Toward Spontaneous Interaction with the Perceptive Workbench

Bastian Leibe, Thad Starner, William Ribarsky, Zachary Wartell, David Krum, Justin Weeks, Bradley Singletary, and Larry Hodges
Georgia Institute of Technology

The Perceptive Workbench provides a spontaneous and unimpeded interface between the physical and virtual worlds. Its vision-based methods for interaction eliminate the need for wired input devices and wired tracking.

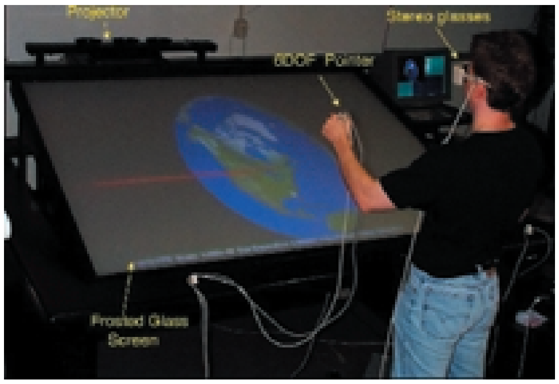
Until now, we've interacted with computers mostly by using wire-based devices. Typically, the wires limit the distance of movement and inhibit freedom of orientation. In addition, most interactions are indirect. The user moves a device as an analog for the action created in the display space. We envision an untethered interface that accepts direct, natural gestures and can spontaneously accept as interactors any objects we choose. In conventional 3D interaction, the devices that track position and orientation are still usually tethered to the machine by wires.

Devices, such as pinch gloves, that permit the user to experience a more natural interface often don't perform as well. Users generally prefer simple handheld devices with buttons.[1] Pinch gloves carry assumptions about the position of the user's hand and fingers with respect to the tracker. Of course, users' hands differ in size and shape, so the assumed tracker position must be recalibrated for each user. This is hardly ever done. Also, the glove interface causes subtle changes to recognized hand gestures. The result is that fine manipulations can be imprecise, and the user comes away with the feeling that the interaction is slightly off in an indeterminate way. If we can recognize gestures directly, we take into account the difference in hand sizes and shapes.

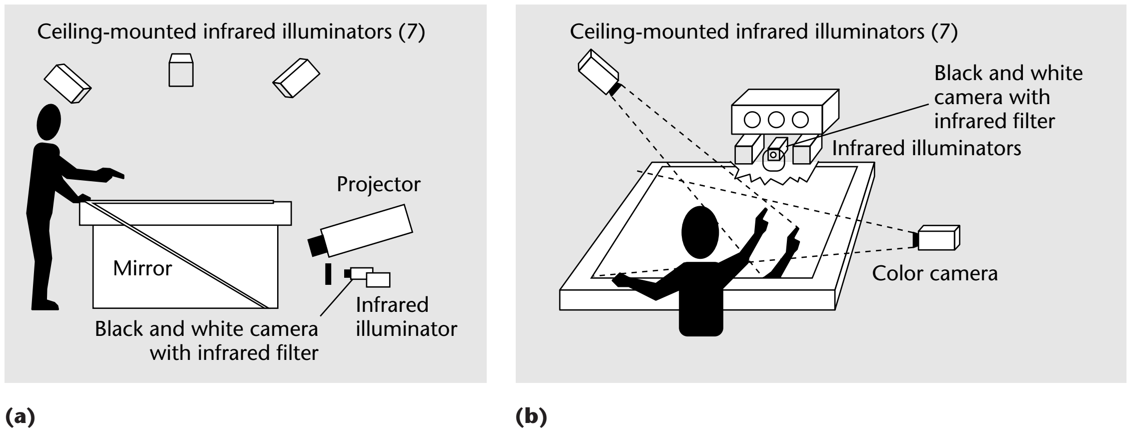
An additional problem is that any device held in the hand can become awkward while gesturing. We've found this even with a simple pointing device, such as a stick with a few buttons[1] (see Figure 1). Also, unless fairly skilled, a user often has to pause to identify and select buttons on the stick. With accurately tracked hands, most of this awkwardness disappears. We're adept at pointing in almost any direction and can quickly pinch fingers, for example, without looking at them.

