# Stochasticks: Augmenting the Billiards Experience with Probabilistic Vision and Wearable Computers

Tony Jebara, Cyrus Eyster, Josh Weaver, Thad Starner, Alex Pentland
{jebara, cyruse, joshw, thad, sandy}@media.mit.edu
Media Laboratory, Massachusetts Institute of Technology
Cambridge, MA 02139, (617) 253-0370

## Abstract

*We propose a practical application of wearable computing and augmented reality which enhances the game of billiards. A vision algorithm is implemented which operates in interactive-time with the user to assist planning and aiming. Probabilistic color models and symmetry operations are used to localize the table, pockets and balls through a video camera near the user's eye. Classification of the objects of interest is performed and each possible shot is ranked in order to determine its relative usefulness. The system allows the user to proceed through a regular pool game while it automatically determines strategic shots. The resulting trajectories are rendered as graphical overlays on a head mounted live video display. The wearable video output and the computer vision system provide an integration of real and virtual environments which enhances the experience of playing and learning the game of billiards without encumbering the player.*

## 1 Introduction

The integration of virtual and real environments promises to yield many interesting applications that combine the natural, tangible and rich interactiveness of physical reality with the freedom from constraints and precision of the digital world. Wearable computers are a particularly convenient platform for this symbiosis and make this integration as seamless as possible [17] [12]. One area where this cooperation would be especially interesting is in the game of billiards. This pastime involves a complex blend of physical dexterity as well as analytic geometry and strategy. To enhance the experience, it would be desirable to supplement the user's ability to select, visualize and aim without reducing the tactile feedback of the real game [8]. This intermediate form of integration lies conceptually somewhere between a merely physical game of billiards and a sterile and unnatural computerized version of it. A middle ground between these two extremes can be achieved by using computer vision to map real world data into a virtual context. Similarly, a head mounted display could remap the virtual results of any strategic analysis onto an image of the user's surroundings [6]. The use of a wearable computer allows the user to navigate freely in an arbitrary physical space without special preparation.

This paper describes a wearable computer-based system that uses probabilistic computer vision to guide the user's game play throughout a match. We expect the user to be a novice player who has yet to become familiar with the game and can quickly benefit from the system as a training aid. Figure 1 depicts the desired output of the system. Given an image of a pool table, compute the best shot by locating the pockets and the balls and computing the easiest shot that can be performed. Then, display the paths the relevant balls should follow.

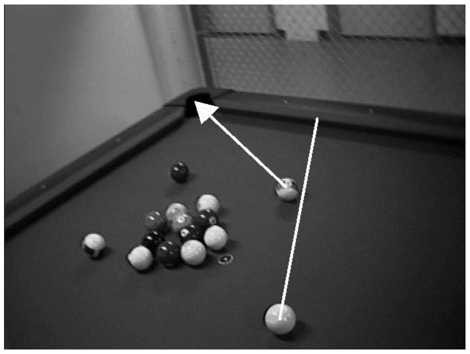


Figure 1: The Desired System Output

Although we have chosen to focus on one particular application (billiards enhancement and training), the motivation behind this work is the integration of augmented reality with computer vision and wearable devices. This integration brings a new dimension to computer-human interaction by essentially reducing the interface barrier. The computer is brought closer to the user in physical terms (wearable), in input (vision and perceptual intelligence) and output (augmented reality and visually integrated output).

We begin by describing some earlier work in the areas of wearable augmented reality and entertainment. We then introduce our system, describing the hardware equipment and the required algorithms. The vision algorithm components including probabilistic color modeling and symmetry computations are explained. The classification and recognition of different balls and pockets is discussed. Subsequently, we present a method for ranking strategic shots from the computed loci of the balls and pockets. The integrated system is finally evaluated and important issues as well as future work are presented.

## 2 Wearable Augmented Reality

Traditionally, immersive environments and augmented reality have required large and non-transferable components. For example, flight simulators and even heads-up-displays (HUDs) have required large, powerful and immobile equipment. However, recent work has reduced the requirements for virtual reality to 1 or 2 workstations, a display and a single camera. For instance, the Pfinder [18] and the Survive [15] systems require relatively minimal computational resources to produce a virtual interactive experience. This virtual experience involves visual tracking of the user and includes interacting with agents as well as arcade gaming. The natural progression would be a totally wearable, person-centric augmented reality with head mounted displays where vision would track the world instead of the user.

The use of such personal imaging and augmented reality has been discussed and investigated in [17], [12] and [16]. This previous research has shown some of the advantages and issues of using computer vision on the external environment to assist the wearable augmented reality experience.

