**Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems
September 28 - October 2, 2004, Sendai, Japan**

Robotic Pool: An Experiment in Automatic Potting

Fei Long[1], Johan Herland[1], Marie-Christine Tessier[1],
Darryl Naulls[1], Andrew Roth[3], Gerhard Roth[4], Michael Greenspan[1,2,5]

[1]Dept. of Electrical and Computer Engineering, [2]School of Computing, Queen's University, Canada
[3]Dept. of Computer Science, University of Waterloo, Canada
[4]Institute for Information Technology, National Research Council of Canada
[5]corresponding author: michael.greenspan@ece.queensu.ca

Abstract—**A robotic system is presented which automatically pots (i.e., sinks) pool balls. A homography is estimated that relates the gantry robot coordinate frame to the overhead (global) camera coordinate frame. This homography is computed by first calculating the mapping between the camera frame and a projection of the robot frame, and then solving the pool table plane equation in the robot frame. A measurement technique has been developed which is based upon a local camera attached to the robot end-effector. This local camera allows the robot to be positioned accurately over circular targets placed on the table. The homography and table plane equation are then estimated by establishing correspondences between at least 4 measured target positions in the global camera and robot frames. The resulting homography allows the gantry to be positioned to within an average of 0.6 mm of a global camera frame position over the extent of a full sized pool table. The system has been used to pot a ball with 67% accuracy over the extent of the table, with a high repeatability.**

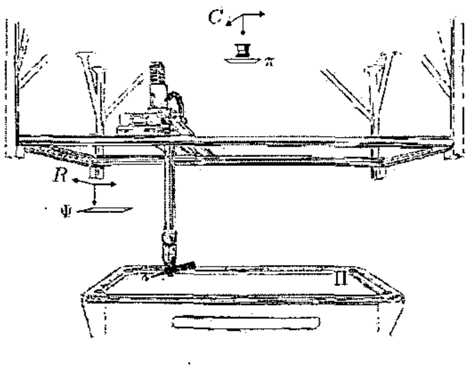


Fig. 1. System Components with Labelled Coordinate Frames

I. Introduction

Among the various forms of entertainment robotics robotic pool is emerging to be an intriguing and challenging problem. The game of pool demands high levels of perception, strategy, and precision which make it ideally suited for the computational challenges of a robotic system, and pool is emerging as the "computer chess" of robotics. In the early work of Shu et al. [1], a robotic gantry system was constructed to play snooker. Other work has tended to focus on high level strategy, independent of platform [2]. Recently, an automated pool training system was developed [3] which includes a machine vision component, but no robotic actuator.

We describe the development of a system based upon a gantry robot, shown in Fig.1. The main elements are: a 5 dof gantry robot, a 1 dof cue end-effector, a ceiling mounted (global) camera, and a standard $4' \times 8'$ pool table. There is also an end-effector mounted (local) camera, that is used here for calibration purposes, and that will be used in the future to compensate for positioning errors. One difference between our system and previous work is our use of an end-effector mounted camera to improve positioning accuracy.

We are taking a bottom-up approach to the development, and our efforts have focused on issues of sensing and actuation. Our belief is that higher-level planning and strategy will be motivated and informed by solutions to the lower-level sensing and actuation systems. A fundamental issue is accuracy. It is possible to design a gantry robot that has fine-grain accuracy ($\sim 15\mu$m) over the desired workplace. For example, Coordinate Measurement Machines (CMMs) have such accuracy over similar working volumes. This high degree of accuracy comes at a cost, however, and such a device would be expensive, delicate, and unlikely to maintain accuracy while absorbing the impacts required when placing shots. A more reasonable approach is to demand less absolute accuracy from the primary positioning device, and to rely upon a vision system for fine-tuning.

In this work, we describe a method whereby a single overhead camera is used to improve the accuracy of the gantry on a plane. The method is validated experimentally, and is used as the basis to automatically pot (i.e., sink) balls. Sec.II describes the problem as a mapping from 2D (camera) to 3D (robot) coordinates, and Sec.III presents a method to solve this mapping. This solution requires identifying a set of circular targets in both the camera and robot frames, and Sec. IV, presents a process for locating these targets. An evaluation of the system accuracy is presented in Sec.V, and an experiment in automatic potting (i.e. ball sinking) is described in Sec. VI. The paper concludes in

Sec. VII with a summary and recommendations for future work.

