Machine Vision & Applications manuscript No.
(will be inserted by the editor)

Computer Vision-Based Gesture Tracking, Object Tracking, and 3D Reconstruction for Augmented Desks

Thad Starner[1], Bastian Leibe[2], David Minnen[1], Tracy Westyn[1], Amy Hurst[1], and Justin Weeks[1]

[1] Contextual Computing Group, GVU Center
Georgia Insititute of Technology
e-mail: {thad,dminn,turtle,sloopy,joostan}@cc.gatech.edu

[2] Perceptual Computing and Computer Vision Group, ETH Zurich
Haldeneggsteig 4, CH-8092 Zurich, Switzerland
e-mail: leibe@inf.ethz.ch

The date of receipt and acceptance will be inserted by the editor

Abstract The Perceptive Workbench endeavors to create a spontaneous and unimpeded interface between the physical and virtual worlds. Its vision-based methods for interaction constitute an alternative to wired input devices and tethered tracking. Objects are recognized and tracked when placed on the display surface. By using multiple infrared light sources, the object's 3D shape can be captured and inserted into the virtual interface. This ability permits spontaneity since either preloaded objects or those objects selected at run-time by the user can become physical icons. Integrated into the same vision-based interface is the ability to identify 3D hand position, pointing direction, and sweeping arm gestures. Such gestures can enhance selection, manipulation, and navigation tasks. The Perceptive Workbench has been used for a variety of applications, including augmented reality gaming and terrain navigation. This paper focuses on the techniques used in implementing the Perceptive Workbench and the system's performance.

Key words gesture – 3D object reconstruction – tracking – computer vision – virtual reality

1 Introduction

Humans and computers have interacted primarily through devices that are constrained by wires. Typically, the wires limit the distance of movement and inhibit freedom of orientation. In addition, most interactions are indirect. The user moves a device as an analogue for the action to be created in the display space. We envision an untethered interface that accepts gestures directly and can accept any objects the user chooses as interactors. In this paper, we apply our goal to workbenches, large tables, which serve simultaneously as projection display and as interaction surface. Originally proposed in 1995 by Krueger *et al* [15], they are now widely used in virtual reality and visualization applications.

Computer vision can provide the basis for untethered interaction because it is flexible, unobtrusive, and allows direct interaction. Since the complexity of general vision tasks has often been a barrier to widespread use in real-time applications, we simplify the task by using a shadow-based architecture.

An infrared light source is mounted on the ceiling. When the user stands in front of the workbench and extends an arm over the surface, the arm casts a shadow on the desk's surface, which can be easily distinguished by a camera underneath.

The same shadow-based architecture is used in the Perceptive Workbench [19,18] to reconstruct 3D virtual representations of previously unseen real-world objects placed on the desk's surface. In addition, the Perceptive Workbench can illuminate objects placed on the desk's surface to identify and track the objects as the user manipulates them. Taking its cues from the user's actions, the Perceptive Workbench switches between these three modes automatically. Computer vision controls all interaction, freeing the user from the tethers of traditional sensing techniques.

In this paper, we will discuss implementation and performance aspects that are important to making the Perceptive Workbench a useful input technology for virtual reality. We will examine performance requirements and show how our system is being optimized to meet them.

2 Related Work

While the Perceptive Workbench [19] is unique in its ability to interact with the physical world, it has a rich heritage of related work [1,14,15,23,26,34,35,37,43]. Many augmented desk and virtual reality designs use tethered props, tracked by electromechanical or ultrasonic means, to encourage interaction through gesture and manipulation of objects [3,1,26,32,

37]. Such designs tether the user to the desk and require the time-consuming ritual of donning and doffing the appropriate equipment.

Fortunately, the computer vision community has taken up the task of tracking hands and identifying gestures. While generalized vision systems track the body in room- and desk-based scenarios for games, interactive art, and augmented environments [2,44], the reconstruction of fine hand detail involves carefully calibrated systems and is computationally intensive [22]. Even so, complicated gestures such as those used in sign language [31,38] or the manipulation of physical objects [28] can be recognized. The Perceptive Workbench uses such computer vision techniques to maintain a wireless interface.

Most directly related to the Perceptive Workbench, Ullmer and Ishii's "Metadesk" identifies and tracks objects placed on the desk's display surface using a near-infrared computer vision recognizer, originally designed by Starner [34]. Unfortunately, since not all objects reflect infrared light and since infrared shadows are not used, objects often need infrared reflective "hot mirrors" placed in patterns on their bottom surfaces to aid tracking and identification. Similarly, Rekimoto and Matsushita's "Perceptual Surfaces" [23] employ 2D barcodes to identify objects held against the "HoloWall" and "HoloTable." In addition, the HoloWall can track the user's hands (or other body parts) near or pressed against its surface, but its potential recovery of the user's distance from the surface is relatively coarse compared to the 3D pointing gestures of the Perceptive Workbench. Davis and Bobick's SIDEshow [6] is similar to the Holowall except that it uses cast shadows in infrared for full-body 2D gesture recovery. Some augmented desks have cameras and projectors above the surface of the desk; they are designed to augment the process of handling paper or interact with models and widgets through the use of fiducials or barcodes [35,43]. Krueger's VideoDesk [14], an early desk-based system, uses an overhead camera and a horizontal visible light table to provide high contrast hand gesture input for interactions which are then displayed on a monitor on the far side of the desk. In contrast with the Perceptive Workbench, none of these systems address the issues of introducing spontaneous 3D physical objects into the virtual environment in real-time and combining 3D deictic (pointing) gestures with object tracking and identification.

3 Goals

Our goal is to create a vision-based user interface for VR applications. Hence, our system must be responsive in real-time and be suitable for VR interaction. In order to evaluate the feasibility of meeting this goal we need to examine the necessary performance criteria.

3.1 System Responsiveness

System responsiveness, the time elapsed between a user's action and the response displayed by the system [41], helps determine the quality of the user's interaction. Responsiveness requirements vary with the tasks to be performed. An acceptable threshold for object selection and manipulation tasks is typically around 75 to 100 ms [39,41]. System responsiveness is directly coupled with latency. It can be calculated with the following formula:

$$Syst.Responsiveness = Syst.Latency + DisplayTime \quad (1)$$

System latency, often also called device lag, is the time it takes our sensor to acquire an image, calculate and communicate the results, and change the virtual world accordingly. Input devices should have low latency, ideally below 50 ms. Ware and Balakrishnan measured several common magnetic trackers and found them to have latencies in the range of 45 to 72 ms [39].

In our situation, system latency depends on the time it takes the camera to transform the scene into a digital image, image processing time, and network latency to communicate the results. Given an average delay of 1.5 frame intervals at 33 ms per interval to digitize the image results in a 50 ms delay. In addition, we assume a 1.5 frame interval delay in rendering the appropriate graphics. Assuming a constant 60 frame per second (fps) rendering rate results in an additional 25 ms delay for system responsiveness. Since we are constrained by a 75 ms overhead in sensing and rendering, we must minimize the amount of processing time and network delay in order to maintain an acceptable latency for object selection and manipulation. Thus, we concentrate on easily computed vision algorithms and a lightweight UDP networking protocol for transmitting the results.

3.2 Accuracy

With the deictic gesture tracking, we estimate that absolute accuracy will not need to be very high. Since the pointing actions and gestures happen in the three dimensional space high above the desk's surface, discrepancies between a user's precise pointing position and the system's depiction of that position is not obvious or distracting. Instead, it is much more important to capture the trend of movement and allow for quick correctional motions.

For the object tracking, however, this is not the case. Here, the physical objects placed on the desk already give a strong visual feedback and any system response differing from this position will be very distracting. This constraint is relatively easy to satisfy, though, since the task of detecting the position of an object on the desk's surface is, by nature, more accurate than finding the correct arm orientation in 3D space.