Other current research in interactive environments has stressed the importance of maintaining intermediate representations between the real and the virtual. These help maintain the rich multi-modality we have come to expect [7]. For instance, Ishii and Ullmer [8] include physical objects as input to the computer to preserve tangibility in the virtual experience. Feiner *et. al.* [6] use the notion of overlayed graphical output on see-through head-mounted displays to keep the rich visual stimuli of the real world.

## 3 System Overview

The following is a coarse description of the Stochasticks system along with some important design issues and practical considerations. Both hardware and algorithmic structure are discussed.

### 3.1 Hardware Overview

The wearable computer is the intended platform for the system and it includes a head-mounted display, head-mounted video camera and central processing unit. The layout and dataflow are depicted in Figure 2.

The head-mounted camera is aligned with the user's head and the primary positions of the eyes (straight ahead). Thus, active control or foveation of the camera on the target (i.e. the pool table) is not necessary since the user's head direction will automatically direct the camera to areas of interest in the scene. This style of imaging is unique to wearable computers and head-mounted cameras and greatly reduces the task of a vision system.

Other input is not required since the system relies only on live video information for input. In addition, it is best if the user is not encumbered with any irrelevant input or output paraphernalia. The billiards game requires the use of both hands so any typed input would be unacceptable. In fact, any extra user-interaction beyond the usual play action is likely to be intrusive. Consequently, the algorithm must run autonomously. The system needs to act like an agent in the background and needs to be intelligent enough to make do with the input it has and to offer useful output. Essentially, the only real interaction from the user is from the natural steering and alignment of the head mounted camera.

The display is head-mounted as well, and a video image of the camera's input is continuously projected into each eye. Two CRTs are driven by one video camera who's focal length is adjusted to avoid severe angular modification of the user's expected visual field. The head-mounted unit is shown in Figure 3. This display creates an immersive environment since the user's perspective is now identical to that of the vision algorithm. In addition, the user's view is constrained in the same way as that of the vision algorithm. In other words, eye movement is limited and the user will un-

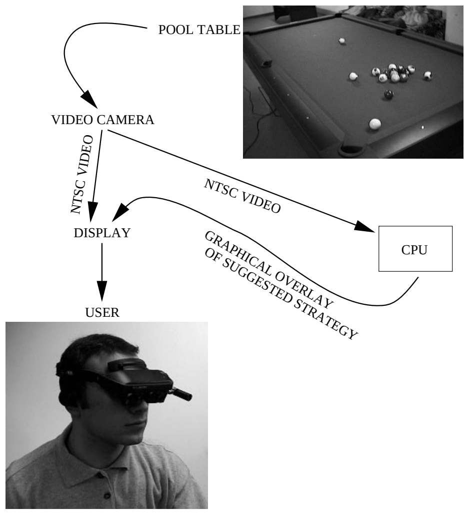


Figure 2: The System Components

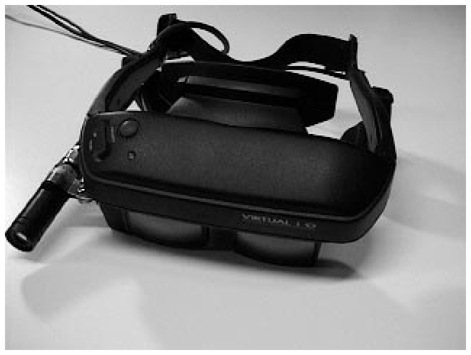


Figure 3: The Head Mounted Video Unit

consciously steer the video camera with head motion to achieve a satisfactory video image projection on the display. This maintains a good quality image for the vision algorithm. The use of the camera display as a virtual "third eye" avoids alignment discrepancies between the vision algorithm and the user's view point. Thus, any graphical overlays that are produced as a result of video processing can be directly projected to the user's eye display [6]. Although the change of perspective induced by looking through a "third eye" (the camera) is unusual for first timers, we have found that users can adapt quite quickly if the camera is only slightly offset from the eyes [14]. We are currently investigating more advanced display techniques which combine the graphical overlays directly onto the user's natural binocular field. This is done using 3D estimation techniques where a full 3D model of the world is recovered so that we can reproject different views of it, customized to each eye's point of view.

The CPU can be a local computer fully contained in the wearable device or it may be a remote computer (or network of computers) which receives a transmitted version of the video signal and sends graphical information back to the user's display. However, since massive computational resources are assumed to be unavailable both locally and remotely, the algorithm must remain efficient enough to operate quickly on standard hardware. Initial development of the algorithm has been made on a 170 Mhz SGI R5000 Indy, the destination platform (a local wearable CPU) is a 150 Mhz Intel Pentium architecture with a 56001 DSP co-processor on the video digitizer board. These two systems have similar performance levels which we shall use to bound the complexity of the vision algorithm.