Finally, physical objects are often natural interactors (such as phicons[2]). However, with current systems these objects must be inserted in advance or specially prepared. We'd like the system to accept objects that we choose spontaneously for interaction.

In this article we discuss methods for producing more seamless interaction between the physical and virtual environments through the Perceptive Workbench. We applied the system to an augmented reality game and a terrain navigating system. The Perceptive Workbench can reconstruct 3D virtual representations of previously unseen real-world objects placed on its surface. In addition, the Perceptive Workbench identifies and tracks such objects as they're manipulated on the desk's surface and allows the user to interact with the augmented environment through 2D and 3D gestures. These gestures can be made on the plane of the desk's surface or in the 3D space above the desk. Taking its cue from the user's actions, the Perceptive Workbench switches between these modes automatically. Computer vision controls all interaction, freeing the user from the wires of traditional sensing techniques.



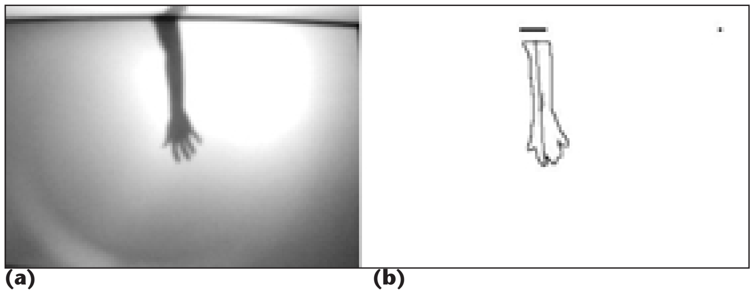
1 A user interacting with Georgia Tech's Virtual Workbench using a 6 degrees-of-freedom (DOF) pointer.


Ceiling-mounted infrared illuminators (7)
Projector
Mirror
Infrared illuminator
Black and white camera with infrared filter
(a)
Ceiling-mounted infrared illuminators (7)
Black and white camera with infrared filter
Infrared illuminators
Color camera
(b)

2 **(a) Light and camera positions for the Perceptive Workbench. (b) The top view shows how shadows are cast and the 3D arm position is tracked.**

Related work

While the Perceptive Workbench is unique in its ability to interact with the physical world, it has a rich heritage of related work.[1-7] Many augmented desk and virtual reality designs use tethered props, tracked by electromechanical or ultrasonic means, to encourage interaction through manipulation and gesture.[1,3,7-9] Such designs tether the user to the desk and require the time-consuming ritual of donning and doffing the appropriate equipment.



3 **(a) Arm shadow from overhead infrared lights. (b) Resulting contour with recovered arm direction.**

Fortunately, the computer vision community has taken up the task of tracking the user's hands and identifying gestures. While generalized vision systems track the body in room- and desk-based scenarios for games, interactive art, and augmented environments,[10] reconstruction of fine hand detail involves carefully calibrated systems and is computationally intensive.[11] Even so, complicated gestures such as those used in sign language[12] or the manipulation of physical objects[13] can be recognized. The Perceptive Workbench uses computer vision techniques to maintain a wireless interface.

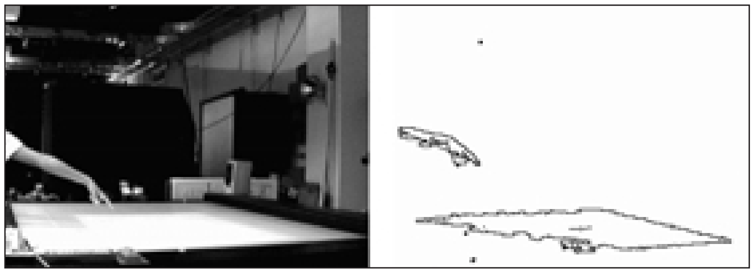
Most directly related to the Perceptive Workbench, Ullmer and Ishii's Metadesk[2] identifies and tracks objects placed on the desk's display surface using a near-infrared computer vision recognizer, originally designed by Starner. Unfortunately, since not all objects reflect infrared light and infrared shadows aren't 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[6] 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[14] resembles 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 and are designed to augment the process of handling paper or interacting with models and widgets through fiducials or barcodes.[15,16] Krueger's VideoDesk,[4] an early desk-based system, used an overhead camera and a horizontal visible light table (for high contrast) to provide hand-gesture input for interactions displayed on a monitor on the desk's far side. In contrast to 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.