4 Apparatus

The display environment for the Perceptive Workbench builds on Fakespace's immersive workbench [40]. It consists of a wooden desk with a horizontal frosted glass surface on which an image can be projected from behind the workbench.

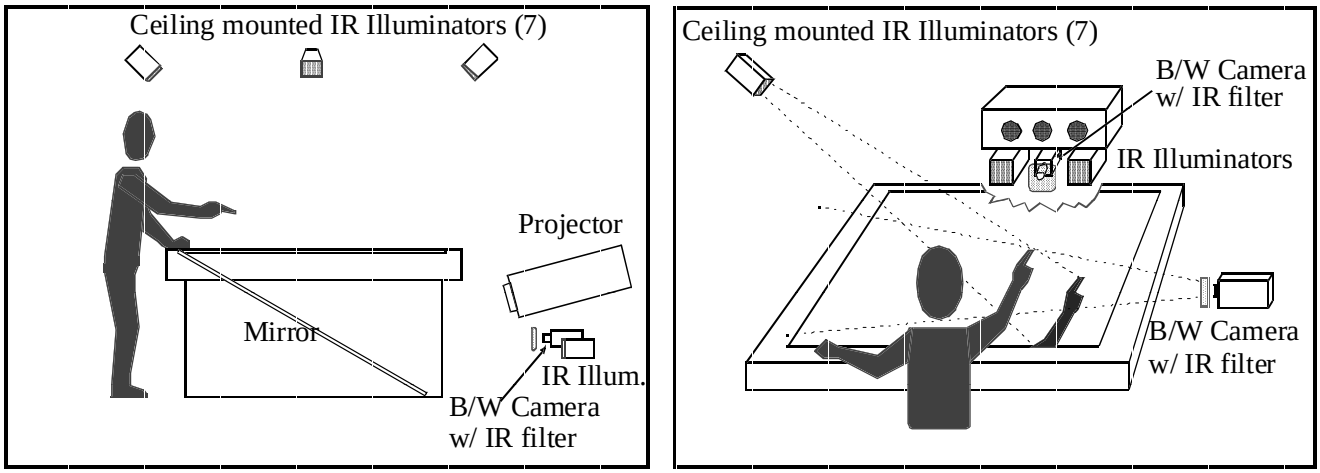


Fig. 1 *Light and camera positions for the Perceptive Workbench. The top view shows how shadows are cast and the 3D arm position is tracked.*

We placed a standard monochrome surveillance camera under the projector to watch the desk's surface from underneath (see Figure 1). A filter placed in front of the camera lens makes it insensitive to visible light and to images projected on the desk's surface. Two infrared illuminators placed next to the camera flood the desk's surface with infrared light that is reflected back toward the camera by objects placed on the desk.

We mounted a ring of seven similar light sources on the ceiling surrounding the desk (Figure 1). Each computer-controlled light casts distinct shadows on the desk's surface based on the objects on the table (Figure 2a). A second infrared camera and another infrared light source are placed next to the desk to provide a side view of the user's arms (Figure 3a). This side camera is used solely for recovering 3D pointing gestures.

Note that at any time during the system's operation, either the ceiling lights, or the lights below the table are active, but not both at the same time. This constraint is necessary in order to achieve reliable detection of shadows and reflections.

We decided to use near-infrared light since it is invisible to the human eye. Thus, illuminating the scene does not interfere with the user's interaction. The user does not perceive the illumination from the infrared light sources underneath the table, nor the shadows cast from the overhead lights. On the other hand, most standard charge-coupled device (CCD) cameras can still see infrared light, providing an inexpensive method for observing the interaction. In addition, by equipping the camera with an infrared filter, the camera image can be analyzed regardless of changes in (visible) scene lighting.

We use this setup for three different kinds of interaction:

- Recognition and tracking of objects placed on the desk surface based on their contour
- Tracking of hand and arm gestures
- Full 3D reconstruction of object shapes from shadows cast by the ceiling light-sources.

For display on the Perceptive Workbench we use OpenGL, the OpenGL Utility Toolkit (GLUT) and a customized version of a simple widget package called microUI (MUI). In addition, we use the workbench version of VGIS, a global terrain visualization and navigation system [40] as an application for interaction using hand and arm gestures.

5 Object Tracking & Recognition

As a basic precept for our interaction framework, we want to let users manipulate the virtual environment by placing objects on the desk surface. The system should recognize these objects and track their positions and orientations as they move over the table. Users should be free to pick any set of physical objects they choose.

The motivation behind this is to use physical objects in a "graspable" user interface [9]. Physical objects are often natural interactors as they provide physical handles to let users intuitively control a virtual application [11]. In addition, the use of real objects allows the user to manipulate multiple objects simultaneously, increasing the communication bandwidth with the computer [9, 11].

To achieve this tracking goal, we use an improved version of the technique described in Starner et al. [30]. Two near-infrared light-sources illuminate the desk's underside (Figure 1). Every object close to the desk surface (including the user's hands) reflects this light, which the camera under the display surface can see. Using a combination of intensity thresholding and background subtraction, we extract interesting regions of the camera image and analyze them. We classify the resulting blobs as different object types based on a 72-dimensional feature vector reflecting the distances from the center of the blob to its contour in different directions.

Note that the hardware arrangement causes several complications. The foremost problem is that our two light sources under the table can only provide uneven lighting over the

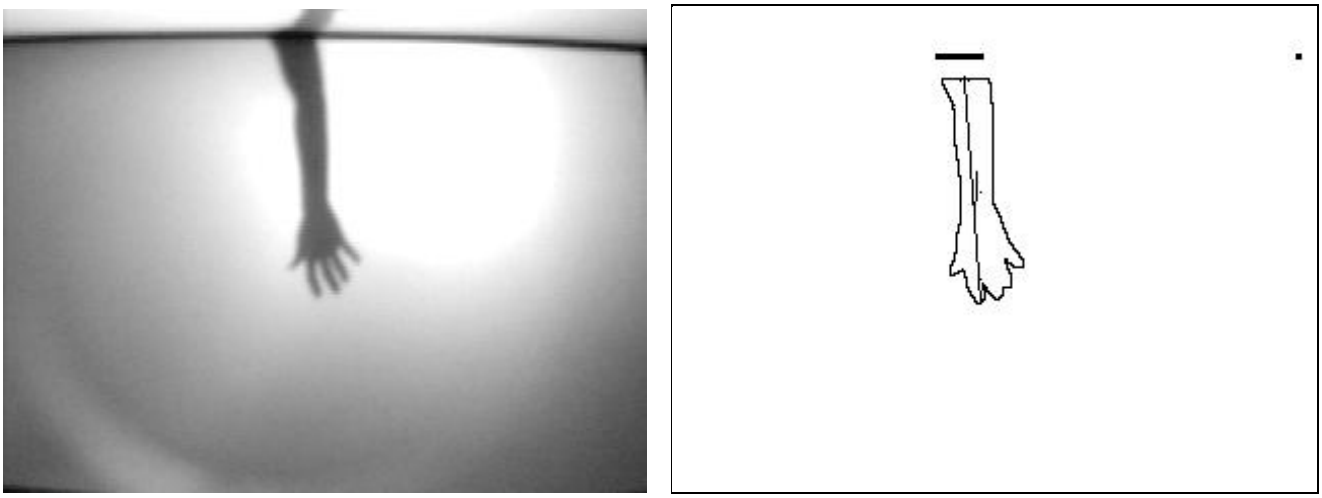


Fig. 2 *(a) Arm shadow from overhead IR lights; (b) resulting contour with recovered arm direction.*

whole desk surface. In addition, the light rays are not parallel, and the reflection on the mirror surface further exacerbates this effect. To compensate for this, we perform a dynamic range adjustment. In addition to a background image, we store a "white" image that represents the maximum intensity that can be expected at any pixel. This image is obtained by passing a bright white (and thus highly reflective) object over the table during a one-time calibration step and instructing the system to record the intensity at each point. The dynamic range adjustment helps to normalize the image so that a single threshold can be used over the whole table. An additional optimal thresholding step is performed for every blob to reduce the effects of unwanted reflections from users' hands and arms while they are moving objects. Since the blobs only represent a small fraction of the image, the computational cost is low.

