. (a) Current (red) and ideal (green) lines before alignment. (b) After alignment, current and ideal lines overlap.

becomes a key element of the game, and professionals are known to plan five or more shots ahead for a given table state. For a robotic system to play competitively, it must therefore strategize computationally, which involves both predicting and planning future table states. This requires the interplay of physics simulation and search.

Physics simulation

To predict the table state after a shot so that subsequent shots can be planned, an accurate physics model is necessary. Spin is an essential element of the game, and imparting spin on the cue ball by displacing and angling the cue at impact is a technique used to control the interaction and placement of balls following a shot.[11] The physics model therefore involves conserving not only linear but also angular momentum.

We have developed a physics simulator that predicts a shot's outcome from a derived physics model.[12] Unlike physics simulators that use the more common numerical integration approach, our method operates in the continuous domain, predicting the times of pending events such as collisions or transitions between motion states. Our technique returns an exact analytic solution based on a parameterization of the separation of two moving balls as a function of time. The resulting equation is a quartic polynomial that can be solved either iteratively or in closed form to determine the collision time. A similar derivation exists for other events, such as ball-rail and -pocket collisions and transitions from sliding-to-rolling and rolling-to-stationary states.

Compared to integration, our approach is more

- accurate, requiring no discrete time step; and
- time efficient, requiring approximately two to three orders of magnitude fewer computations per shot.

This added efficiency is especially important when the physics simulator is used in expanding a game tree, as many different shots—sometimes tens of thousands or more—might need to be simulated prior to making a decision.

Our physics simulator was the basis for the Computational 8 Ball Tournaments at the 10th and 11th International Computer Olympiads.[13] These tournaments let teams develop different strategy engines and compete using the common physics simulator.

One consideration in modeling the physics was shot noise. When a human or robotic player takes a shot, error in the cue's position and velocity makes each shot nonideal. To make the simulation more realistic and the competition more challenging, we added zero-mean random Gaussian noise to each of the five shot parameters that determine the outcome of a shot: two angles (θ,ϕ), two offsets (a, b), and the striking speed V.[14] The sigma values of each distribution were empirically determined to cause one missed shot every 10 shots on average, a success rate similar to that of advanced human play. When planning a shot for robotic play, a noise model based on the robot's calibrated positioning accuracy can be used to determine the probability of a given shot's success.

Search

With the physics simulator's ability to predict a shot's outcome, it's then necessary to evaluate many possible shot sequences to determine the best shot to place given the current table state. Our approach to this search is based on the minimax game tree used in games like chess and checkers.[15,16] While the basic concept is the same as in chess, one difference is that pool is played in a continuous, rather than a discrete, domain. The size of the search space for any particular shot is therefore truly infinite, rather than the huge but finite search space of chess.

Another unique consideration in pool is shot noise. In practice, each of the five shot parameters has an element of uncertainty that can be modeled as a probability distribution. For this reason, we have adapted the expectimax search tree, which has been applied to games like backgammon that have a probabilistic component. Because pool is played in a continuous domain, the chosen tree search algorithm incorpo-

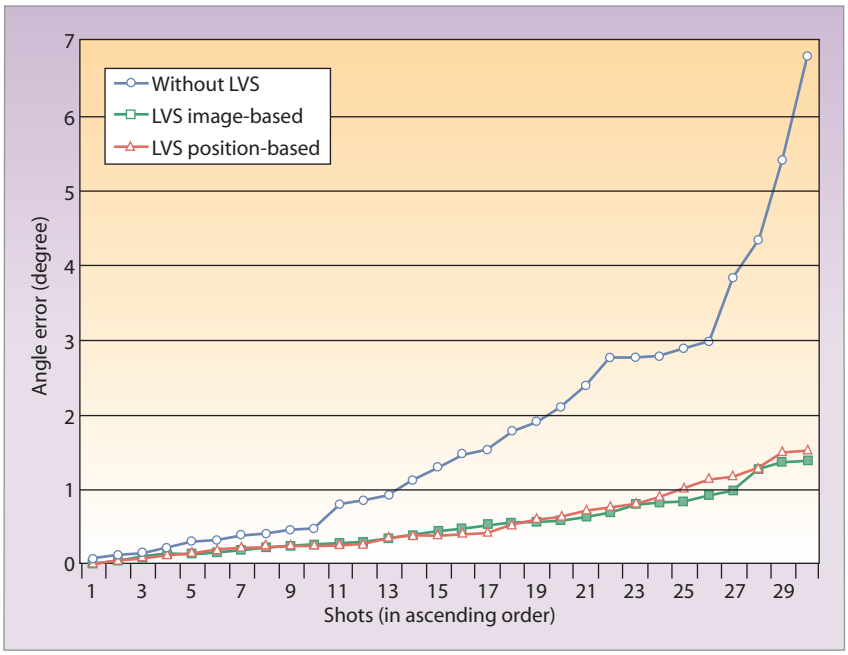

Figure 6. Angular error in straight shot tests with LVS correction. Alignment reduced the error by more than two thirds, to 0.51 degrees and 0.56 degrees for the image- and position-based methods, respectively.