II. System and Problem Definition

Let C be the global camera coordinate reference frame, and let R be the robot frame. A point ${}^{c}\vec{p}={}^{C}(x,y,z)$ in C is measured in pixel units (u,v) by its projection onto the camera retinal plane π. Coordinates in R are measured by the robot joint encoders, albeit with limited accuracy. The table forms a 3D plane Π which can be described in either frame.

The objective at this stage is to automatically pot a single ball. To achieve this goal we must perceive the position of the ball on Π within C, and calculate the correct location within R to reposition the cue and place a shot. We must therefore map coordinates which lie on Π from C to R;

$$h:(u,v)\longmapsto {}^{R}(x,y,z)\ \forall\ {}^{c}\vec{p}\in\Pi \tag{1}$$

The function h is a 2D to 3D homography, which is a mapping that preserves collinearities and therefore maps between planes in the two frames. Homographies have been well-explored within the Computer Vision research community [4], and the 2D to 2D homography is particularly useful in establishing correspondences between two camera frames.

III. Solution Approach

A naive approach would assume that π, Π, and the $x-y$ plane of R are all parallel, and solve Eq.1 based purely on the geometry and scaling of their projections. It would in practice be difficult to ensure that these planes are parallel, and small deviations could result in large inaccuracies. It is also desirable for h to compensate for nonlinearities inherent in both the camera and the robot measurement systems.

One possibility is to estimate h from a set of correspondences between the two frames. An alternative approach, which we take here, is to first estimate a 2D homography that relates Π to C and R, and then apply the solution of the plane equation of Π in R.

In general, at least 4 non-collinear point correspondences are required to compute the homography between two 2D homogeneous spaces. In our case only the (u,v) camera frame measurements are in homogeneous coordinates, whereas the robot ${}^{R}\vec{p}={}^{R}(x,y,z)$ coordinates are measured in Cartesian space. The ${}^{R}\vec{p}$ must therefore first be mapped into homogeneous coordinates:

$$(\psi,\omega,1)={}^{R}(x/z,y/z,1) \tag{2}$$

This is equivalent to projecting these coordinates onto the plane at $z=1$ in R, which we shall denote as Ψ.

We compute the homography between π and Ψ by placing a set of $N\geq4$ targets on Π. The targets are white circles printed on a black background, and the center of the i^{th} target is determined as (u_i,v_i) in π and ${}^{R}(x,y,z)_i$ in R. The homography H is a 3x3 matrix estimated from these correspondences. For any subsequent position (u_j,v_j) of target j on Π, the least square error estimate of the corresponding projection onto Ψ is then given by:

$$[\psi_j,\omega_j,1]^{\top}=H[u_j,v_j,1]^{\top} \tag{3}$$

Eq.3 describes the mapping between π and Ψ up to a scale factor. To satisfy Eq.1, however, we require an additional mapping from Ψ to Π, which we accomplish by making further use of the measured targets in R. The targets were positioned on Π, so the equation of Π in R can be determined directly from the N values of ${}^{R}\vec{p}_j$. As $N>3$, the plane equation is calculated as the least squares estimate of these measured values. The ψ_j and ω_j determined from Eq.3 can then be substituted directly into the plane equation to solve directly for ${}^{R}(x_j,y_j,z_j)$.

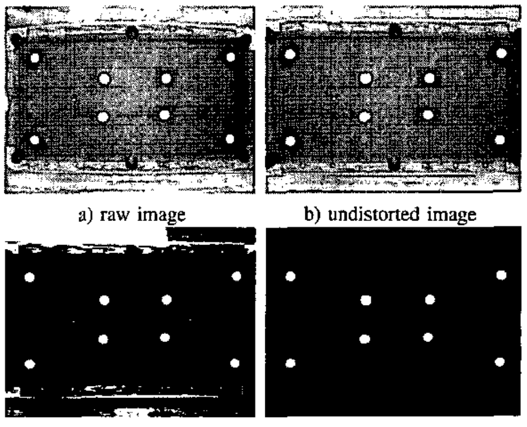


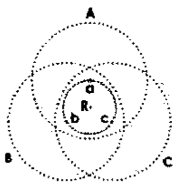
Fig. 2. Locating Targets in C

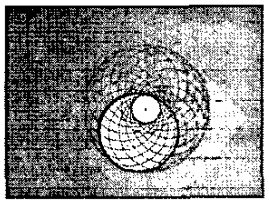
IV. Target Extraction

Estimating H and Π requires establishing correspondences between $N>3$ target locations. The center of each circular target j is measured as a (u_j,v_j) pixel location in π, and as an ${}^{R}(x,y,z)_j$ robot location in R.