In order to handle the remaining uncertainty in the recognition process, two final steps are performed: detecting the stability of a reflection and using tracking information to adjust and improve recognition results. When an object is placed on the table, there will be a certain interval when it reflects enough infrared light to be tracked but is not close enough to the desk's surface to create a recognizable reflection. To detect this situation, we measure the change in size and average intensity for each reflection over time. When both settle to a relatively constant value, we know that an object has reached a steady state and can now be recognized. To further improve classification accuracy, we make the assumption that objects will not move very far between frames. Thus, the closer a blob is to an object's position in the last frame, the more probable it is that this blob corresponds to the object and the less reliable the recognition result has to be before it is accepted. In addition, the system remembers and collects feature vectors that caused some uncertainty (for example, by an unfamiliar orientation that caused the feature vector to change) and adds them to the internal description of the object, thus refining the model.

In this work, we use the object recognition and tracking capability mainly for cursor or place-holder objects. We focus on fast and accurate position tracking, but the system may be trained on a different set of objects to serve as navigational tools or physical icons [34]. A future project will explore different modes of interaction based on this technology.

6 Deictic Gesture Tracking

Following Quek's taxonomy [21], hand gestures can be roughly classified into symbols (referential and modalizing gestures) and acts (mimetic and deictic gestures). Deictic (pointing) gestures depend strongly on location and orientation of the performing hand. Their meaning is determined by the location at which a finger is pointing, or by the angle of rotation of some part of the hand. This information acts not only as a symbol for the gesture's interpretation, but also as a measure of the extent to which the corresponding action should be executed or to which object it should be applied.

For navigation and object manipulation in a virtual environment, many gestures will have a deictic component. It is usually not enough to recognize that an object should be rotated – we will also need to know the desired amount of rotation. For object selection or translation, we want to specify the object or location of our choice just by pointing at it. For these cases, gesture recognition methods that only take the hand shape and trajectory into account will not suffice. We need to recover 3D information about the users' hands and arms in relation to their bodies.

In the past, this information has largely been obtained by using wired gloves or suits, or magnetic trackers [3, 1]. Such methods provide sufficiently accurate results but rely on wires tethered to the user's body or to specific interaction devices, with all the aforementioned problems. We aim to develop a purely vision-based architecture that facilitates unencumbered 3D interaction.

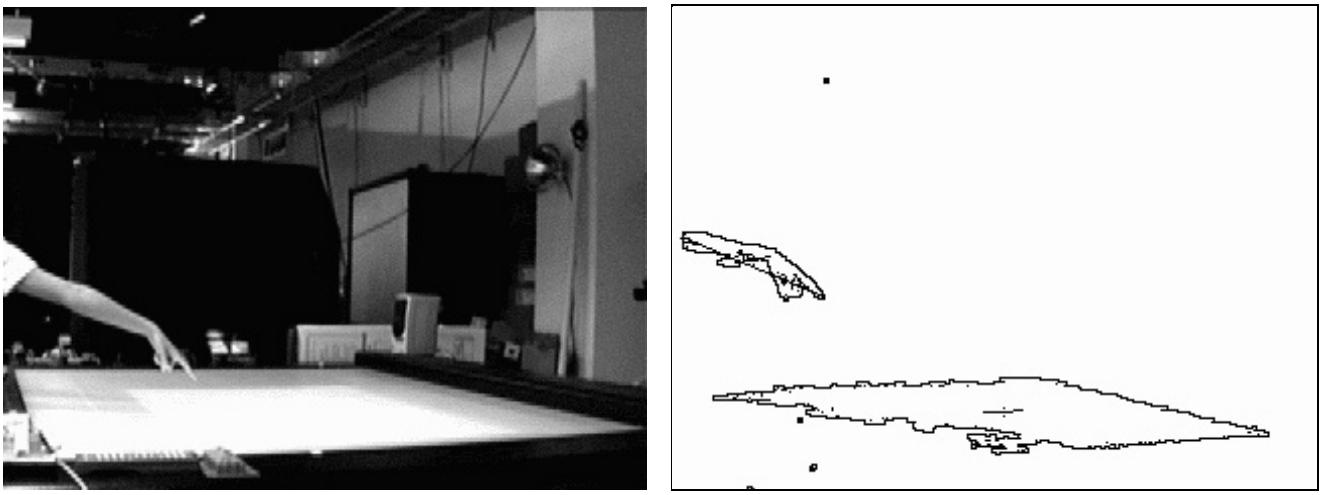


Fig. 3 *(a) image from side camera (without infrared filter); (b) arm contour from similar image with recovered arm direction.*

With vision-based 3D tracking techniques, the first issue is to determine what information in the camera image is relevant – that is, which regions represent the user's hand or arm. What makes this difficult is the variation in user clothing or skin color and background activity. Previous approaches on vision-based gesture recognition used marked gloves [8], infrared cameras [25], or a combination of multiple feature channels, like color and stereo [13] to deal with this problem, or they just restricted their system to a uniform background [36]. By analyzing a shadow image, this task can be greatly simplified.

Most directly related to our approach, Segen and Kumar [27] derive 3D position and orientation information of two fingers from the appearance of the user's hand and its shadow, co-located in the same image. However, since their approach relies on visible light, it requires a stationary background and thus cannot operate on a highly dynamic back-projection surface like the one on our workbench. By using infrared light for casting the shadow, we can overcome this restriction.

The use of shadows solves, at the same time, another problem with vision-based architectures: where to put the cameras. In a virtual workbench environment, there are only few places from where we can get reliable hand position information. One camera can be set up next to the table without overly restricting the available space for users. In many systems, in order to recover three dimensional information, a second camera is deployed. However, the placement of this second camera restricts the usable area around the workbench. Using shadows, the infrared camera under the projector replaces the second camera. One of the infrared light sources mounted on the ceiling above the user shines on the desk's surface where it can be seen by the camera underneath (see Figure 4). When users move an arm over the desk, it casts a shadow on the desk surface (see Figure 2a). From this shadow, and from the known light-source position, we can calculate a plane in which the user's arm must lie.

Simultaneously, the second camera to the right of the table (Figures 3a and 4) records a side view of the desk surface and the user's arm. It detects where the arm enters the image and the position of the fingertip. From this information, the computer extrapolates two lines in 3D space on which the observed real-world points must lie. By intersecting these lines with the shadow plane, we get the coordinates of two 3D points – one on the upper arm, and one on the fingertip. This gives us the user's hand position and the direction in which the user is pointing. We can use this information to project an icon representing the hand position and a selection ray on the workbench display.

Obviously, the success of the gesture-tracking capability relies heavily on how fast the image processing can be done. Fortunately, we can make some simplifying assumptions about the image content. We must first recover arm direction and fingertip position from both the camera and the shadow image. Since the user stands in front of the desk and the user's arm is connected to the user's body, the arm's shadow should always touch the image border. Thus, our algorithm exploits intensity thresholding and background subtraction to discover regions of change in the image. It also searches for areas in which these regions touch the desk surface's front border (which corresponds to the shadow image's top border or the camera image's left border). The algorithm then takes the middle of the touching area as an approximation for the origin of the arm (Figures 2b and Figure 3b). Similar to Fukumoto's approach [10], we trace the shadow's contour and take point farthest away from the shoulder as the fingertip. The line from the shoulder to the fingertip reveals the arm's 2D direction.