### 3.2 Algorithmic Overview

The objects of interest in the billiards game (the table, the pockets and the balls) possess some properties which make it particularly well-suited for computer vision. An important feature that comes to mind is color. The green felt of the table and the distinctive contrasting colors of the balls and pockets allow robust color segmentation. In addition, the structures we are tracking have simple geometric shapes. The balls and pockets have a standard circular structure so radial symmetry can be used.

Figure 4 depicts the algorithm loop that iterates repeatedly over video data. The system begins by searching the image for a region that contains green felt colors. The largest green felt patch corresponds to the top of the table and its contour is determined. The interior of the table is then analyzed to find balls by computing symmetry. The centers of circular objects are detected in this region. Similarly, the periphery of the table is also processed to compute radial symmetry and generates symmetry peaks at the centers of the pockets. Each radial symmetry peak is then classified to determine which object generated it (ball, pocket, table). This is done by analyzing the colors of the symmetric objects and checking them against a database of color models for each ball, pocket and table. Once all symmetry peaks are classified, we can compute linear trajectories from the cue ball to every

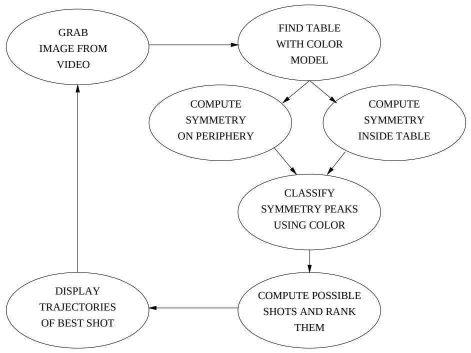
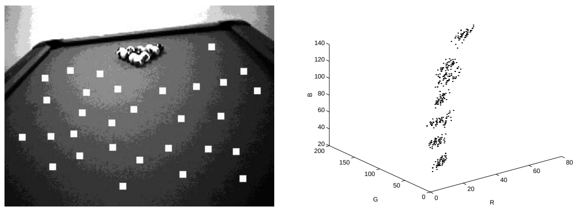

GRAB IMAGE FROM VIDEO
FIND TABLE WITH COLOR MODEL
COMPUTE SYMMETRY ON PERIPHERY
COMPUTE SYMMETRY INSIDE TABLE
CLASSIFY SYMMETRY PEAKS USING COLOR
COMPUTE POSSIBLE SHOTS AND RANK THEM
DISPLAY TRAJECTORIES OF BEST SHOT

Figure 4: The Algorithm's Main Loop



(a) Training pixels (b) RGB distribution

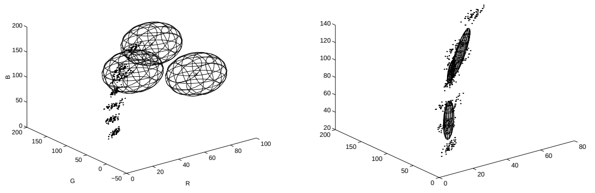
Figure 5: The color training samples

other solid or striped ball (depending on what the user selects). Then, we compute trajectories from each of these to a destination pocket. Each of these trajectories is evaluated to see if it is feasible and rated in difficulty depending on how accurate the shot must be to sink the ball. The easiest shot is then displayed on-screen using linear trajectories. These trace out the paths of the cue ball and the ball to sink.

## 4 Vision Processing

### 4.1 Color Feature Detection

To find the pool table, we train a probabilistic color model of the green felt that covers it [18] [9]. This is done by taking multiple training samples of several images of a pool table under many imaging situations (offline). Each pixel in this distribution forms a 3 element vector, [R G B] which corresponds to the red, green and blue components of the color. We perform



(a) Random Initialization (b) EM Convergence

Figure 6: Fitting a Gaussian Mixture

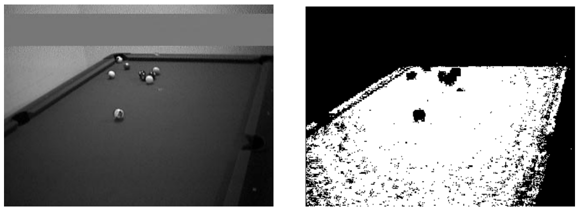

(a) Input Image (b) Color Classification

Figure 7: Pool Table Localization

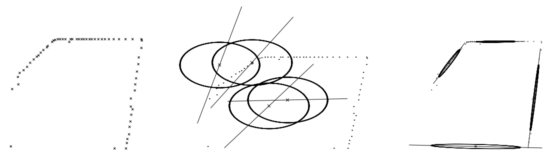

(a) Contour (b) Start (c) Converge

Figure 8: Pool Table Contour Lines via EM

clustering on this distribution of pixels which is shown in Figure 5. The clustering uses Expectation Maximization (EM) to find a probability distribution model for pool table colors [5] [19]. This model is a mixture of Gaussians (the appropriate number of Gaussians is determined a priori with cross-validation). The EM algorithm iterates by adjusting the parameters of the Gaussian probability model to maximize the likelihood of the training samples. The probability distribution model starts off in a random configuration and converges to the configuration in Figure 6. The mixture model is described by Equation 1 where **x** is an (R,G,B) vector.