Table 1. Summary across search depths for zero-, low-, and high-noise tournaments.

Noise	Player	Average wins (percent)	Average points scored	Average point differential	Average misses (percent)	Average ball-in-hand (percent)
Zero	Greedy	9.9	771.6	–1,093.3	0.0	10.3
	All depth 1	61.1	1,390.8	278.4	0.0	2.5
	All depth 2	79.9	1,622.5	814.9	0.0	2.5
Low	Greedy	19.9	963.1	–791.0	6.3	12.0
	All depth 1	62.7	1,458.8	323.9	2.6	3.6
	All depth 2	67.4	1,523.9	467.1	1.6	3.4
High	Greedy	36.5	1,301.7	–314.6	11.8	14.3
	All depth 1	54.8	1,484.4	114.2	8.9	10.4
	All depth 2	58.7	1,519.9	200.4	9.3	9.0

rates statistical sampling to account for uncertainty in shot execution. The utility of a future table state is weighted by its probability of occurrence, and the weighted utilities of the children of each node are combined when considering which path to traverse.

Empirical evaluation of strategic play

To explore the benefits of strategic play in pool, we executed a set of experiments using this tree-search framework.

Methodology. We simulated a series of 8 Ball tournaments involving 19 competitors, all with identical shot-generation algorithms. Eighteen of the competitors used different tree-search depths, tree-scoring variations, and evaluation-function variations; the 19th used a depth-zero "greedy" shot-selection algorithm based solely on the probability of the current shot's success with no regard for the resulting table state or future shots. This greedy player had the same skill level as the other competitors but thought like an amateur.

Three tournaments were played with three different noise models reflecting the players' technical skill level. For the high-noise model, about 80 percent of balls were sunk as planned; for low noise, about 90 percent; and for zero noise, all shots were executed exactly as planned. All players in each tournament used the same noise model and search algorithm. Each tournament therefore isolated performance as a function of tree-search depth and evaluation-function variation. A search depth of 1, for example, considers not only the current shot but also all shots resulting from the current shot. The various scoring and evaluation functions differed in how they rated a leaf node's utility as well as in how they combined the information from child nodes in propagating back up the tree.

This is similar to comparing two human players by categorizing their play in two areas:

- technical skill—precision in executing shots; and
- level of strategic play—how far ahead in the game the player looks, and how the player controls the cue ball position for the next shot.

We examined numerous combinations of tree-scoring variations—Monte Carlo, probabilistic, or success-weighted—and evaluation-function variations: average, maximum, or weighted. Within each tournament, the players with common search algorithm/evaluation functions (but varying search depth) played 200-game matches against one another and against the greedy player in a round-robin format. The winning player of each game received a total of 10 points, and the losing player received one point for each pocketed ball of its color group (stripes or solids), for a maximum of seven points. The match score was the sum of the game scores.

Results. Table 1 summarizes the results from these experiments. Players are ranked by their overall performance by averaging the percentage of games won, points scored, point differential, miss rate, and percentage of shots resulting in a ball-in-hand. The percentage of shots resulting in a BIH indicates not only how often a player fouled, but more importantly how often it left itself with no shot. The greedy player was more heavily penalized by this setting because it never considered the table state resulting from its chosen shot.

In the zero-noise tournament, the deeper-searching players consistently outplayed their shallower-searching competitors. For a given search type/evaluation function variant, the depth 2 player always defeated the greedy player easily and then defeated the depth 1 player in turn. The greedy player was defeated in all matches in the zero-noise tournament, winning at best 16.5 percent of the games in its match against one player. Against the greedy player, all of the depth 2 players scored more wins with a higher point differential than the corresponding depth 1 player.

Look-ahead. Positional play in the form of look-ahead is clearly an important consideration in pool. Choosing the easiest shot, or the shot with the highest probability

of success, doesn't result in a competitive player; planning strategically using look-ahead does. These results mirror the expectation for human players similarly characterized by technical skill and level of strategic reasoning. Players are always limited by their technical skill, regardless of how strategically they plan shots. However, for sufficiently skilled players, the benefits of strategic reasoning and cue-ball placement in the form of look-ahead always dominate over less strategic play.