In our experiments, the point thus obtained was coincident with the pointing fingertip in all but a few extreme cases (such as the fingertip pointing straight down at a right angle to the arm). The method does not depend on a pointing gesture, but also works for most other hand shapes, including a hand held horizontally, vertically, or in a fist. These shapes may be

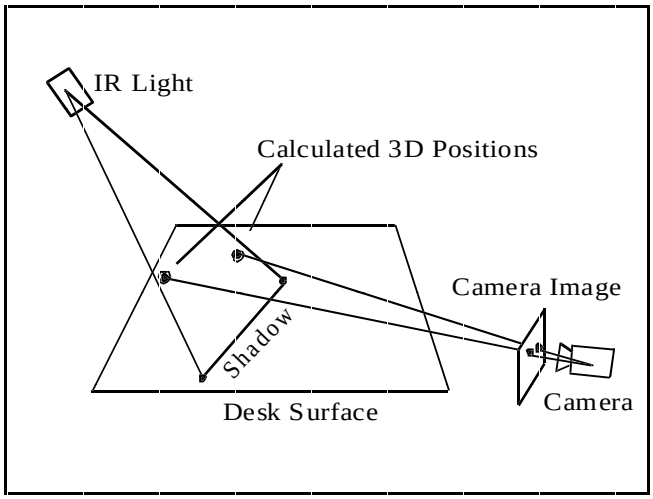


Fig. 4 *Principle of pointing direction recovery.*

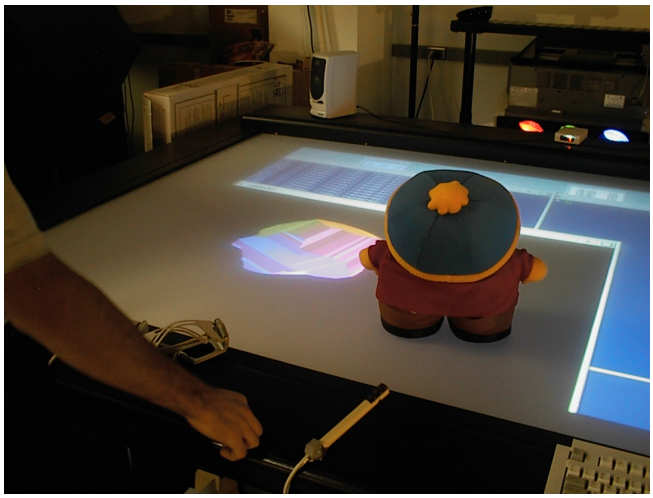


Fig. 5 *Real object inserted into the virtual world. The figure shows a reconstruction of the doll in the foreground.*

distinguished by analyzing a small section of the side camera image and may be used to trigger specific gesture modes in the future.

The computed arm direction is correct as long as the user's arm is not overly bent (see Figure 3). In such cases, the algorithm still connects the shoulder and fingertip, resulting in a direction somewhere between the direction of the arm and the one given by the hand. Although the absolute resulting pointing position does not match the position towards which the finger is pointing, it still captures the trend of movement very well. Surprisingly, the technique is sensitive enough so that users can stand at the desk with their arm extended over the surface and direct the pointer simply by moving their index finger without any arm movement.

6.1 Limitations and Improvements

Figure 3b shows a case where segmentation based on color background subtraction in an older implementation detected both the hand and the change in the display on the workbench. Our new version replaces the side color camera with an infrared spotlight and a monochrome camera equipped with an infrared-pass filter. By adjusting the angle of the light to avoid the desk's surface, the user's arm is illuminated and made distinct from the background. Changes in the workbench's display do not affect the tracking.

One remaining problem results from the side camera's actual location. If a user extends both arms over the desk surface, or if more than one user tries to interact with the environment simultaneously, the images of these multiple limbs can overlap and merge into a single blob. Consequently, our approach will fail to detect the hand positions and orientations in these cases. A more sophisticated approach using previous position and movement information could yield more reliable results, but at this stage we chose to accept this restriction and concentrate on high frame rate support for one-handed interaction. In addition, this may not be a serious limitation for a single user for certain tasks. A recent study shows that for a task normally requiring two hands in a real environment, users have no preference for one versus two hands in a virtual environment that does not model effects such as gravity and inertia [26].

7 3D Reconstruction

To complement the capabilities of the Perceptive Workbench, we want to be able to insert real objects into the virtual world and share them with other users at different locations (see Figure 5). An example application for this could be a telepresence or computer-supported collaborative work (CSCW) system. This requires designing a reconstruction mechanism that does not interrupt the interaction. Our focus is to provide a nearly instantaneous visual cue for the object, not necessarily on creating a highly accurate model.

Several methods reconstruct objects from silhouettes [29, 33] or dynamic shadows [5] using either a moving camera or light source on a known trajectory or a turntable for the object [33]. Several systems have been developed for reconstructing relatively simple objects, including some commercial systems.

However, the necessity to move either the camera or the object imposes severe constraints on the working environment. Reconstructing an object with these methods usually requires interrupting the user's interaction with it, taking it out of the user's environment, and placing it into a specialized setting. Other approaches use multiple cameras from different viewpoints to avoid this problem at the expense of more computational power to process and communicate the results.

In this project, using only one camera and multiple infrared light sources, we analyze the shadows cast by the object from multiple directions (see Figure 6). Since the process

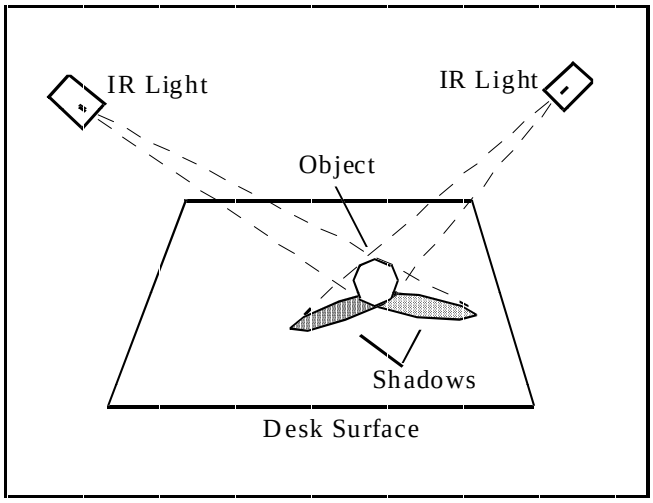


Fig. 6 *Principle of the 3D reconstruction.*

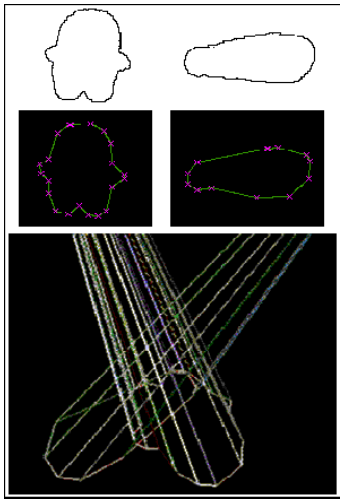


Fig. 7 *Steps of the 3D reconstruction of the doll from Figure 5, including the extraction of contour shapes from shadows and the intersection of multiple view cones (bottom).*

is based on infrared light, it can be applied independently of the lighting conditions and with minimal interference with the user's natural interaction with the desk.

To obtain the different views, we use a ring of seven infrared light sources in the ceiling, each independently switched by computer control. The system detects when a user places a new object on the desk surface, and renders a virtual button. The user can then initiate reconstruction by touching this virtual button. The camera detects this action, and in approximately one second the system can capture all of the required shadow images. After another second, reconstruction is complete, and the newly reconstructed object becomes part of the virtual world. Note that this process uses the same hardware as the deictic gesture-tracking capability discussed in the previous section, and thus requires no additional cost.