Apparatus

The display environment for the Perceptive Workbench builds on Fakespace's Immersive Workbench. It consists of a wooden desk with a horizontal frosted glass surface on which a stereoscopic image can be projected from behind the workbench (see Figure 1).

We placed a standard monochrome surveillance camera under the projector that watches the desk surface from underneath (see Figure 2). 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 reflected 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 2). Each computer-controlled light casts distinct shadows on the desk's surface based on the objects on the table (Figure 3a). A second camera, this one in color, is placed next



4 (a) Image from side camera. (b) Arm contour (from similar image) with recovered arm direction.

to the desk to provide a side view of the user's arms (Figure 4a). We use this side camera solely for recovering 3D pointing gestures (Figure 5).

We decided to use near-infrared light, since it's invisible to the human eye. Thus, illuminating the scene with it doesn't interfere with the user's interaction. The user can't observe 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. This makes it 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.

All vision processing occurs on two SGI R10000 O2s (one for each camera), which communicate with a display client on an SGI Onyx via sockets. However, the vision algorithms can also be run on one SGI with two digitizer boards or implemented using inexpensive, semicustom signal-processing hardware.

We use this setup for three different kinds of interaction (which we'll explain in more detail in the following sections):

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

For display on the Perceptive Workbench, we use the Simple Virtual Environment (SVE) Toolkit, a graphics and sound library the Georgia Tech Virtual Environments Group developed.[17] SVE lets us rapidly prototype applications used in this work. In addition, we use the workbench version of the Virtual Geographic Information System (VGIS), a global terrain visualization and navigation system[18] as an application for interaction using hand and arm gestures. The workbench version of VGIS has stereoscopic rendering and an intuitive interface for navigation.[19] Both systems are built on OpenGL and have SGI and PC implementations.

Object tracking and recognition

As a basic building block 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 freely pick any set of physical objects.

The motivation behind this is to use physical objects in a graspable user interface.[20] Physical objects are often natural interactors. They can provide physical handles to let users control the virtual application intuitively.[21] In addition, the use of objects encourages two-handed direct manipulation and allows parallel input specification, thereby improving the communication bandwidth with the computer.[20,21]

To achieve this goal, we use an improved version of the technique described in Starner et al.[22] Two near-infrared light-sources illuminate the desk's underside (Figure 2). 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 set of features, including area, eccentricity, perimeter, moments, and contour shape.

However, 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 whole desk surface—bright in the middle and weaker toward the borders. In addition, the light rays aren't parallel, and the reflection on the mirror surface further exacerbates this effect. As a result, the perceived sizes and shapes of objects on the desk surface can vary depending on position and orientation. Finally, when users move an object, the reflection from their hands can also add to the perceived shape. This requires an additional stage in the recognition process: matching recognized objects to objects known to lie on the table and filtering out wrong classifications or even handling complete loss of information about an object for several frames.

In this work, we use the object recognition and tracking capability mainly for cursor 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.[2] A future project will explore different modes of interaction based on this technology.

Deictic gesture tracking

Hand gestures can be roughly classified into *symbolic* (iconic, metaphoric, and beat) and *deictic* (pointing) gestures. Symbolic gestures carry an abstract meaning that may still be recognizable in iconic form in the associated hand movement. Without the necessary cultural context, however, the meaning may be arbitrary. Examples for symbolic gestures include most conversational gestures in everyday use, and whole gesture languages, such as American Sign Language. Previous work by Starner[12] showed that a large set of symbolic gestures can be distinguished and recognized from live video images using hidden Markov models.

Deictic gestures, on the other hand, depend strongly on location and orientation of the performing hand. Their meaning is determined by the location at which a finger points 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.

