A COMPUTER VISION TANGIBLE USER INTERFACE FOR MIXED REALITY BILLIARDS

Brian Hammond
Pace University
Seidenberg School of Computer Science
brian@brianhammond.com

ABSTRACT

Conventional input devices such as the mouse and keyboard are unnatural and limited for various forms of human-computer interaction. We have created a tangible user interface that mixes the physical world with a virtual reality billiards game. Our system uses computer vision techniques to analyze images acquired from a single inexpensive digital video camera in real-time in order to passively sense the full 3D pose of a markerless cue stick. A physically-modeled billiards game updates the virtual cue stick to mimic detected poses and shots. Our system affords users with a convenient, noninvasive, inexpensive, and natural interface for entertainment or training without requiring a billiards table installation. Moreover, our system is robust to changes in illumination and occlusion, and is noteworthy for its use of computer vision as the sole means of sensing user input.

Index Terms— computer vision, tangible user interface, VR, billiards, games

1. INTRODUCTION

The most common input devices involved in human-computer interaction today are the mouse and keyboard. Such input devices do not provide natural interfaces to various classes of software. We argue that software developers and interaction designers are constrained to design simple, often unnatural human-computer interactions due to the expected use of the mouse and keyboard for user input.

The goal of this work is to prove by example that the creation of noninvasive and natural human-computer interfaces is possible with consumer grade hardware, computer vision techniques, and common unmarked physical objects. To this end we have created a prototype tangible user interface (TUI) to a virtual reality (VR) pocket billiards game in which a single inexpensive digital video (DV) camera mounted atop a tripod is employed to passively watch a user’s interactions with a markerless billiards cue stick and cue ball. The 3D pose of the cue stick is estimated in real-time given acquired images from the DV camera. The pose of a cue stick avatar in the VR billiards game mimics that of the physical cue stick. Collisions between the physical cue stick and cue ball (shots) are detected and simulated in the billiards game as well. Our system expects a particular scene layout as depicted in Figure 1. Interactions between TUI elements (referred to as “tokens” following [10]) occur on or above a flat surface such as the top of a table.

Note that our TUI does not require a physical billiards table installation. Moreover, billiards balls other than the cue ball exist only in the VR billiards game. We assume the reader has at least a cursory knowledge of billiards equipment (e.g. cue stick, billiards balls, billiards table) and the basic forms of gameplay (e.g. aiming, shotting).

2. RELATED WORK

The foundation of research on TUIs was laid in the Tangible Bits initiative at MIT [4] and Fitzmaurice’s Ph.D thesis on graspable user interfaces [2] in which the general goal was “to bridge the gaps between both cyberspace and the physical environment” [4]. Krueger’s VIDEOPLACE [7] sparked interest in the use of cameras as a passive input mechanism, inferring user intent by watching users instead of forcing them to create inputs by way of a physical device. Our TUI
does not require special magnetic sensors, electronic tags, or fiducials (cf. [6]). Furthermore, our goal is to require a minimal number of proprietary tokens (cf. [10, 6]), focusing interaction design on the use of natural tangible objects for a given context (cf. [1]). Several other systems have been developed that augment the billiards experience in one form or another. In [3], a haptics-based control interface to a VR billiards game was created, achieving highly accurate measurements of shot forces. Our goal is to obviate the need for haptics by allowing users to interact with real objects instead of haptic models thereof. Moreover, the haptic models of [3] do not account for important ball spin effects in input sensing, and is more expensive to construct than our system. In [5], a user wears a HMD whose view of a physical billiards table is augmented with arrows offering the billiards player an insight of the potential outcome of a shot. While such a system may help train users, the use of a HMD is an unnatural and expensive design option. Also, we wish to avoid the requirement of a physical billiards table installation.

3. SYSTEM ARCHITECTURE

Our TUI prototype uses a client-server architecture in which a server process performs computer vision based token detection, cue stick pose estimation, and shot analysis while a client process manages a distributed computer load to different computers if needed.

4. THE EFFECTS OF IMAGE QUALITY ON MEASUREMENT ACCURACY

We faced several challenges in obtaining accurate measurements using computer vision techniques only for sensing user input due to the desired use of an inexpensive consumer-grade DV camera. Such cameras are not generally manufactured for the utmost in image quality. Indeed, we experienced image quality issues such as geometric distortion, blooming, sensor noise, and chromatic aberration. To combat these issues, we balanced image quality and overall system performance by using camera calibration, temporal averaging, and weighted moving average techniques.