$$p(\mathbf{x}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \frac{\mathrm{p(i)} \times \exp\{-\frac{1}{2}(\mathbf{x}-\mu_{\mathrm{i}})^{\mathrm{T}}\Sigma_{\mathrm{i}}^{-1}(\mathbf{x}-\mu_{\mathrm{i}})\}}{(2\pi)^{(\mathrm{d}/2)} \mid \Sigma_{\mathrm{i}} \mid^{1/2}} \tag{1}$$

When a new image is acquired, the likelihood of each pixel is evaluated using this model and if it is above a threshold of probability, it is labeled as a piece of the pool table. Then, a connected component analysis is used to gather adjacent green pixels to determine larger regions of grouped table pixels in the image. This process is demonstrated in Figure 7. The largest connected region of green pixels is selected as the table top and proceeds through the algorithm for further processing.

## 4.2 Contour Computation

We are particularly interested in the contour of the pool table since this identifies its shape and periphery where the pockets lie. We quickly compute the centroid of the pool table and compute the most radially distant points at each angle from the center as in Figure 8(a).

Since we know that the pool table top is a quadrilateral object we model its contour as a set of 4 lines. This is done by fitting the distribution of contour pixels using an EM algorithm where each sample **x** is a (u,v) vector. In this vector, u is the horizontal coordinate and v is the vertical coordinate of the contour pixel. A mixture model consisting of four Gaussians is updating using EM as shown in Figure 8(b) and (c). The model converges so that the thin Gaussians form lines that approximate the sides of the table. The dominant eigenvector of each Gaussian is along its main axis and is used to compute a linear model as shown.

Thus, we have four line equations describing the boundary of the pool table in each image frame. It is possible to compute intersections to find pool table corners and to determine the lengths of the sides. In addition, we can accurately specify the interior and periphery of the pool table.

## 4.3 Symmetry Detection

We then propose the use of the general symmetry transform [3] [11] [13] [10]. This is an annular sampling region which detects edge configurations that enclose an object. Unlike template matching, a perceptual measure of symmetric enclosure is computed and blob centers are detected. When applied at the appropriate scale this transform consistently detects circular objects. The dark symmetry transform is a variation that restricts the wave propagation so that only dark radial symmetry is computed. In other words, the algorithm will generate a strong response from circular objects that are darker than the background.

Beginning with a phase and edge map of the image, we perform wave propagation. The wave propagation integrates cocircular edge segments in a computationally efficient way. For each point in the image $p$, at each scale or radius $r$ and for each symmetry orientation $\psi$ we find the set of cocircular pairs of edges , $_{r,\psi}(p)$. The magnitude of axial symmetry in the ($p$, $r$, $\psi$) space is as follows:

$$S_{r,\psi}(p) = \sum_{\lambda_i,\lambda_j \epsilon \Gamma_{r,\psi}(p)} \|\lambda_i\| \|\lambda_j\| (\sin\phi/2)^{w_1} \tag{2}$$

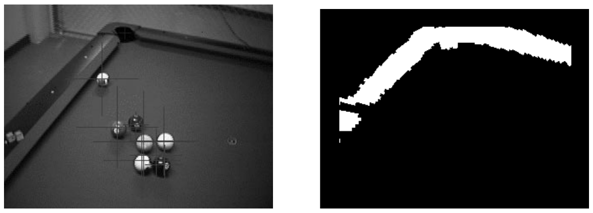
where $\|\lambda_i\|$ and $\|\lambda_j\|$ are the edge intensities of the two co-circular edges and $\phi$ is the angle separating their normals.