A. Target Extraction in C

Prior to target extraction, the radial distortion of the the camera is estimated using a standard technique [5]. All images are corrected for radial distortion prior to subsequent processing as illustrated in Fig.2b. A simple threshold is then applied to produce a binary image from the gray scale image. It can be seen in Fig.2c that the white circles of the 8 targets pass easily through the binary filter, but that some noise results. A connected components algorithm is applied next to remove the noise. Any components which are significantly larger or smaller than the expected target size are considered as noise and eliminated. After this step, the remaining components belong to a true target circle, as illustrated in Fig. 2d. For each circle j, the edge points are identified and a least square error estimate of the center (u_j,v_j) is determined [6] and used as a target location in C.





a) minimal 3 rotations b) actual 21 rotations

Fig. 3. Center of Rotation

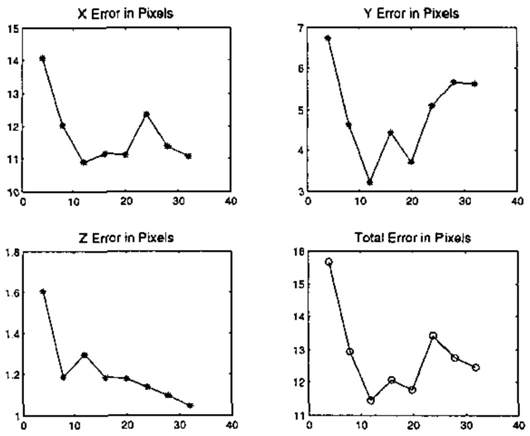


Fig. 4. Error vs. # of Targets

B. Target Extraction in R

Whereas determining target locations in C is straightforward, determining the corresponding locations in R is more involved. To do so, it is necessary to position the robot accurately over a target j at a fixed height offset. The values of the robot's first 3 joint encoders are then taken to be the target's 3D position in the robot frame, i.e., ${}^{R}(x,y,z)_j$.

The challenge is to position the robot over the target as accurately as possible. One possibility is to mount a fixed pointing device, such as a pin, on the robot pointing down toward the center of the target. The robot could then be manually "jogged" to position the pin at the center point and just touching the target. While this method is simple, it is imprecise, subject to human error, and prohibitively labourious when many target locations are desired.

As an alternative, we employ a small camera mounted on the end-effector aimed down along the ${}^{R}z$-axis. The center and radius of a target as imaged from this camera is used to accurately position the robot over the target's center at a fixed ${}^{R}z$ offset.

1) Determining the Center of Rotation: We desire to position the robot ${}^{R}z$-axis directly over the target center. It is therefore necessary to identify the pixel within the local camera image plane that intersects with this axis.

This is accomplished by positioning the gantry roughly over a target and acquiring an image with the local camera. The robot is positioned in xyz using the first 3 proximal joints, with the distal revolute joints 4 and 5 in their home positions. The target center ${}^{z}(u,v)_1$ is extracted from the acquired image.

Joint 4 is next rotated to at least 2 additional locations, and the target centers are extracted for each rotation. If the gantry were initially positioned such that ${}^{R}z$ lay directly over the target center, then all K extracted centers would be equal, i.e., ${}^{z}(u,v)_1 = {}^{z}(u,v)_2 = ... = {}^{z}(u,v)_K$. Otherwise, the K center points circumscribe a circle, the center of which is the intersection of ${}^{R}z$ with the image plane of the local camera.

This process is illustrated in Fig.3a using 3 rotations, which is the minimum required. Here, the 3 extracted circles A, B, and C have respective centers a, b, and c. The circle circumscribed by a, b, and c is centered at R, which is the center of rotation of robot joint 4, i.e., the intersection of ${}^{R}z$ with π. In practice, it is both convenient and more accurate to use a larger number of rotations, and to estimate the circle center in the least squared sense [6]. Using 21 rotations as illustrated in Fig.3b, the center pixel was identified with an accuracy of ± 0.1 pixels, or 0.04 mm.