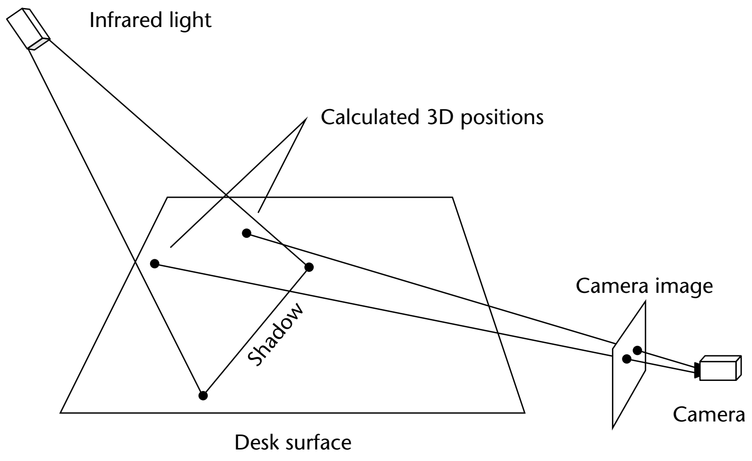
For navigation and object manipulation in a virtual environment, many gestures will have a deictic component. It's usually not enough to recognize that an object should be rotated—we'll also need to know the desired amount of rotation. For object selection or translation, we want to specify the object 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 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.[3,8] 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.

With vision-based 3D tracking techniques, the first issue is to determine which information in the camera image is relevant—that is, which regions represent the user's hand or arm. What makes this even more difficult is the variation in user clothing or skin color and background activity. Although typically only one user interacts with the environment at a given time using traditional methods of interaction, the physical dimensions of large semi-immersive environments such as the workbench invite people to watch and participate.

In a virtual workbench environment, a camera can only be placed in a few places to provide reliable hand position information. One camera can be set up next to the table without overly restricting the available space for users, but if a similar second camera were to be used at this location, either multiuser experience or accuracy would be compromised. We addressed this problem by employing our shadow-based architecture (as described in the "Apparatus" section). The user stands in front of the workbench and extends an arm over the surface. One of the infrared light sources mounted on the ceiling to the left of, and slightly behind the user, shines its light on the desk surface, from where it can be seen by the infrared camera under the projector (see Figure 5). When users move an arm over the desk, it casts a shadow on the desk surface (see Figure 3a). 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 4a and 5) 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, it 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 for the hand position and a selection ray in the workbench display.



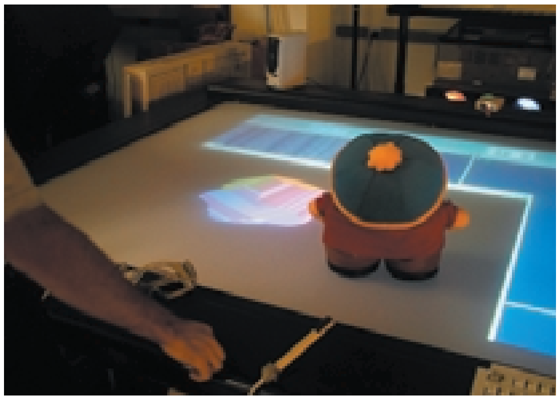
5 Principle of pointing direction recovery.

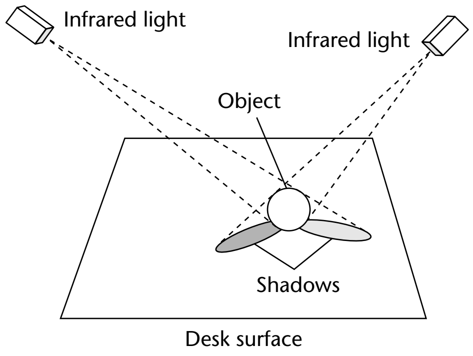
Obviously, the success of the gesture-tracking capability relies strongly on how fast the image processing can be done. Therefore, we must use simple algorithms. Fortunately, we can make some simplifying assumptions about the image content.

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 3b and 4b). For simplicity, we call this point the "shoulder," although in most cases it is not. Tracing the shadow's contour, the algorithm searches for the point farthest away from the shoulder and takes it 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 doesn't depend on a pointing gesture, but also works for most other hand shapes, including but not restricted to, a hand held horizontally, vertically, or in a fist. These shapes may be 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 isn't overly bent (see Figure 4). In such cases, the algorithm still connects the shoulder and fingertip,


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



7 Principle of the 3D reconstruction.

resulting in a direction somewhere between the direction of the arm and the one given by the hand. Although the absolute resulting pointing position doesn't match the position towards which the finger is pointing, it still manages to capture the movement's trend 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 arm movement.