Then, radial symmetry, $I(p)$, is determined from the axial symmetry map as in Equation 3 and Equation 4. Finally, the symmetry map undergoes Gaussian smoothing and local maxima are determined.

$$S_\psi(p) = \max_{r=0}^{r_{max}} S_{r,\psi}(p) \tag{3}$$

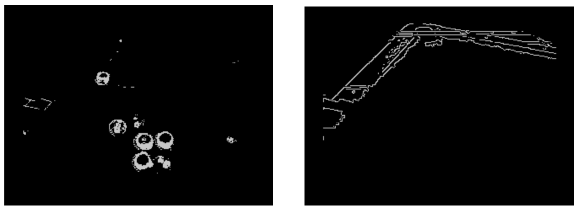
$$I(p) = \sum_{\psi_i,\psi_j} S_{\psi_i}(p) S_{\psi_j}(p) (\sin(\psi_i - \psi_j))^{w_2} \tag{4}$$

We apply the symmetry transform twice for each image. First, on the interior of the table, we apply the general symmetry transform to find both dark and



(a) The pool table (b) Periphery of the table

Figure 9: Pool Table Periphery



(a) Edges in interior (b) Edges in periphery

Figure 10: Edge Detection

bright balls. Then, on the periphery of the table, we apply dark symmetry to find the consistently darker pockets.

Figure 9 shows a pool table and the periphery where we might expect to find pockets (computed from the previous stage). Figure 10 (a) displays the edge map of the interior of the table and (b) displays the edge map of the periphery of the table. Edge maps and phase maps (not shown) are computed using the Sobel operator. The edges due to green portions of the table are suppressed and the edge maps undergo some processing (non-maximal suppression and thresholding).

We then compute the symmetry transforms and obtain peaks of symmetry which have been overlaid on the table image in Figure 11. These peaks are triggered by balls and pockets (mostly) but there are some weak false alarm triggers. So, we wish to filter these candidate balls and pockets to reject false ones and also to label them.

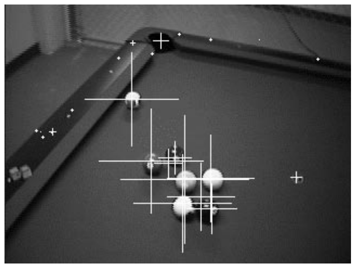
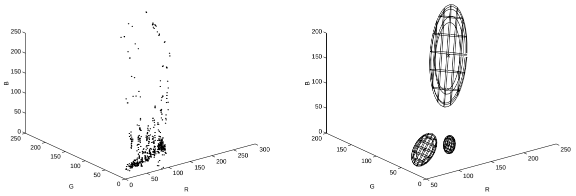
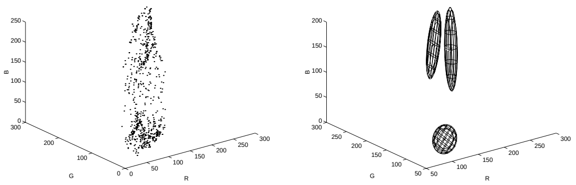


Figure 11: Ball and Pocket Candidates



(a) RGB Data (b) Color Model

Figure 12: The Solid Red Ball (Trained)



(a) RGB Data (b) Color Model

Figure 13: The Striped Orange Ball (Trained)

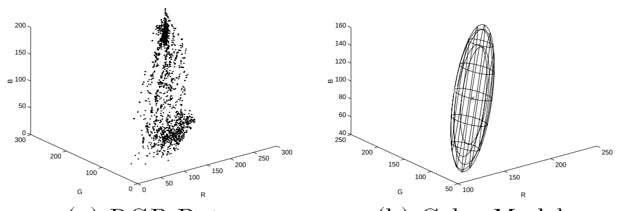
## 4.4 Color Model Classification

We have now localized a set of possible candidates for objects of interest (balls and pockets). To recognize and classify these objects, we propose the use of color models. We train a set of probabilistic color models for the pockets and the balls (cue ball, 8-ball, the seven striped balls and the seven solid balls). We also train models of other objects that might appear in the image as false alarms: a color model of the pool cue and the player's hand. This process is identical to the color modeling used for the table and we obtain a total of 20 models which have a form similar to Equation 1. We shall refer to these models as $p_1(\mathbf{x}), ..., p_{20}(\mathbf{x})$.

Figure 12 and Figure 13 show the color modeling process for the solid red ball and the striped orange ball. For each model, we begin with a distribution of pixels in RGB space as shown in Figure 12(a) and Figure 13(a). These sets of 3 dimensional RGB vectors are labeled $data_1, ..., data_{20}$. For each $data_i$ we compute a model $p_i(\mathbf{x})$ when we first train the system.

Next we apply all 20 color models to examine each of the possible candidate symmetry peaks. Around each peak, we collect a small window of pixels. All non-table pixels (i.e. the ones that did not match the table's color model) are collected to form a distribution of RGB data similar to the one in Figure 14(a). This test distribution will be called $data_t$. We form a single Gaussian color model (because EM with multiple Gaussians would be too slow to compute online). This model, shown in Figure 14(b), is called $p_t(\mathbf{x})$.