2) x-y Positioning: Once the center of rotation has been determined, the robot can be accurately positioned in x-y over a target center. First the robot is positioned so that the target is entirely imaged within the local camera, and the target center and radius are extracted. The robot is then repositioned in x-y by the difference between the target center and the previously determined rotation center. A new image can be acquired in the new position and the above steps repeated until the desired accuracy is achieved. In practice, we have found that it takes 3 to 5 iterations to achieve a positional accuracy of ± 0.3 pixels (0.1 mm).

3) z Positioning: Once the robot is positioned accurately in x-y, the z-offset of the robot can be determined from the target radius. First, the radius of the target at the desired z-offset is predetermined. The current observed radius $\hat{\rho}$ is compared with the desired ρ, and the z value is adjusted accordingly. If $\hat{\rho} < \rho$, then the robot is lowered toward the table: if $\hat{\rho} > \rho$, then it is raised. This is repeated until the desired accuracy is achieved. In practice, we have found that it takes 5 to 10 iterations to achieve an accuracy of ± 0.3 pixels (0.1 mm).

Repeating the above x-y and z positioning processes, the robot can be positioned accurately over each target, and the values of joints 1 to 3 for each target j yield ${}^{R}(x,y,z)_j$. These values are combined with the corresponding target centers (u_j, v_j) extracted from the global camera and are then used to estimate H and Π in Eq.1.

V. Experiment 1: Accuracy Measurement

The objective of this experiment was to characterize the positioning error. There are a number of random and systemic sources of error in the system, including: limited accuracy of the gantry controller; limited resolution of the global camera; inaccuracies in the estimation of the lense radial distortion; and numerical errors in the homography computation.

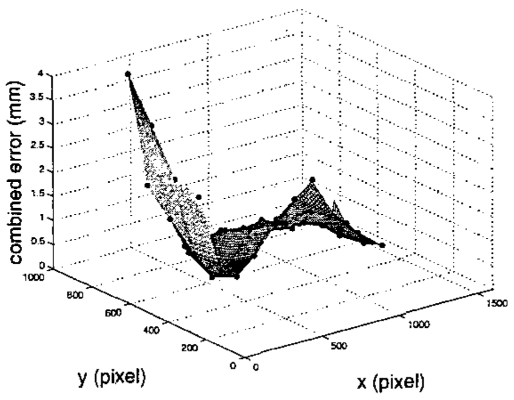


Fig. 5. Positioning Error

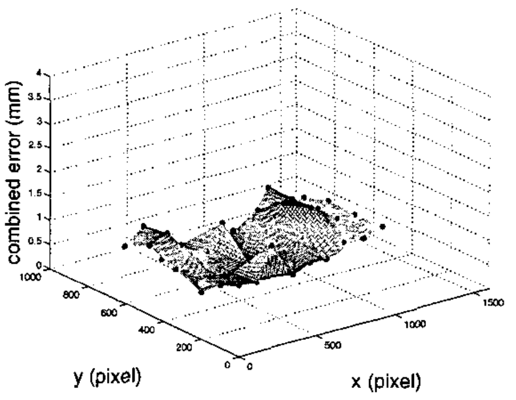


Fig. 6. Positioning Error With Correction

	$\|min\|$	$\|max\|$	*mean*	*st. dev*
x	0.042	4.209	-0.079	1.196
y	0.010	1.436	-0.074	0.358
z	0.001	0.626	0.084	0.219
total	0.171	4.278	0.984	0.811

TABLE I
POSITIONING ERROR (MM)

	$\|min\|$	$\|max\|$	*mean*	*st. dev*
x	0.002	1.058	-0.518	0.304
y	0.008	0.310	0.137	0.081
z	0.002	0.727	-0.140	0.228
total	0.185	1.114	0.618	0.277

TABLE II
POSITIONING ERROR WITH CORRECTION (MM)

The positioning error can be measured directly by a process similar to that described in Sec.IV. The target is placed on Π at a previously unmeasured location and its center (u, v) is extracted from the global camera image. Eq.1 is then used to compute the robot frame coordinate $^R(x, y, z)$ and the robot is repositioned to this coordinate. Once respositioned, the end-effector is centered over the target at a fixed height, and an image of the target is acquired from the local camera. The center of the target is extracted from this image, and is compared with the previously determined center of rotation. The offset between these two values is the x-y positional error of the process. The same process described above to compare the extracted target radius can be reapplied to determine the z-positioning error.