Limitations

The architecture we used poses several limitations. The primary problem with the shadow approach is finding a position for the light source that produces a good shadow of the user's arm for a large set of possible positions while avoiding capturing the shadow from the user's body. Since the area visible to the infrared camera is coincident with the desk surface, in some regions the shadow isn't visible, or it touches the borders or falls outside of them. To solve this problem, we switch automatically to a different light source whenever such a situation occurs. The choice of the new light source depends on where the shadows touched the border. By choosing overlapping regions for all light sources, we can keep the number of light-source switches to a necessary minimum. In practice, four light sources in the original set of seven covered the relevant area of the desk surface. However, we added an additional spotlight, mounted directly over the desktop, to provide more direct coverage of the desktop surface.

Figure 4b shows another problem where segmentation based on color background subtraction detects both the hand and the change in the display on the workbench. A more recent implementation 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 or any other close objects, the user's arm is illuminated and made distinct from the background. Changes in the workbench's display don't affect the tracking.

A bigger 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 we chose at this first stage to accept this restriction and concentrate on high frame-rate support for one-handed interaction. 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 doesn't model effects such as gravity and inertia.[1]

3D reconstruction

To complement the capabilities of the Perceptive Workbench, we want to insert real objects into the virtual world and share them with other users at different locations. An example application for this could be a telepresence or computer-supported collaborative work (CSCW) system. Instead of verbally describing an object, users would be able to quickly create a 3D reconstruction and send it to their coworkers (see Figure 6). This requires designing a reconstruction mechanism that doesn't interrupt the interaction. Our focus is to provide an almost instantaneous visual cue for the object as part of the interaction, not necessarily on creating a highly accurate model.

Several methods reconstruct objects from silhouettes[23,24] or dynamic shadows[25] using either a moving camera or light source on a known trajectory or a turntable for the object.[24] 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 (sometimes in a different room). 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 the infrared light sources, we analyze the shadows cast on the object from multiple directions (see Figure 7). Since the process is based on infrared light, it can be applied independently of the lighting conditions and without interfering with the user's natural interaction with the desk.

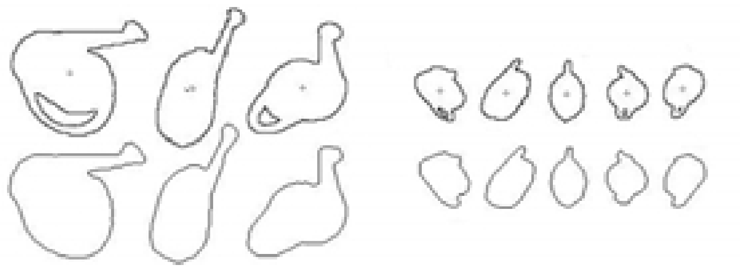
The existing approaches to reconstruct shape from shadows or silhouettes can be divided into two camps. The volume approach, pioneered by Baumgart,[26] intersects view volumes to create a representation for the object. Common representations for the resulting model are polyhedra[26] or octrees.[23] The surface approach reconstructs the surface as the envelope of its tangent planes. Both approaches can be combined, as in Sullivan's work,[24] which uses volume intersection to create an object, then smoothes the surfaces with splines.

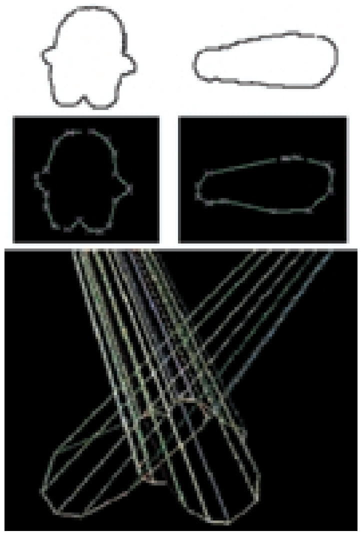
We used a volume approach to create polyhedral reconstructions for several reasons. We want to create models that users employ instantaneously in a virtual environment. Our focus isn't on getting a photorealistic reconstruction, but on creating a quick, low-polygon-count model for an arbitrary real-world object in real time without interrupting the ongoing interaction. Polyhedral models offer significant advantages over other representations, such as generalized cylinders, superquadrics, or polynomial splines. They're simple and computationally inexpensive, Boolean set operations can be performed on them with reasonable effort, and most current VR engines are optimized for fast rendering of polygonal scenes. In addition, polyhedral models form the basis for many later processing steps. If desired, they can still be refined with splines using a surface approach.