To classify our test distribution $data_t$ we use a common distance metric between probabilistic models. This metric is the Kullback-Liebler divergence



(a) RGB Data (b) Color Model

Figure 14: The Test Object (Orange Striped Ball)

[4] [1]. By measuring the 'distance' between test data $p_t(\mathbf{x})$ and our training data, $p_i(\mathbf{x})$ for $i\epsilon[1,20]$ we can see how similar it is to other objects. We determine the closest model $i$ for each symmetry peak and label that peak accordingly. This process is iterated over all the symmetry peaks which are ultimately labeled as solid ball, striped ball, cue ball, 8-ball, pocket and 'other'.

## 5 Strategy and Planning

We now describe a strategy for selecting the shot that is most likely to succeed. Starting from the cue-ball, we compute linear trajectories to all solid balls (or striped balls if the user selects to play as stripes). We then compute linear trajectories from these to the detected pockets. Then we test each of these possible trajectories: from cue-ball to solid-ball to pocket to find those that don't involve any collisions with other objects. If no solid balls are found, we test the 8-ball for possible trajectories.

We then weight each of the 'valid' trajectories by the following heuristics. Each time the user is to take a shot, we are really interested in the angle at which he or she hits the cue ball (we shall assume that the force with which the cue ball is hit is going to be determined by the user). So, what is fundamentally being suggested to the user is an angle at which to hit a cue ball, say $\angle c$ as a deviation from the line connecting the cue ball to the solid ball. The more accuracy we need on $\angle c$, the harder the shot. Figure 15 shows how we can determine the required angle. The distance from the cue-ball to the solid-ball is $l$ and the distance from there to the pocket is $L$. The angle formed by the three objects is $\angle a$ at the solid ball. We can compute $\angle c$ using this information as well as a bound on the maximum error on $\angle c$ which is $\pm\angle d$.

For each possible shot, we compute $\angle d$ and display to the user the shot with the greatest value of $\angle d$ (the shot which requires the least accuracy) with the required trajectory (along $\angle c$) superimposed on the image. This is a simplified first-order strategy model since it is only 2D and does not include spin effects, kinematics or rebounds in the computation of the shots.

[1] We use a symmetric form of the divergence and approximate the solution using the weak law of large numbers

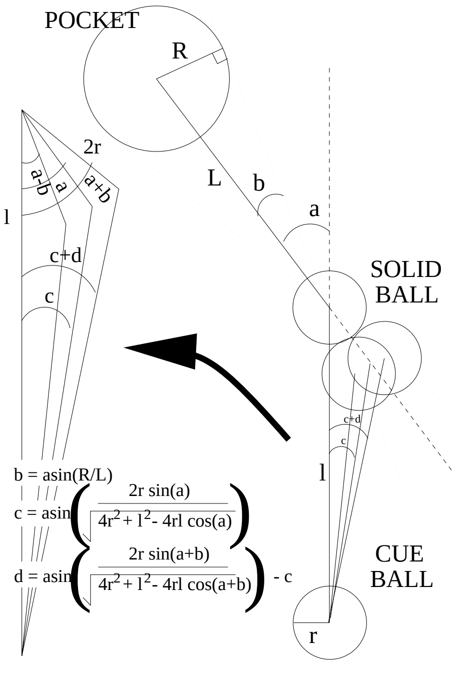

POCKET
R
2r
L
b
a
a-b
a
a+b
l
c+d
c
SOLID BALL
c+d
c
l
CUE BALL
r
b = asin(R/L)
c = asin(2r sin(a) / √(4r² + l² - 4rl cos(a)))
d = asin(2r sin(a+b) / √(4r² + l² - 4rl cos(a+b))) - c

Figure 15: Determining the Angle and the Difficulty

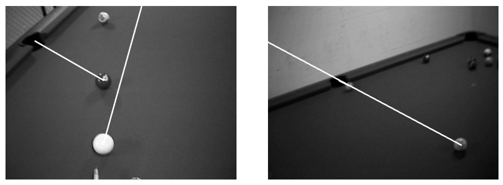


Figure 16: Suggested Shots

## 6 Output and Interaction Issues

Figure 16 shows a few sample shots that the system suggested during the course of normal play. These are the images directly projected onto the user's eye. At this point, we are undertaking a performance analysis of the overall system. The reliability of the algorithm is being investigated as well as its accuracy. In addition, time lag from input to output is currently on the order of hundreds of milliseconds. Thus, the user has to maintain smooth, slow head motion and occasionally wait for the system's output. We are investigating techniques for making the algorithm more efficient to avoid lag which is particularly distracting when the user moves excessively. An evaluation of the effect of the system on the user's billiards skill level is also to be performed. For instance, how much better (or worse) does the user play depending on his or her initial degree of competence? How much adaptation time is generally required to see positive improvements? Another issue is offering multiple styles of output display graphics to suit the particular preferences of the user. Finally, we wish to evaluate the user's non-augmented