The estimation of H and the equation of Π was repeated 6 times using 4,8, ... 32 respective target locations, distributed as evenly as possible over the extent of the table. While minimally 4 correspondences are sufficient to calculate H and 3 measurements for Π, a more accurate least squares estimate can be obtained with a greater number of measurements.

For each resulting estimate, the robot was positioned over another pattern of 5 target locations using Eq.1 and the positioning error was measured as described above. The sum of the positional errors in each direction and the total error, as a function of the number of target locations, are plotted in Fig.4. Whereas the z error continues to decrease with increasing number of targets, the x, y, and total error are minimal for 12 locations.

In a second stage, a robust estimation of H and Π were computed by selecting from the set of 32 target locations the subset of 23 locations with a reprojection error that was less than a threshold value. Those points with a larger reprojection error were believed to bias the result and were removed. The targets were then placed at an array of 6x13 (i.e., a total of 78) positions evenly spaced over the table. The robot was positioned above each target as described, and the positioning error was measured. Fig.5 plots the linearly interpolated distribution for the total error values as a function of their table location. The magnitude of the minimum and maximum errors, and the mean and standard deviation for each directional component of the error and the total error are also summarized in Table I.

It can be observed that the error increases significantly over the leftmost end of the table. The main reason for this behaviour is that these measurements were taken farthest from the gantry home position. The measurement errors and nonlinearities in the mechanism accumulate and are therefore more pronounced in this region. To reduce the effect of this cumulative error on the remainder of the table, the target locations in this region exceeded the above threshold value, and were discarded in the estimation of H and Π. The resulting estimates extrapolate rather than interpolate over this region, and therefore have a limited accuracy therein.

The major source of error is due to the limited accuracy of the gantry positioning mechanism. As this error is systemic and therefore repeatable, we can improve the positioning accuracy by using the estimate of this error as a correction term. The three directional components of the error were configured into functions that returned an estimate of the directional error value for every table position, and this estimate was added as a correction term to the each direction of the determined robot position. The above experiment was then repeated for the 78 target

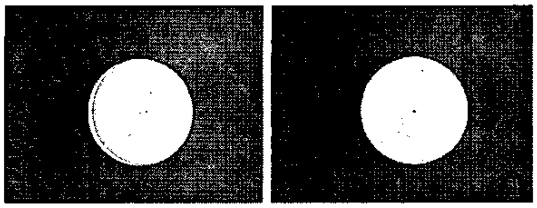

a) without correction b) with correction

Fig. 7. Local Camera View of Target

locations using this correction term, and the new error measurements were plotted in Fig.6. It can be seen that the total error is significantly reduced using this new scheme. The errors for these plots are summarized in Table II. Comparing against the corresponding values in Table I it can be seen that the maximum, mean, and standard deviation of the total error has been significantly reduced.

The benefit of the use of error correction can be seen in Fig.7. The target was placed on the table and its center was extracted in the global camera image. The robot was then positioned over the target using the coordinate calculated from Eq.1. Fig.7a shows the target view from the local camera when the error correction term was not used. The expected target location and radius was overlayed in the image, and the positioning error from the true target location can be clearly seen. In Fig.7b, the robot was repositioned using the error correction term. It can be seen that the overlayed expected location and radius match the true position much more closely when the correction term is used.

VI. Experiment 2: Automatic Potting

At this stage, our goal is to pot a single ball (the cue ball) into a specific corner pocket. This is known as a *scratch*, and is an illegal shot by most rules. It is, however, the most basic skill in pool, and once mastered will lead directly to more complex shots involving object balls.

The experiment will evaluate how accurately the system can automatically pot the cue ball into the target pocket at various distances and angles. We first manually select a pixel point $(u, v)_t$ in the global camera image at the center of the "jaw" of the target pocket. This point indicates the target pocket location. The cue ball is then identified in the global camera image using the same process as the circular target extraction described in Sec.IV. The cue ball center location $(u, v)_c$ and $(u, v)_t$ are both mapped into their respective R coordinates, ${}^R(x, y, z)_c$ and ${}^R(x, y, z)_t$, using Eq.1. The R vector connecting these two points is denoted as $\vec{V}_{ct}$, and indicates the direction from which the cue should strike the ball.