Figure 7 shows a series of contour shadows and a visualization of the reconstruction process. By approximating each shadow as a polygon (not necessarily convex) [24], we create a set of polyhedral "view cones" extending from the light source to the polygons. The intersection of these cones creates a polyhedron that roughly contains the object.

Intersecting nonconvex polyhedral objects is a complex problem, further complicated by numerous special cases. Fortunately, this problem has already been extensively researched and solutions are available. For the intersection calculations in our application, we use Purdue University's TWIN Solid Modeling Library [7]. Recently, a highly optimized algorithm has been proposed by Matusik et al. that can perform these intersection calculations directly as part of the rendering process [20]. Their algorithm provides a significant improvement on the intersection code we are currently using, and we are considering it for a future version of our system.

Figure 8c shows a reconstructed model of a watering can placed on the desk's surface. We chose the colors to highlight the different model faces by interpreting the face normal as a vector in RGB color space. In the original version of our software, we did not handle holes in the contours. This feature has since been added by constructing light cones for both the object contours and for those representing holes. By inspecting the pixels adjacent to the outside of the contour, we can distinguish between the two types of borders. Then, rather than intersecting the light cone with the rest of the object, we perform a boolean differencing operation with the cones formed from the hole borders.

7.1 Limitations

An obvious limitation to our approach is that we are confined to a fixed number of different views from which to reconstruct the object. The turntable approach permits the system to take an arbitrary number of images from different viewpoints. In addition, not every nonconvex object can be exactly reconstructed from its silhouettes or shadows. The closest approximation that can be obtained with volume intersection is its visual hull, that is, the volume enveloped by all the possible circumscribed view cones. Even for objects with a polyhedral visual hull, an unbounded number of silhouettes may be necessary for an exact reconstruction [17]. However, Sullivan's work [33] and our experience have shown that usually seven to nine different views suffice to get a reasonable 3D model of the object.

Exceptions to this heuristic are spherical or cylindrical objects. The quality of reconstruction for these objects depends largely on the number of available views. With only seven light sources, the resulting model will appear faceted. This problem can be solved either by adding more light sources, or by improving the model with the help of splines.

In addition, the accuracy with which objects can be reconstructed is bounded by another limitation of our architecture. Since we mounted our light sources on the ceiling, the system can not provide full information about the object's shape.

There is a pyramidal blind spot above all flat, horizontal surfaces that the reconstruction can not eliminate. The slope of these pyramids depends on the angle between the desk surface and the rays from the light sources. Only structures with a greater slope will be reconstructed entirely without error. We expect that we can greatly reduce the effects of this error by using the image from the side camera and extracting an additional silhouette of the object. This will help keep the error angle well below 10 degrees.

8 Performance Analysis

8.1 Object and Gesture Tracking

Both object and gesture tracking currently perform at an average of between 14 and 20 frames per second (fps). Frame rate depends on both the number of objects on the table and the size of their reflections. Both techniques follow fast motions and complicated trajectories.

To test latency, we measured the runtime of our vision code. In our current implementation with an image size of 320*240 pixels, the object tracking code took around 43 ms to run with a single object on the desk surface and scaled up to 60 ms with five objects. By switching from TCP to UDP, we were able to reduce the network latency from a previous 100 ms to approximately 8 ms. Thus, our theoretical system latency is between 101 and 118 ms. Experimental results confirmed these values.

For the gesture tracking, the results are in the same range since the code used is nearly identical. Measuring the exact performance, however, is more difficult because two cameras are involved.

Even though the system responsiveness (system latency plus display lag) exceeds the envisioned threshold of 75 to 100 ms, it still seems adequate for most (navigational) pointing gestures in our current applications. Since users receive continuous feedback about their hand and pointing positions, and most navigation controls are relative rather than absolute, users adapt their behavior readily to the system. With object tracking, the physical object itself provides users with adequate tactile feedback. In general, since users move objects across a very large desk, the lag is rarely troublesome in the current applications.

Nonetheless, we are confident that some improvements in the vision code can further reduce latency. In addition, Kalman filters may compensate for render lag and will also add to the tracking system's stability.

8.2 3D Reconstruction

Calculating the error from the 3D reconstruction process requires choosing known 3D models, performing the reconstruction process, aligning the reconstructed model and the ideal model, and calculating an error measure. For simplicity, we chose a cone and pyramid. We set the centers of mass of the ideal and reconstructed models to the same point in space, and aligned their principal axes.

	Cone	Pyramid
Maximal Error	0.0215 (7.26%)	0.0228 (6.90%)
Mean Error	0.0056 (1.87%)	0.0043 (1.30%)
Mean Square Error	0.0084 (2.61%)	0.0065 (1.95%)

Table 1 *Reconstruction errors averaged over three runs (in meters and percentage of object diameter).*

To measure error, we used the Metro tool developed by Cignoni, Rocchini, and Scopigno [4]. It approximates the real distance between the two surfaces by choosing a set of 100,000 to 200,000 points on the reconstructed surface, then calculating the two-sided distance (Hausdorff distance) between each of these points and the ideal surface. This distance is defined as $\max(E(S_1, S_2), E(S_2, S_1))$ with $E(S_1, S_2)$ denoting the one-sided distance between the surfaces S_1 and S_2:

$$E(S_1, S_2) = \max_{p \in S_1}(dist(p, S_2)) = \max_{p \in S_1}(\min_{p' \in S_2}(dist(p, p'))) \quad (2)$$

The Hausdorff distance corresponds directly to the reconstruction error. In addition to the maximum distance, we also calculated the mean and mean-square distances. Table 1 shows the results. In these examples, the relatively large maximal error was caused by the difficulty in accurately reconstructing the tip of the cone and the pyramid.

Improvements may be made by precisely calibrating the camera and lighting system, adding more light sources, and obtaining a silhouette from the side camera to eliminate ambiguity about the top of the surface. However, the system meets its goal of providing virtual presences for physical objects in a timely manner that encourages spontaneous interactions.

8.3 User Experience

To evaluate the current usability of the system, we performed a small user study with the goal of determining the relative efficiency and accuracy of the object tracking capability. We designed a task that required users to drag virtual balls of various sizes to specified locations on the table's surface with the help of physical "cursor" objects. The system recorded the time required to complete the task of correctly moving four such balls.

Although the number of participants was too small to yield significant quantitative results, we discovered several common problems users had with the interface. The main difficulties arose from selecting smaller balls, both because of an imprecise "hot spot" for physical interactors, and because the physical object occluded its virtual representation. By designing a context-sensitive "crosshair" cursor that extended beyond the dimensions of the physical object, we were able to significantly increase performance in those cases. In the future, we plan to conduct a more thorough user study, with more participants, that also measures the usability of the gesture tracking subsystem.

9 Putting It to Use: Spontaneous Gesture Interfaces

All the components of the Perceptive Workbench – deictic gesture tracking, object recognition, tracking, and reconstruction – can be seamlessly integrated into a single, consistent framework. The Perceptive Workbench interface detects how users want to interact with it and automatically switches to the desired mode.

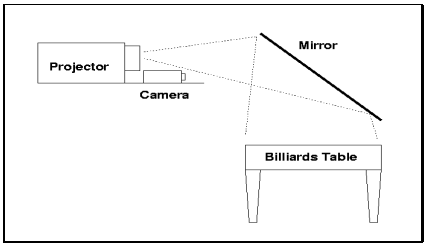
When users move a hand above the display surface, the system tracks the hand and arm as described in Section 6. A cursor appears at the projected hand position on the display surface, and a ray emanates along the projected arm axis. These can be used in selection or manipulation, as in Figure 8a. When users place an object on the surface, the cameras recognize this and identify and track the object. A virtual button also appears on the display (indicated by the arrow in Figure 8b). By tracking the reflections of objects near the table surface, the system determines when the hand overlaps the button, thus selecting it. This action causes the system to capture the 3D object shape, as described in Section 7.