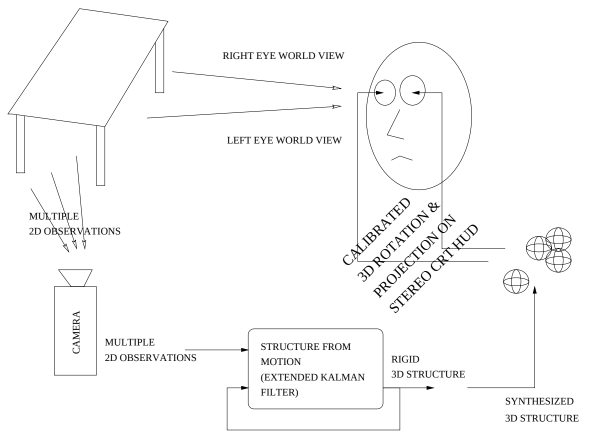


Figure 17: Recovery and Synthesis of 3D Models

performance after he or she has been 'trained' with the Stochasticks system. It would be interesting to see if the user acquires a greater visualization ability as a result of the augmented reality experience and to analyze the change in hand-eye coordination due to the head-mounted display.

## 7 Future Work

Current efforts are being made to port the system to run locally and efficiently on an untethered wearable computer with only on-board processing. This would enable multiple users to engage in a usability study where the system can be tested on a variety of pool tables in more diverse situations.

One important addition in progress is the inclusion of a 3D estimation stage which would analyze object locations over time to compute a 3D structural estimate of the scene. This would maintain virtual positions of balls and pockets that fell out of the current image view. 3D recursive estimation and 3D alignment algorithms could increase the precision and stability of the output [9] [2] [1].

In addition, we are using the recovery of 3D structure to combine the system's output on the user's natural binocular field (instead of the video camera's images). This allows a much more natural display since the distortion due to the camera's awkward placement on the side of the head can be factored out and a virtual camera view (centered at each eye) can be simulated. Figure 17 depicts the recovery of 3D information and its use for calibrated graphical overlay reprojection.

The user sees the world through the stereo, transparent HUD and the computer sees it through its video camera. However, using a recursive structure from motion estimator, we can accumulate multiple 2D views of the objects in the scene to converge to a 3D model of the scene. The 3D model can then be analyzed and used to generate appropriate synthetic 3D overlays. A simple 3D transformation maps the synthetic 3D model into the view point of each eye. This 3D transformation is determined manually during initialization by having the user adjust the 3D rendering software's view point. Assume that Figure 18(a) is the image seen by the user's left eye. Figure 18(b) is a recovered 3D model of the pool table synthesized in a real-time 3D software package and displayed on the left CRT of the HUD. The user changes the view in the 3D display such that it matches the image seen behind the CRT (of the true pool table) and the system is calibrated to synthesize images for the left eye as shown in Figure 18(c). The calibration stays constant and the user can freely move while the synthetic overlays rotate to compensate with the head rotation. The system, however, requires very accurate, frame to frame tracking of the pool table to estimate its 3D pose and structure (which is necessary for synthesizing the 3D overlays).

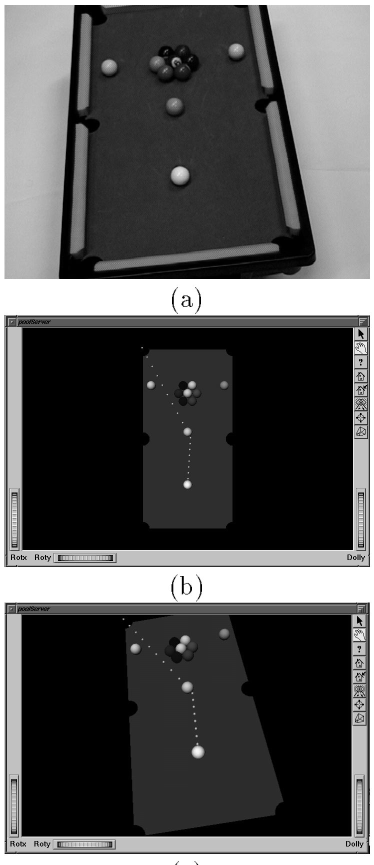


Figure 18: Calibration of 3D Output

We are also investigating more sophisticated methods for strategy and planning which would consider higher-order effects such as rebounds, multiple collisions as well as preparatory placement for subsequent shots.