To place the shot, it is necessary to position the cue at a specific fixed height above the table, at a certain fixed offset distance along $\vec{V}_{ct}$ from ${}^R(x, y, z)_c$. The fixed height and offset distances were predetermined manually so that the cue would strike the ball after it has accelerated to the desired strike velocity. In human play, it is common for the cue to have a slight downward angle when striking

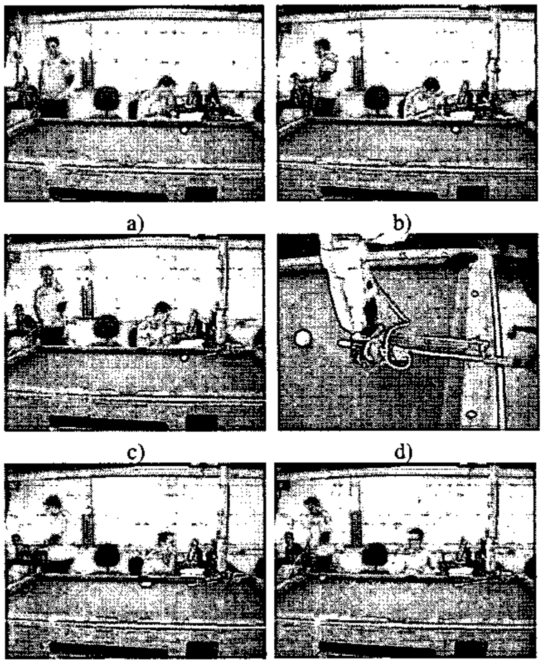


Fig. 8. Motion Sequence When Placing Shot

the cue ball, and so the cue was angled slightly from the horizontal by adjusting joint 5. This configuration had the benefit of allowing the end-effector to clear the table rails without colliding as shown in Fig.8d. The offset distance was selected to be large enough so that the cue tip did not touch the table surface at the end of its travel.

Once the robot position was calculated, the robot was repositioned using a sequence of trapezoidal motions, as illustrated in Fig.8a-f. This sequence was designed to position the robot correctly without colliding with the rails or the cue ball prior to placing the shot. Starting from the home position (a), the robot is raised to its maximum z-height and repositioned in joints 1,2,4, and 5. This placed it in the correct x,y location with the correct shooting angles, raised above the table (b). The robot next drew back the cue and lowered in z (c), and placed the shot (e). If the positions were calculated correctly and the positioning accuracy sufficient, then the cue ball should sink into the target pocket (f).

The cue ball was placed at a variety of positions on the table, and the success with which the system potted was recorded. The shots where taken at a single moderate striking velocity of 1 m/s, which is approximately 1/3 of the cue's maximum strike velocity. For each location, three separate shots were placed, and the results of these experiments were summarized in Fig.9, where the ∘ symbols indicate the table positions that resulted in successful pots, and × indicate those positions that resulted in failed pots. Overall, the system had a success rate of 67%, potting 44 of the 66 shots. The shots that were not successful occurred more frequently in certain regions of the table, away from

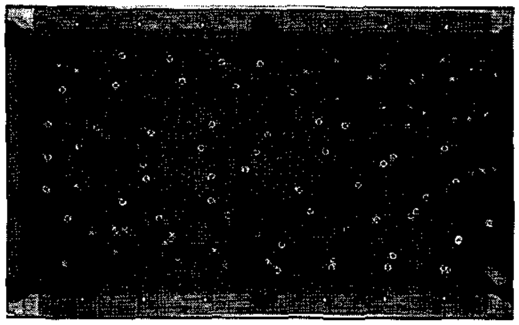


Fig. 9. Potting Results, 1 m/s (o=pot, x=miss)