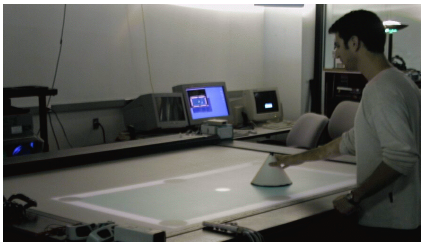
Since shadows from the user's arms always touch the image border, it is easy to decide whether an object lies on the desk surface. If the system detects a shadow that does not touch any border, it can be sure that an object on the desk surface was the cause. As a result, the system will switch to object-recognition and tracking mode. Similarly, the absence of such shadows, for a certain period, indicates that the object has been taken away, and the system can safely switch back to gesture-tracking mode. Note that once the system is in object-recognition mode, it turns off the ceiling lights, and activates the light sources underneath the table. Therefore users can safely grab and move objects on the desk surface, since their arms will not cast any shadows that could disturb the perceived object contours.

These interaction modes provide the elements of a perceptual interface that operates without wires and without restrictions on the objects. For example, we constructed a simple application where the system detects objects placed on the desk, reconstructs them, and then places them in a template set where they are displayed as slowly rotating objects on the workbench display's left border. Users can grab these objects, which can act as new icons that the user can attach to selection or manipulation modes or use as primitives in a model building application.

9.1 An Augmented Billiards Game

We have developed a collaborative interface that combines the Perceptive Workbench with a physical game of pool in a two-player telepresence game. The objective of the game is for the player at the billiards table to sink all of the balls while avoiding a virtual obstacle controlled by the other player at the workbench. A previous system [12] concentrated on suggesting shots for the player using a head-up display and camera as opposed to the projected display used here.





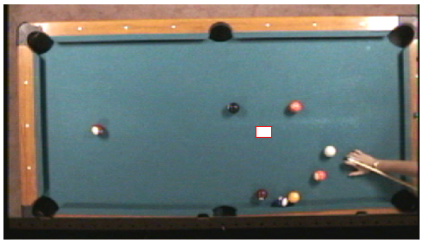


Fig. 9 *(a) The Augmented Billiards Table; (b) Workbench player placing an obstacle; (c) Virtual obstacle overlaid on the real pool table.*

The billiard table is augmented with a setup resembling the Perceptive Workbench apparatus (see Figure 9a). A camera positioned above the table tracks the type and position of the pool balls, while a projector in a similar location can create visual feedback directly on the playing surface. The billiard table's current state is transmitted to the workbench client and rendered as a 3D model. As the game progresses, the workbench updates this model continuously using streaming data from the billiards client.

During the workbench player's turn, he places a physical object on the surface of the workbench (Figure 9b). The workbench derives a 2D representation from the outline of the object and transmits the shape to the billiards client. The outline is projected onto the surface of the billiards table and acts as a virtual obstacle (Figure 9c). If, while the billiards player tries to make his shot any of the balls pass through the obstacle, the workbench player is awarded a point. If the pool player can successfully sink a ball without this happening, he is awarded a point. The workbench player is completely free to choose any object as an obstacle as long as it fits certain size constraints. Thus, the Perceptive Workbench's ability to use previously unknown physical objects enhances the users' possibilities for gameplay. In addition, this "tangible" interface is apparent to a novice user as it involves manipulating a

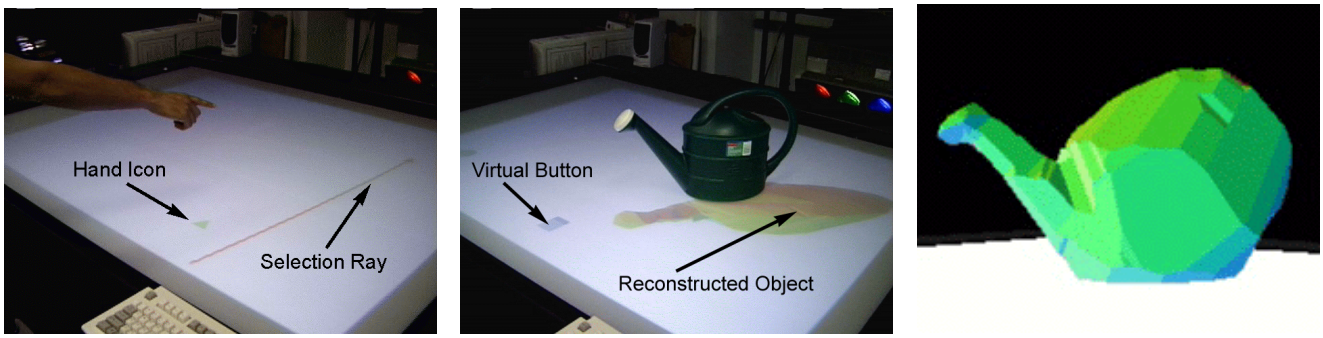


Fig. 8 *(a) Pointing gesture with hand icon and selection ray; (b) Virtual button rendered on the screen when object is detected on the surface; (c) Reconstruction of this watering can.*

physical object as a representation of the virtual obstruction on a display similar in size to the billiards table itself.

9.2 An Augmented Reality Game

We created a more elaborate collaborative interface using the Perceptive Workbench in an augmented reality game. Two or more game masters can communicate with a person in a separate space wearing an augmented reality headset (Figure 10a). The workbench display surface acts as a top-down view of the player's space. The game masters place different objects which appear to the player as distinct monsters at different vertical levels in the game space. While the game masters move the objects around the display surface, this motion is replicated by monsters in the player's view, which move in their individual planes. The player's goal is to dispel these monsters by performing Kung Fu gestures before they can reach him. Since it is difficult for the game master to keep pace with the player, two or more game masters may participate (Figure 10a). The Perceptive Workbench's object tracker scales naturally to handle multiple, simultaneous users. For a more detailed description of this application, see Starner et al. [30, 19].

9.3 3D Terrain Navigation

In another application, we use the Perceptive Workbench's deictic gesture tracking capability to interface with VGIS, a global terrain navigation system that allows continuous flight from outer space to terrain at 1 foot or better resolution. Main interactions include zooming, panning, and rotating the map. Previously, interaction took place by using button sticks with 6-DOF electromagnetic trackers attached.

We employed deictic gesture tracking to remove this constraint. Users choose the direction of navigation by pointing and can change the direction continuously (Figure 10b). Moving the hand toward the display increases the speed toward the earth and moving it away increases the speed away from the earth. Panning and rotating can be accomplished by making lateral gestures in the direction to be panned or by making a rotational arm gesture. Currently, users choose these three modes by keys on a keyboard attached to the workbench, while the extent of the action is determined by deictic tracking. In the future, this selection could be made by analyzing the user's hand shape, or by reacting to spoken commands. In a recent paper, Krum *et al* propose such a navigation interface that implements a combination of speech and user-centered gestures, recognized by a small camera module worn on the user's chest [16].

9.4 Telepresence and CSCW

Last but not least, we built a simple telepresence system. Using the sample interaction framework described at the beginning of this section, users can point to any location on the desk, reconstruct objects, and move them across the desk surface. All of their actions are immediately applied to a VR model of the workbench mirroring the current state of the real desk (Figure 10c). Thus, when performing deictic gestures, the current hand and pointing position appear on the model workbench as a red selection ray. Similarly, the reconstructed shapes of objects on the desk surface are displayed at the corresponding positions in the model. This makes it possible for coworkers at a distant location to follow the user's actions in real-time, while having complete freedom to choose a favorable viewpoint.

10 Integration and Interface Design Issues

The Perceptive Workbench was designed with application integration in mind. Applications implement a simple interface protocol and can take advantage of those parts of the workbench functionality they need. However, for successful application integration, several issues have to be adressed.