To obtain the different views, we mounted a ring of seven infrared light sources in the ceiling, each independently controlled by a computer. The system detects when users place a new object on the desk surface, and they can initiate the reconstruction by touching a virtual button rendered on the screen. The camera detects this action, and, during only one second, captures all shadow images. In another second, the reconstruction is complete, and the newly reconstructed object becomes part of the virtual world.

Our approach is fully automated and doesn't require any special hardware (such as stereo cameras, laser range finders, structured lighting, and so on). The method is extremely inexpensive, both in hardware and in computational cost. In addition, there's no need for extensive calibration, which is usually necessary in other approaches to recover the object's exact position or orientation in relation to the camera. We only need to know the approximate position of the light sources (± 2 cm) and adjust the camera to reflect the display surface's size (which only needs to be done once). We don't move the camera, light sources, or the object during the reconstruction process. Thus, recalibration is unnecessary. We substituted all mechanical moving parts—often prone to wear and imprecision—with a series of light beams from known locations.

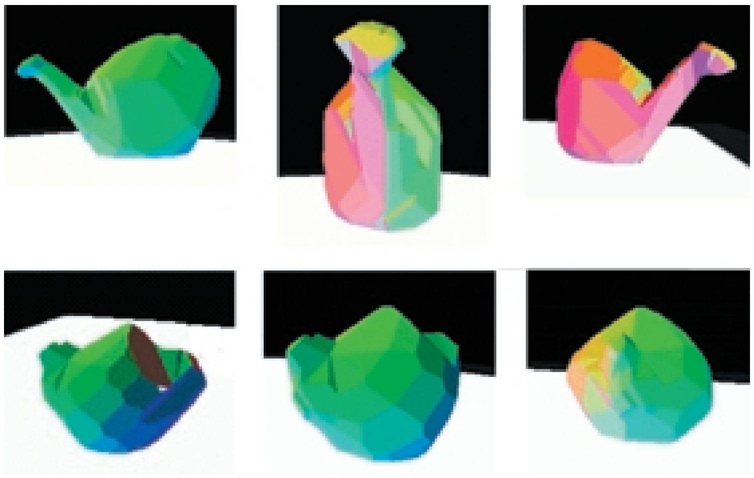
An obvious limitation to this approach is that we're confined to a fixed number of different views from which to reconstruct the object. The turntable approach permits the system to take an arbitrary number of images from different viewpoints. However, Sullivan's work[24] and our experience with our system have shown that even for quite complex objects, usually seven to nine different views suffice to get a reasonable 3D model of the object. Note that the reconstruction uses the same hardware as the deictic gesture-tracking capability discussed in the previous section. Thus, it comes at no additional cost.


8 **Shadow contours (top) and interpolated polygons (bottom) from a watering can (left) and a teapot (right). The current version of our software doesn't handle holes in the contours yet.**


9 **Steps of the 3D reconstruction of the doll from Figure 6, including the extraction of contour shapes from shadows and the intersection of multiple view cones (bottom).**

The switching time of the light sources limits the speed of the reconstruction process. Whenever a new light source is activated, the image-processing system has to wait for several frames to receive a valid image. The camera under the desk records the sequence of shadows cast by an object on the table when illuminated by the different lights. Figure 8 shows two series of contour shadows extracted from two sample objects by using different infrared sources sources. By approximating each shadow as a polygon (not necessarily convex),[27] 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. Figure 9 shows more shadows with the resulting polygons and a visualization of the intersection of polyhedral cones.