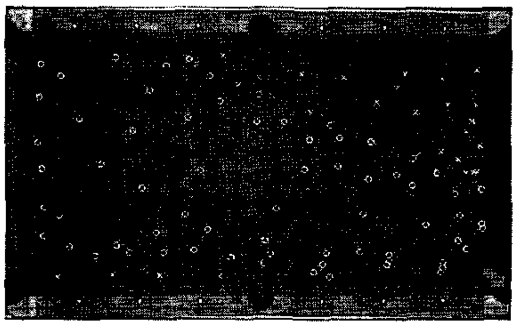


Fig. 10. Potting Results, 2 m/s (o=pot, x=miss)

the central corridor, and were not particularly strongly correlated with distance from the pocket. This is indicative of a remaining systemic error in the gantry positioning, and indeed there was a noticeable and repeatable offset when the gantry was positioned in the regions of the table adjacent to the target pocket, i.e., the lower regions of Fig.9.

This experiment was repeated for a striking velocity of 2 m/s, and the results plotted in Fig.10. The pattern of successful pots is very similar to those in the previous trials, with the number of pots dropping toward the edges of the table. It was expected that the success rate would tend to drop with a higher striking velocity, as tends to occur in human play. This was not observed, however, and indeed, it was noticed that the success rate in the lower extremity of the table actually increased at the higher velocity. The explanation is that that the slight positional inaccuracies over this table region imparted a spin on the ball, which was beneficial to the shooting direction, and therefore increased potting success.

VII. Summary and Future Work

Pool is a game that requires a high level of positional accuracy. Despite its popularity and long history, there exists no information about the positioning accuracy that is required to play effectively. Without an a priori accuracy constraint, the approach that we have taken is to implement the system based upon a general gantry robot, determine its potting performance experimentally, and iteratively improve the accuracy until a suitable level of performance is achieved.

In this work, we have improved the accuracy of a gantry using a calibration technique based upon an overhead global camera, and an end-effector mounted local camera. We have demonstrated that the resulting accuracy is sufficient to pot a single ball reliably over certain regions of the table. The high degree of repeatability makes it likely that the system would be extremely reliable at potting in these regions.

Our experiments also show that there are some regions of the table over which the gantry accuracy is insufficent to reliably pot with the given setup. One possible course of action could be to improve the accuracy by revisiting some of our assumptions. For example, we are currently modelling the robot coordinate frame as a Cartesian system with a linear non-parametric correction term. It may be more suitable to develop a higher order parametric estimate of this error term. Rather than one overhead camera, it may also be beneficial to use two or more cameras, each with a longer focal length, which therefore resolve the target and ball locations to a greater precision.

An alternative approach to improve accuracy is to make use of a local vision system mounted on the end-effector. The role of the global vision system would then be to identify the ball locations in the table coordinate frame, and the local vision system would correct for gantry positioning errors by comparing the locations of the balls in the local image with their known positions in the table frame. In this way, the gantry positioning system requires only a limited accuracy, which is more than likely satisfied using the techniques presented here.

Potting object balls is similar to the process described here, but requires greater positioning accuracy. In the future, we plan to implement the local vision system to improve positional accuracy. We will also implement a ball identification method based upon color indexing, and collision detection. Beyond that, developing a competitive system will require research into the physics of cue, ball, and table interactions [7], and strategy.

Acknowledgements

We gratefully acknowledge NSERC, CFI, IRIS, and our industrial sponsors for partially funding this work.

References

[1] Sang William Shu. *Automating Skills Using a Robot Snooker Player*. PhD thesis, Bristol University, 1994.

[2] S.C. Chua, E.K. Wong, Alan W.C. Tan, and V.C. Koo. Decision algorithm for pool using fuzzy system. In *iCAiET 2002: Intl. Conf. AI in Eng. & Tech.*, pages 370–375, June 2002.

[3] L.B. Larsen, M.D. Jensen, and W.K. Vodzi. Multi modal user interaction in an automatic pool trainer. In *ICMI 2002: 4th IEEE Intl. Conf. Multimodal Interfaces*, pages 361–366, Oct. 2002.

[4] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[5] Z. Zhang. Flexible calibration by viewing a plane from unknown orientations. In *ICCV'99: Int. Conf. Comp. Vis.*, pages 666–673, Sept. 1999.

[6] Samuel M. Thomas and Y.T. Chan. A simple approach for the estimation of circular arc center and its radius. *CVGIP*, 45:362–370, 1989.

[7] Jack H. Koehler. *The Science of Pocket Billiards*. Sportology Publications, 1989.