From an interface design standpoint, limitations are imposed by both the physical attributes, the hardware restrictions, and the software capabilities of the workbench. While the workbench size permits the display of a life-size model of, for example, the billiards table, the user's comfortable reaching range limits the useful model size. In addition, interface

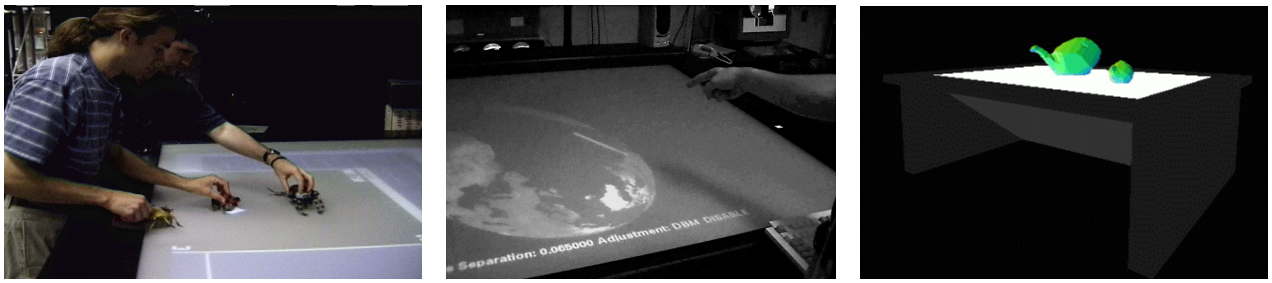


Fig. 10 *Applications: (a) Two game masters controlling virtual monsters; (b) Terrain navigation using deictic gestures; (c) A virtual instantiation of the workbench.*

design is restricted in that one side of the workbench is inaccessible due to the placement of the projector and camera. If gesture tracking is to be used, the available range is even more limited to just one side of the workbench.

The sensing hardware places restrictions on the potential use of tracking information for a user interface. Precise positioning of objects and pointing gestures is limited by the camera resolution. If the application requires watching the whole workbench surface, our current camera resolution of 320*240 pixels limits single-pixel accuracy to about 5 mm. By interpolating the contours with polygons and thus averaging over several samples, we can however arrive at a much higher precision. In a related issue, the contour of a moving object on the workbench is not necessarily stable over time, especially when the motion is so fast that the camera image is blurred. To deal with this, the billiard system detects when an object first comes to rest, determines the object's contour, and simply translates it instead of trying to recompute it. In both cases, the error can be reduced by increasing the resolution or switching to a more expensive progressive scan camera.

On the software side, the question is how to use the information the Perceptive Workbench provides to create a compelling user interface. For example, there are two conceivable types of gestural interactions. The first uses deictic gestures for relative control, for example for directing a cursor, or for adjusting the speed of movement. The other detects gestures that cause a discrete event, like pushing a virtual button to start the reconstructing process, or assuming a specific hand shape to switch between interaction modes. Which of these interaction types is appropriate, and which hand shapes make sense depends largely on the application.

Another question is how much information to transmit from the workbench to its clients. If too much information about static objects is transmitted to the display client, the time needed to read out and process the corresponding network messages can reduce the effective display frame rate. In our applications, we found it useful to only transmit updates on objects whose positions had changed.

On the client side, sensor intergration needs to be addressed. For gesture tracking, information from two cameras is integrated. If the application requires lower latencies than those currently provided, Kalman filtering may be used. Since the cameras are not explicitly synchronized, asynchronous filters, like the single-constraint-at-a-time method by Welch and Bishop [42], may also prove useful.

11 Maintenance of Perceptual Interfaces

One detriment to perceptual interfaces is that the underlying sensor platform needs to be maintained. Video cameras may be bumped, lights may burn out, or the entire structure may need to be moved. The Perceptual Workbench has served as a experimental platform for several years and has undergone several major revisions. In addition, the underlying Fakespace hardware is often used for virtual environment demonstrations. Such heavy use stresses an experimental system. Thus, the system must be self-calibrating wherever possible.

Object identification and tracking on the the surface of the desk is one of the most valued services for the Perceptive Workbench. Fortunately, it is also the most easy to maintain. This service requires only one camera and the infrared lights under the desk. This system is easy to install and realign when necessary. In addition, the computer vision software automatically adjusts to the lighting levels available each time the system is initialized, making the system relatively robust to changes that occur on a day-to-day basis.

The Perceptive Workbench's gesture tracking software is also used extensively. While the infrared light above the table is relatively protected in everyday use, the side-view camera is not. If a user bumps the side camera out of position, its calibration procedure must be redone. Fortunately, this procedure is not difficult. Embedding the camera into a wall near the side of the workbench may reduce this problem.

Three dimensional reconstruction on the Perceptive Workbench requires the positions of the overhead lights to be known to within centimeters. The position of each light constrains the positions of the other lights due to the limited surface of the desk on which a reconstruction subject can be placed and still cast a shadow that does not interest the desk's edge. In addition, reconstruction requires the most pieces of apparatus and the most careful alignment. Thus, reconstruction proves the biggest challenge to physically moving the Perceptive Workbench. Fortunately, the Perceptive Workbench stays in one place for extended periods of time, and the overhead

lights are out of the way of most experiments and other apparatus. However, the overhead lights do burn out with time must be replaced.

12 Future Work

Many VR systems use head-tracked shutter glasses and stereoscopic images to get a more immersive effect. In order to make these systems fully wireless, we need to apply vision-based methods to also track the user's head. At present, we are researching inexpensive and robust ways to do this that still meet the performance criteria. Results from Ware and Balakrishnan [39] suggest that, in contrast to fully immersive systems where users wear a head-mounted display and relatively small head rotations can cause large viewpoint shifts, semi-immersive systems do not impose such high restrictions on head-movement latency. In fact, since the head position is much more important than the head orientation in these systems, latency can even be slightly larger than with the gesture and object tracking.

In addition, we will work on improving the latency of the gesture-rendering loop through code refinement and the application of Kalman filters. For the recognition of objects on the desk's surface, we will explore the use of statistical methods that can give us better ways of handling uncertainties and distinguishing new objects. We will also employ hidden Markov models to recognize symbolic hand gestures [31] for controlling the interface. Finally, as hinted by the multiple game masters in the gaming application, several users may be supported through careful, active allocation of resources.

13 Conclusion

The Perceptive Workbench uses a vision-based system to enable a rich set of interactions, including hand and arm gestures, object recognition and tracking, and 3D reconstruction of objects placed on its surface. Latency measurements show that the Perceptive Workbench's tracking capabilities are suitable for real-time interaction.

All elements combine seamlessly into the same interface and can be used in various applications. In addition, the sensing system is relatively inexpensive, using standard cameras and lighting equipment plus a computer with one or two video digitizers, depending on the functions desired. As seen from the multiplayer gaming, terrain navigation, and telepresence applications, the Perceptive Workbench encourages an untethered and spontaneous interface that encourages the inclusion of physical objects in the virtual environment.

Acknowledgements

This work is supported in part by funding from Georgia Institute of Technology's Broadband Institute. We thank Brad Singletary, William Ribarsky, Zachary Wartell, David Krum, and Larry Hodges for their help building the Perceptive Workbench and interfacing it with the applications mentioned above. In addition we thank Paul Rosin and Geoff West for their line segmentation code, the Purdue CADLab for TWIN, and Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno for Metro.