We used Intel OpenCV to determine the distortion coefficients of our DV camera’s lens in a one-time calibration process. At runtime, we undistort acquired images using these coefficients. Lines that were curved prior to undistortion become straight afterwards with obvious benefit in shape finding. To combat sensor noise, we found that averaging several (4 worked the best for us) recently acquired images significantly reduced the effects of sensor noise, albeit with a noticeable drop in effective acquisition rate and thus a drop in system response time. An alternative to such temporal averaging is to use weighted moving averages in which the effective value of some measurement is averaged with the past $N$ such values with larger influence or weight given to more recent measurements (e.g. with linear falloff). We have found that weighted moving averages are better suited to deal with noisy images than temporal averaging as more image-based measurements are possible per unit time which can still be effectively smoothed to combat the numerical “ jitter” caused by sensor noise, quantization artifacts, subtle changes in scene illumination, etc.

5. TOKEN DETECTION

Our goal is to accurately estimate the pose of the cue stick given color images of the scene as depicted in Figure 1-N. We define the pose of the cue stick as the state vector $(h, \theta, \psi, dist_w, a, b)$ where $h$ is the vertical offset of the tip of the cue stick from the plane of the table, $\theta$ is the pitch of the cue stick, $\psi$ is the yaw of the cue stick, $dist_w$ is the distance in world space between the cue tip and the closest point on the cue ball, and $a$ and $b$ are predicted coefficients of spin about the cue ball’s horizontal and vertical axes respectively (see [8] for a depiction of such axes).

The first step in our token detection process is to detect the cloth by finding pixels in each acquired image that match a flexible color model. Such a color model has been designed to handle variance in ambient room brightness as well as the poor color reproduction quality of consumer-grade DV cameras. We have found the HSV color space to be a good choice for our image processing work as hue is independent of saturation and value. For example, we may wish to detect image regions of green hue whether dark or bright, muted or vibrant. To this end we have devised a simple color image operator in which an destination binary image contains white pixels where corresponding pixels in the source image fit a flexible color model. We define a flexible color model as $(\bar{H} \pm H_\Delta, \bar{S} \pm S_\Delta, \bar{V} \pm V_\Delta)$ where $\bar{H}$ is the mean hue, $\bar{S}$ is the mean saturation, $\bar{V}$ is the mean value, $H_\Delta$ is the allowed variance in hue from $\bar{H}$, $S_\Delta$ is the allowed variance from $\bar{S}$, and $V_\Delta$ is the allowed variance from $\bar{V}$. We say that a pixel of an HSV image fits a flexible color model when the hue component of the pixel falls within the range $[\bar{H} - H_\Delta, \bar{H} + H_\Delta]$; likewise for saturation and value. Note that in practice this image operator must properly handle the case when the allowed hue range wraps around from $360^\circ$ to $0^\circ$ or vice versa.

The stages of our token detection algorithm can be seen in Figure 2. We first apply our color matching operator to the acquired image using a flexible color model defined for the cloth (e.g. $(80^\circ \pm 60^\circ, 50\% \pm 50\%, 50\% \pm 50\%)$), resulting in a binary image $I_{cloth}$ as seen in Figure 2-B. We perform a connected component analysis on $I_{cloth}$ which results in a hierarchy of connected components or contours. For instance, we consider the black pixels that comprise the region of the cue ball as a child of the cloth contour (the parent in this relationship) since this region is nested inside the cloth contour. We select the largest contour by area as the contour representing the cloth token (the white region of Figure 2-B). To detect the cue ball we select the child of the cloth contour that maximizes the circularity metric $area/radius$ where $area$ is the area of candidate contour and $radius$ is the radius of a circle $C$ that tightly encloses the candidate contour.

We next apply the flexible color matching operator to find regions that lay inside the cloth contour that represents shadows using a flexible color model with a low value range, disregarding hue and saturation (e.g. $(180^\circ \pm 180^\circ, 50\% \pm 50\%, 8\% \pm 8\%)$). Since the destination binary image will detect dark areas throughout the image, we logically AND the destination image with $I_{cloth}$. The resultant image $I_{shadow}$ resembles Figure 2-C. We perform connected component analysis on $I_{shadow}$ and remove contours from the result set with an $area < \epsilon$ to deal with noise. We select the topmost image space contour as the shadow of the cue stick and the bottommost contour as the shadow of the cue ball; distance measures are based on each contour’s centroid.