While these experiments have evaluated look-ahead only to a depth of 2, the benefits of look-ahead should continue to be apparent for search depths up to 8, at which point all game tree branches will have terminated, with all balls sunk and the game completed. In practice, expanding the game tree to greater depths can be quite time expensive, and so tournament competitors have restricted their searches to depths of 2 or 3.

ADVANTAGES OF MACHINE PLAY

In many ways, pool is an ideal game for automation. A great deal of human pool instruction and practice is oriented toward establishing an accurate and repeatable stroke. Machines routinely outperform humans at positioning accuracy and repeatability, and they function consistently, without the performance-degrading effects of muscle fatigue. They also aren't susceptible to psychological pressure, a significant source of variation and failure in human play.

In addition, a machine like Deep Green can sense the balls' absolute metric locations in the table coordinate reference frame. Humans can ascertain the balls' geometric arrangement based on their relative positions on the table, allowing them to plan and execute challenging shots, but in certain situations even skilled humans have difficulty perceiving the correct angles. For example, shots that involve multiple banks are inherently difficult to perceive, and humans often use inexact systems based on table landmarks (diamonds) to augment their perception. In contrast, the machine resolves the metric location of all balls and table elements such as rails and pockets. This allows for more exact geometric planning, and enhances the machine's ability to predict a shot's outcome.

Another advantage of the machine is its computational simulation of the table's physics. Most human players rely on an intuitive understanding of this aspect of the game. Typically with little or no formal knowledge of physics, they develop heuristics to predict the subsequent table state that results from the multiple interactions of any particular shot. While often useful, these heuristics have limited fidelity. In contrast, the machine has an executable physics model and, so long as a handful of parameters have been estimated through calibration, can use a physics simulator to predict the resulting table state both accurately and efficiently.

Moreover, the cue end-effector provides precise control of stroke speed. The electromagnetic linear actuator responsible for the forward motion of Deep Green's stroke has a dedicated digital control unit that can be commanded in either position or velocity modes. The cue's speed can range from almost stationary to approximately 3 m/s, with an average error of approximately 0.1 percent. In contrast, humans tend to strike with one of six speeds: slow, medium-slow, medium, medium-fast, fast, or break. The added graduation in controlling cue speed translates to an increased ability to place the cue ball and predict and control the table state.

Once the mechanics of placing a shot have been mastered, pool becomes a strategic game, and here too the machine has a potential advantage. The essence of pool strategy is the ability to look ahead and predict the table's state following a potential shot or series of potential shots. This same capability lets computers outperform people at chess and other games recently believed to be only within the realm of human mastery.

NEED FOR INTELLIGENCE

The Deep Green project has inspired polar opposite responses on the degree of difficulty required to attain our goal. Some people who are familiar with technology but not with pool have regarded it as a straightforward task, requiring only standard robotic techniques to provide a solution. In contrast, proficient players who have no special relationship with technology tend to argue that pool is a distinctly human activity, requiring human intelligence and skill, and that automating it is impossible.

Our view lies somewhere between these two extremes. We believe that developing a robotic system to play pool competitively against a proficient human opponent is achievable. The technical problems are both interesting and sufficiently challenging to motivate advanced research, but not so difficult as to evade a meaningful solution.

Another question that Deep Green raises is whether computational intelligence is necessary for robotic pool. Isn't an accurate positioning system and simple shot planning based purely on geometry sufficient? There are two answers to this question. First, accurate positioning of a standard gantry robot is itself a challenging goal requiring sensor-based methods for calibration and correction. Second, even if perfectly accurate positioning were possible, it's still advantageous to play strategically and plan ahead a number of shots, as evidenced by our experiments with zero-noise tournaments.

Deep Green currently plays at a better-than-amateur level, planning and executing difficult combination and rail shots from across the table. It has pocketed runs of four consecutive balls, and it's only a matter of time before it can consistently run the table.

Several research challenges must be addressed to advance the system further. The most difficult will emerge in competing against proficient human opponents. Humans are

crafty competitors, able to efficiently recognize and exploit weaknesses in their opponents. To play at a competitive level, Deep Green must incorporate insights from machine-learning and opponent-modeling techniques.

Acknowledgments

The authors thank Precarn Inc., the Institute for Robotics and Intelligent Systems, the Canada Foundation for Innovation, and the Natural Sciences and Engineering Research Council of Canada for their financial support, and Elisha Hardwick for her photographic work. They also thank the many students who have contributed long hours to the development of the Deep Green system.