Other sensors could be added to the system to increase precision. For example, if a laser pointer could be mounted in the pool cue, we could determine the exact impact point on the cue ball and compute english and spin to predict more complex trajectories. Also, tracking the pool stick would allow the system to realize when its suggested shot is being disregarded so that it may assist the user in completing another shot.

## 8 Conclusions

A wearable-based augmented reality system has been demonstrated. An intermediate virtual space has been generated where the user can play a game of billiards, maintaining tactile interaction with the physical objects and maintaining real visual sensing while the system simultaneously overlays a graphical output to suggest shots and assist targeting. Through the use of computer vision and wearable computing, we enhance the game without encumbering the user or limiting playability. An autonomous probabilistic vision algorithm drives the Stochasticks system providing it with the necessary intelligence to make a useful contribution to the human player.

### Acknowledgments

The authors express their thanks to the reviewers for their valuable comments.

## References

[1] T.D. Alter, "3D Pose from 3 Corresponding Points under Weak-Perspective Projection", *A.I. Memo No. 1378*, 1992.

[2] A. Azarbayejani and A. Pentland, "Recursive Estimation of Motion, Structure and Focal Length", *IEEE Pattern Analysis and Machine Intelligence*, June 1995.

[3] M. Bolduc, G. Sela and M.D. Levine, "Fast Computation of Multiscalar Symmetry in Foveated Images", *Proceedings of the Conference on Computer Architectures for Machine Perception*, pp. 2-11, 1995.

[4] T. Cover and J. Thomas, *Elements of Information Theory*, John Wiley & Sons Inc. 1991.

[5] A. Dempster, N. Laird and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society*, B39, pp. 1-38, 1977.

[6] S. Feiner, B. MacIntyre, D. Seligmann, "Annotating the Real World with Knowledge-Based Graphics on a See-Through Head-Mounted Display", *Proceedings of Graphics Interface '92*, p 78-85, 1992.

[7] EP. Glinert and MM. Blattner, "Multimodal Interaction", *IEEE Multimedia*, 3(4):13, Winter 1996.

[8] H. Ishii and B. Ullmer, "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms", *Human Factors in Computing Systems: CHI '97 Conference Proceedings*, pp. 234-241, 1997.

[9] T. Jebara and A. Pentland, "Parametrized Structure from Motion for 3D Adaptive Feedback Tracking of Faces", *IEEE International Conference on Computer Vision and Pattern Recognition*, 1997.

[10] T. Jebara, "3D Pose Estimation and Normalization for Face Recognition", Bachelor's Thesis*McGill Centre for Intelligent Machines*, 1996.

[11] M.F. Kelly and M.D. Levine, "Annular Symmetry Operators: A Method for Locating and Describing Objects", *Fifth International Conference on Computer Vision*, pp. 1016-1021, 1995.

[12] S. Mann, "Smart Clothing: Wearable Multimedia Computing and Personal Imaging...", *Proceedings of ACM Multimedia* pp. 163-174, November 1996.

[13] D. Reisfeld and Y. Yeshurun, "Robust Detection of Facial Features by Generalized Symmetry", *11th IAPR International Conference on Pattern Recognition*, Vol. 1, pp. 117-120, 1992.

[14] J. Rolland, F. Biocca, T. Barlow and A. Kancherla, "Quantification of Adaptation to Virtual-Eye Location in See-Thru Head-Mounted Displays", *IEEE Annual Virtual Reality International Symposium* p. 56-66, 1995.

[15] K. Russell, T. Starner and A. Pentland, "Unencumbered Virtual Environments", *1995 International Joint Conference on Artificial Intelligence Entertainment and AI/ALife Workshop*, 1995.

[16] R. Sharma and J. Molineros, "Role of Computer Vision in Augmented Virtual Reality", *Proceedings of SPIE - The International Society for Optical Engineering*, 2409 pp. 220-231, 1995.

[17] T. Starner, S. Mann, B. Rhodes, J. Levine, J. Healey, D. Kirsch, R. Picard, and A. Pentland, "Augmented Reality through Wearable Computing", *Presence*, 1997.

[18] C. Wren, A. Azarbayejani, T. Darrell, A. Pentland, "Pfinder: Real-Time Tracking of the Human Body", *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, 1996 .

[19] L. Xu and M. Jordan, "On Convergence Properties of the EM Algorithm for Gaussian Mixtures", *Neural Computation*, 8, 129-151, 1996.