References

1. O. Bimber. Gesture controlled object interaction: A virtual table case study. In *7th Int'l Conf. in Central Europe on Computer Graphics, Visualization, and Interactive Digital Media (WSCG'99)*, volume 1, Plzen, Czech Republic, 1999.
2. A. Bobick, S. Intille, J. Davis, F. Baird, C. Pinhanez, L. Campbell, Y. Ivanov, and A. Wilson. The kidsroom: A perceptually-based interactive and immersive story environment. *PRESENCE: Teleoperators and Virtual Environments*, 8(4):367–391, August 1999.
3. R. Bolt and E. Herranz. Two-handed gesture in multi-modal natural dialogue. In *ACM Symposium on User Interface Software and Technology (UIST'92)*, pages 7–14, 1992.
4. P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, June 1998.
5. D. Daum and G. Dudek. On 3-d surface reconstruction using shape from shadows. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pages 461–468, 1998.
6. J.W. Davis and A.F. Bobick. Sideshow: A silhouette-based interactive dual-screen environment. Technical Report TR-457, MIT Media Lab Tech Report, 1998.
7. Computer Aided Design and Graphics Laboratory (CADLAB). *TWIN Solid Modeling Package Reference Manual*. School of Mechanical Engineering, Purdue University, http://cadlab.www.ecn.purdue.edu/cadlab/twin, 1995.
8. K. Dorfmueller-Ulhaas and D. Schmalstieg. Finger tracking for interaction in augmented environments. In *Proceedings of the 2nd ACM/IEEE International Symposium on Augmented Reality (ISAR'01)*, 2001.
9. G.W. Fitzmaurice, H. Ishii, and W. Buxton. Bricks: Laying the foundations for graspable user interfaces. In *Proceedings of CHI'95*, pages 442–449, 1995.
10. M. Fukumoto, K. Mase, and Y. Suenaga. Real-time detection of pointing actions for a glove-free interface. In *Proceedings of IAPR Workshop on Machine Vision Applications*, Tokyo, Japan, 1992.
11. H. Ishii and B. Ullmer. Tangible bits: Towards seamless interfaces between people, bits, and atoms. In *Proceedings of CHI'97*, pages 234–241, 1997.
12. T. Jebara, C. Eyster, J. Weaver, T. Starner, and A. Pentland. Stochasticks: Augmenting the billiards experience with probabilistic vision and wearable computers. In *Proceedings of the First Intl. Symposium on Wearable Computers*, Cambridge, MA, 1997.
13. C. Jennings. Robust finger tracking with multiple cameras. In *Proc. of the International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 152–160, 1999.
14. M. Krueger. *Artificial Reality II*. Addison-Wesley, 1991.
15. W. Krueger, C.-A. Bohn, B. Froehlich, H. Schueth, W. Strauss, and G. Wesche. The responsive workbench: A virtual work environment. *IEEE Computer*, 28(7):42–48, July 1995.

16. D.M. Krum, O. Ometoso, W. Ribarsky, T. Starner, and L. Hodges. Speech and gesture multimodal control of a whole earth 3d virtual environment. In *submitted to IEEE Virtual Reality 2002 Conference*, 2002.
17. A. Laurentini. How many 2d silhouettes does it take to reconstruct a 3d object? *Computer Vision and Image Understanding (CVIU)*, 67(1):81–87, July 1997.
18. B. Leibe, D. Minnen, J. Weeks, and T. Starner. Integration of wireless gesture tracking, object tracking, and 3d reconstruction in the perceptive workbench. In *Proceedings of 2nd International Workshop on Computer Vision Systems (ICVS 2001)*, volume 2095 of *Lecture Notes in Computer Science*, pages 73–92. Springer, Berlin, July 2001.
19. B. Leibe, T. Starner, W. Ribarsky, Z. Wartell, D. Krum, B. Singletary, and L. Hodges. Toward spontaneous interaction with the perceptive workbench. *IEEE Computer Graphics & Applications*, 20(6):54–65, Nov. 2000.
20. W. Matusik, C. Buehler, S. Gortler, R. Raskar, and L. McMillan. Image based visual hulls. In *Proceedings of SIGGRAPH 2000*, 2000.
21. F.K.H. Quek. Eyes in the interface. *Image and Vision Computing*, 13(6):511–525, Aug. 1995.
22. J.M. Rehg and T. Kanade. Visual tracking of high dof articulated structures: an application to human hand tracking. In *Third European Conference on Computer Vision (ECCV'94)*, pages 35–46, 1994.
23. J. Rekimoto and N. Matsushita. Perceptual surfaces: Towards a human and object sensitive interactive display. In *Workshop on Perceptual User Interfaces (PUI'97)*, 1997.
24. P.L. Rosin and G.A.W. West. Non-parametric segmentation of curves into various representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1140–1153, 1995.
25. Y. Sato, Y. Kobayashi, and H. Koike. Fast tracking of hands and fingertips in infrared images for augmented desk interface. In *Proc. of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 462–467, 2000.
26. A.F. Seay, D. Krum, W. Ribarsky, and L. Hodges. Multimodal interaction techniques for the virtual workbench. In *Proceedings CHI'99 Extended Abstracts*, pages 282–283, 1999.
27. J. Segen and S. Kumar. Shadow gestures: 3d hand pose estimation using a single camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99)*, volume 1, pages 479–485, 1999.
28. R. Sharma and J. Molineros. Computer vision based augmented reality for guiding manual assembly. *PRESENCE: Teleoperators and Virtual Environments*, 6(3):292–317, 1997.
29. S.K. Srivastava and N. Ahuja. An algorithm for generating octrees from object silhouettes in perspective views. *Computer Vision, Graphics, and Image Processing: Image Understanding (CVGIP:IU)*, 49(1):68–84, 1990.
30. T. Starner, B. Leibe, B. Singletary, and J. Pair. Mind-warping: Towards creating a compelling collaborative augmented reality gaming interface through wearable computers and multi-modal input and output. In *IEEE International Conference on Intelligent User Interfaces (IUI'2000)*, 2000.
31. T. Starner, J. Weaver, and A. Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.
32. D. Sturman. *Whole-hand Input*. PhD thesis, MIT Media Lab, 1992.
33. S. Sullivan and J. Ponce. Automatic model construction, pose estimation, and object recognition from photographs using triangular splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1091–1097, 1998.
34. B. Ullmer and H. Ishii. The metadesk: Models and prototypes for tangible user interfaces. In *ACM Symposium on User Interface Software and Technology (UIST'97)*, pages 223–232, 1997.
35. J. Underkoffler and H. Ishii. Illuminating light: An optical design tool with a luminous-tangible interface. In *Proceedings of CHI'98*, pages 542–549, 1998.
36. A. Utsumi and J. Ohya. Multiple-hand-gesture tracking using multiple cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99)*, volume 1, pages 473–478, 1999.
37. R. van de Pol, W. Ribarsky, L. Hodges, and F. Post. Interaction in semi-immersive large display environments. In *Proceedings of Virtual Environments'99*, pages 157–168, 1999.
38. C. Vogler and D. Metaxas. Asl recognition based on coupling between hmms and 3d motion analysis. In *Sixth International Conference on Computer Vision (ICCV'98)*, pages 363–369, 1998.
39. C. Ware and R. Balakrishnan. Reaching for objects in vr displays: Lag and frame rate. *ACM Transactions on Computer-Human Interaction*, 1(4):331–356, 1994.
40. Z. Wartell, W. Ribarsky, and L.F. Hodges. Third-person navigation of whole-planet terrain in a head-tracked stereoscopic environment. In *IEEE Virtual Reality '99 Conference*, pages 141–149, 1999.
41. B. Watson, N. Walker, W. Ribarsky, and V. Spaulding. The effects of variation of system responsiveness on user performance in virtual environments. *Human Factors*, 40(3):403–414, 1998.
42. G. Welch and G. Bishop. Scaat: Incremental tracking with incomplete information. In *Conference Proceedings, Annual Conference Series, 1997, ACM SIGGRAPH*, Aug. 1997.
43. P. Wellner. Interacting with paper on the digital desk. *Communications of the ACM*, 36(7):86–89, 1993.
44. C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.