References

1. Sporting Goods Manufacturing Assoc., *State of the Industry Report,* 2005.
2. S.W.S. Chang, "Automating Skills Using a Robot Snooker Player," doctoral dissertation, Univ. of Bristol, Bristol, UK, 1994.
3. R. Popper, "Bring Back the Snooker-Playing Robot!," *The Guardian*, 7 Aug. 2007, p. 3.
4. M.E. Alian et al., "Roboshark: A Gantry Pool Player Robot," *Proc. 35th Int'l Symp. Robotics* (ISR 04), 2004; www.cs.sfu.ca/~psabzmey/personal/publ/papers/alian_roboshark_isr04.pdf.
5. S.C. Chua et al., "Performance Evaluation of Fuzzy-Based Decision System for Pool," *Applied Soft Computing,* Jan. 2007, pp. 411-424.
6. Z.M. Lin, J.S. Yang, and C.Y. Yang, "Grey Decision-Making for a Billiard Robot," *Proc. 2004 IEEE Int'l Conf. Systems, Man and Cybernetics,* vol. 6, IEEE Press, 2004, pp. 5350-5355.
7. L.B. Larsen, M.D. Jensen, and W.K. Vodzi, "Multimodal User Interaction in an Automatic Pool Trainer," *Proc. 4th IEEE Int'l Conf. Multimodal Interfaces* (ICMI 2002), IEEE CS Press, 2002, pp. 361-366.
8. H. Denman, N. Rea, and A. Kokaram, "Content-Based Analysis for Video from Snooker Broadcasts," *Computer Vision and Image Understanding,* Nov./Dec. 2003, pp. 176-195.
9. F. Long et al., "Robotic Pool: An Experiment in Automatic Potting," *Proc. 2004 IEEE/RSJ Int'l Conf. Intelligent Robots and Systems* (IROS 2004), vol. 3, IEEE Press, 2004, pp. 361-366.
10. J. Lam, "Eye-in-Hand Visual Servoing to Improve Accuracy in Pool Robotics," master's thesis, Dept. Electrical and Computer Eng., Queen's Univ., Kingston, Canada, 2007.
11. D.G. Alciatore, *The Illustrated Principles of Pool and Billiards,* Sterling, 2004.
12. W. Leckie and M. Greenspan, "Pool Physics Simulation by Event Prediction 2: Collisions," *Int'l Computer Games Assoc. J.,* Mar. 2006, pp. 24-31.
13. M. Greenspan, "Pickpocket Wins Pool Tournament," *Int'l Computer Games Assoc. J.,* Sept. 2006, pp. 153-156.
14. W. Leckie and M. Greenspan, "Pool Physics Simulation by Event Prediction 1: Motion Transitions," *Int'l Computer Games Assoc. J.,* Dec. 2005, pp. 214-222.
15. W. Leckie and M. Greenspan, "Monte-Carlo Methods in Pool Strategy Game Trees," *Proc. 5th Int'l Conf. Computers and Games* (CG 06), LNCS 4630, Springer, 2007.
16. M. Smith, "PickPocket: A Computer Billiards Shark," *Artificial Intelligence*, Nov. 2007, pp. 1069-1091.

Michael Greenspan *is an associate professor in the Department of Electrical and Computer Engineering and in the School of Computing, Queen's University, Kingston, Ontario, Canada, and is spending the year as a visiting scientist at the University of Coimbra, Portugal. His research focuses on problems in computer vision and computer gaming. Greenspan received a PhD in systems and computer engineering from Carleton University, Ottawa. Contact him at michael.greenspan@queensu.ca.*

Joseph Lam *is a PhD candidate in the Department of Electrical and Computer Engineering at Queen's University. His research interests include computer vision and visual servoing. Contact him at jctlam@gmail.com.*

Marc Godard *is a bachelor's student in the School of Computing, Queen's University. His research interests include computer vision, cognitive models, and prediction algorithms. Contact him at 4mg12@qlink.queensu.ca.*

Imran Zaidi *is a bachelor's student in the School of Computing, Queen's University. Contact him at 3aiz@qlink.queensu.ca.*

Sam Jordan *is a master's student in the Department of Electrical and Computer Engineering at Queen's University. His research interests include robotics, augmented reality, and human-computer interaction. Contact him at 3sj1@queensu.ca.*

Will Leckie *is an electro-optics hardware designer with Nortel at its Carling Campus in Ottawa. His research interests include robotics and artificial intelligence. Leckie received a master's degree in electrical and computer engineering from Queen's University. Contact him at will.leckie@ece.queensu.ca.*

Ken Anderson *is a software engineer at Larus Technologies in Ottawa. His research interests include robotics, single-agent search, and computer games. Anderson received a master's degree in computing science from the University of Alberta. Contact him at anderson@cs.ualberta.ca.*

Donna Dupuis *is a master's student in the Department of Electrical Engineering at the University of British Columbia, Vancouver, British Columbia, Canada. Her research interests include computer vision, robotics, and artificial intelligence. Contact her at dupuisd@mech.ubc.ca.*