To detect the cue stick we first find the convex hull of the cloth
The distance of each component of the cue stick state vector. We begin by estimating the pose of the cue stick, which must estimate a value for \( M \) and \( T \). The cue stick contour has been enclosed inside the cloth contour. (E) The most circular child of the cloth contour has been filled with white. (F) Screenshot in which the cloth, cue ball, the cue ball shadow, cue stick, and the cue stick shadow have been highlighted.

6. CUE STICK POSE ESTIMATION

To estimate the pose of the cue stick we must estimate a value for each component of the cue stick state vector. We begin by estimating the distance \( h \) of the cue stick at its tip to the plane of the table. We have adapted the work of Reid and North [9] in which the 3D location of a soccer ball is determined using computer vision and projective geometry techniques. Their algorithm requires the image location of the vanishing line of the plane of the playing field, the top and bottom of an arbitrary reference object, the shadow of the top of the reference object, the object of interest, and the shadow of the object of interest. Adapted to our scene, the object of interest is the tip of the cue stick. We have described how to find the image location of the tip of the cue stick as well as its shadow in Section 5. Our TUI allows an arbitrary reference object (see Figure 1-L) and requires the user to select the image locations of the top, bottom, and shadow of the top of the reference object (see Figure 1-M) once at the start of a session using a mouse. We find the vanishing line of the plane of the table by approximating the cloth contour with a quadrilateral. We then find the vanishing point of lines that run through opposite sides of this quadrilateral (see the dashed lines of Figure 1). The vanishing line of the plane of the table intersects both of these vanishing points. Note that this vanishing line will be quite a distance outside the bounds of acquired images given our overhead view of the scene. Please refer to [9] for further details.

We define the pitch \( \theta \) of the cue stick as the angle between the plane of the table and cue stick shaft (see Figure 3-A). For instance, when \( \theta = 90^\circ \), the cue stick points down into the surface of the table. Our strategy for estimating \( \theta \) is based on the observation that the angle between the shaft of the cue stick and the shaft’s shadow is roughly equal to the angle between the shaft and the surface of the table (i.e. \( \theta \); see Figure 3-A). Note that this observation is most accurate when the light source is situated next to the cue ball. This has a useful side effect of orienting the shadow of the cue stick to an area where it will generally not be combined with the shadow of the cue stick. Let \( shaft \) be the direction or heading of the shaft of the cue stick in image space. That is, given the image space location of the tip of the cue stick \( T \) and a point on the shaft of the cue stick \( M \), \( shaft = [T - M]\). Let \( shaft' = [M - T\,\,][M - T\,\,]. \) Let \( M_s \) be the image space point of the shadow of \( T \) and \( M_t \) to be the image space point of the shadow of \( M \). Let \( shadow = [M_s - T\,\,][M_t - T\,\,]. \) Thus, \( \theta = \cos^{-1}(shaft \cdot shadow) \).

The yaw \( \psi \) of the cue stick is the angle between \( shaft \) and the image space “down” vector (e.g. \( down = (0, 1) \)). Thus, \( \psi = \cos^{-1}(shaft \cdot down) \). While we could allow the player to fully orbit the cue ball, we instead limit \( \psi \) to the range \([-45, 45]\) and introduce the notion of the “current side” of the billiards table avatar. The current side is the side of the billiards table avatar the user is situated on in the game at any given time. Note that the user remains in the same general physical location and shoots the cue ball in the same general direction throughout their playing session.

Changing the current side to the next side results in the cue stick avatar being rotated by \( 90^\circ \) from its current yaw. When we wish \( \psi \) to be in the range \([-45, 45]\) we must define when \( \psi \) is negative and positive. We have arbitrarily chosen that the \( \psi > 0 \) when the cue stick points down and to the right in image space and \( \psi < 0 \) when the cue stick points down and to the left. See Figures 3-C and 3-D for examples. The billiards game treats \( \psi \) as a relative change in yaw from the current side when orienting the cue stick avatar.

The distance between the tip of the cue stick and the closest point on the cue ball is the indicator of when a shot occurs for when this distance is 0, the cue stick tip is touching the cue ball. Let \( R_u \) be the cue ball radius in world units (known a priori) and \( C_r \) be the radius of the cue ball in image space. We thus derive the number of world units per pixel with \( R_u/C_r \). Let \( P \) be the intersection point of \( shaft \) and \( C \). If no such intersection exists, the system notifies the user that the cue stick is not pointing at the cue ball. Otherwise, we define \( dist_u = (R_u/C_r)[P - T] \) (see Figure 3-E).

Striking the cue ball off-center (in addition to the effects of friction) will cause the cue ball to undergo some amount of spin. We wish to define the amount of spin that a user might impart to the cue ball given a pose of the cue stick and assuming that the user only moves the cue stick along its current heading into the cue ball. Let \( a \) denote the amount of left spin \( (a < 0) \) or right spin \( (a > 0) \) and \( b \) denote the amount of backspin \( (b < 0) \) or topspin \( (b > 0) \) that results from such an off-center collision. To estimate \( a \) we use the point of intersection \( P \) of \( shaft \) with the circle \( C \) to get \( a = (P_x - C_x)/C_r \). To estimate \( b \) we relate the vertical offset of the cue tip above the plane of the table to the vertical offset of the center of the cue ball above the plane of the table. Since we know the diameter of the cue ball a priori, we can find \( b \) with \( b = (h - R_u)/R_u \). Note that we restrict \( a \) and \( b \) as percentages of \( R_u \) to allow the client billiards game to use any size sphere to represent billiards ball avatars.

7. SHOT DETECTION

A shot occurs in a billiards game when the user strikes the stationary cue ball with the cue stick, causing a collision between the stick and ball. Given our constraint of consumer-grade hardware and latency caused by our image processing, we will not generally be able to directly sense such a collision due to undersampling. However, we can estimate the cue stick speed and the time of collision given only “before” and “after” images of the scene. To help us detect shots...
Fig. 3. Cue stick pose. (A) Example of $\theta \approx 45^\circ$. (B) Top view of (A). (C) Example of $\psi > 0$. (D) Example of $\psi < 0$. (E) Example of $\text{dist}_{w} = |T - P|$.

Fig. 4. Example of distance history. The vertical axis denotes $\text{dist}_{w}$ while the horizontal axis denotes time. (A) User taking several practice strokes. (B) Most recent local maximum. (C) User striking the cue ball.

we use a distance history which is a history of $\text{dist}_{w}$ measurements over time. In Figure 4 we see the measured distance of the cue stick over a 10 second time period. Since we are not in general able to detect the cue stick at a distance of exactly 0, we monitor the distance history observations for a zero crossing which indicates a collision between the cue stick and the cue ball. To determine the velocity of the cue stick at the time of collision we find the most recent local maximum (see Figure 4-B) and assume a constant velocity over the time between this local maximum and the zero crossing. The velocity is thus the distance traveled over the given time interval.

8. BILLIARDS GAME

While our system can support multiple TUI clients, we have focused on one such client, a rigid body dynamics based 3D billiards game. The major responsibilities of the client process are to physically model the dynamics of rigid objects such as billiards balls, cue sticks, and billiards tables, perform collision detection and response, receive server messages over TCP/IP, orient the cue stick avatar to mimic the pose of the cue stick token, derive the forces of each shot on the cue ball, physically model spin effects, and to render the scene in 3D. Other features such as game logic will not be discussed in this paper.

The billiards game has three main states: aiming, shooting, and simulation. The client receives messages from the server during the aiming state containing the full cue stick pose state vector as described in Section 5. The game orients the cue stick avatar given the pose state vector of the physical cue stick. The shooting state is active when a shot message is received by the server. The forces applied to the cue ball are based on the work of [8]. The simulation state is active when there are one or more billiards balls in motion.

9. CONCLUSIONS

We have described a TUI that augments the billiards playing experience using solely computer vision for sensing input. Our prototype system relies heavily on freely available libraries such as Intel OpenCV, Newton Game Dynamics, Raknet, and OGRE. Our implementation suffers from the use of consumer-grade hardware due to image quality issues (numerical jitter, system latency). The quality of our system is dependent on the hardware used making objective analysis difficult. However, we note that our pose detection algorithms are generally within 10% of ground truth. We feel that computer vision as an input sensor will become a more viable technique as higher quality camera hardware becomes cheaper. In future work we plan on researching several similar vision-based TUIs to determine generalities and best practices. We have found the combination of a VR billiards game coupled with a physical control interface affords users with the potential to practice difficult shots (e.g. bank shots, combinations, etc.) or simply have fun while at the same time learning the feel of taking shots using a physical cue stick.


10. REFERENCES