Intersecting nonconvex polyhedral objects is a complex problem, further complicated by numerous special cases. Fortunately, this problem has already been exhaustively researched and solutions are available. For the intersection calculations in our application, we used Purdue University's Twin Solid Modeling Library.[28]



10 3D reconstruction of a watering can (top) and a teapot. We chose the colors to highlight the model faces.

In the current version of our software, an additional error is introduced because we aren't yet handling holes in the shadows. This is merely an implementation issue, which will be resolved in a future extension to our project.

Performance analysis

We analyzed the performance of the Perceptive Workbench's object and gesture tracking and the 3D reconstruction of objects.

Object and gesture tracking

Both object and gesture tracking perform at a stable 12 to 18 frames per second (fps). Frame rate depends on the number of objects on the table and the size of the shadows, respectively. Both techniques follow fast motions and complicated trajectories. Latency is currently 0.25 to 0.33 second, including display lag. (An acceptable threshold for 3D object selection and manipulation tasks is typically around 0.075 to 0.100 second.[30]) Surprisingly, this level of latency 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 can provide users with adequate tactile feedback as the system catches up with their manipulations. In general, since users move objects across a very large desk surface, the lag is noticeable but rarely troublesome in the current applications.

Even so, we expect that simple improvements in the socket communication between the vision and rendering code and in the vision code itself will improve latency significantly. For the terrain navigation task below, rendering speed provides a limiting factor. However, Kalman filters may compensate for render lag and will also add to the tracking system's stability.

Figure 10 shows the reconstructed models of a watering can and a teapot placed on the desk surface. We chose the colors to highlight the different model faces by interpreting the face normal as a vector in RGB color space.

Limitations

An obvious limitation to our approach is that 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.[29] Even for objects with a polyhedral visual hull, an unbounded number of silhouettes may be necessary for an exact reconstruction. In practice, we can get sufficiently accurate models for a large variety of real-world objects, even from a relatively small number of different views.

Exceptions 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.

Apart from this, the accuracy with which objects can be reconstructed is bounded by another limitation of our architecture. We mounted all our light sources to the ceiling. From this point of view they can't provide full information about the object's shape. There's a pyramidal blind spot above all horizontal flat surfaces that the reconstruction can't eliminate. The slope of these pyramids depends on the angle between the desk surface and the rays from the light sources. For our current hardware setting, this angle ranges between 37 and 55 degrees, depending on the light source. Only structures with a greater slope will be reconstructed entirely without error. This problem is intrinsic to the method and also occurs with the turntable approach, but on a much smaller scale.

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. Calculations based on the current position of the side camera (optimized for gesture recognition) promise an error angle of just 7 degrees.

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.

To measure error, we used the Metro tool developed by Cignoni, Rocchini, and Scopigno.[31] 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 $(E(S1, S2), E(S2, S1))$ with $E(S1, S2)$ denoting the one-sided distance between the surfaces $S1$ and $S2$:

$$E(S1, S2) = \max_{p \in S1} (\text{dist}(p, S2)) = \max_{p \in S1} (\min_{p' \in S2} (\text{dist}(p, p')))$$

The Hausdorff distance directly corresponds 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 quick and timely manner that encourages spontaneous interactions.

Putting it to use: Spontaneous gesture interfaces

All the components of the Perceptive Workbench—deictic gesture tracking, object recognition, tracking, and reconstruction—can be combined 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 the section "Object tracking and recognition." 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 11a. 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 11b). Through shadow tracking, 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 the section "Deictic gesture tracking."

Since shadows from the user's arms always touch the image border, it's easy to decide whether an object lies on the desk surface. If the system detects a shadow that doesn't touch any border, it can be sure that an object on the desk surface caused the shadow. 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.

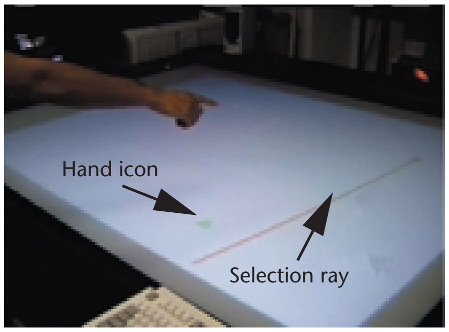
Table 1. Reconstruction errors averaged over three runs (in meters and percentage of object diameter).

	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%)

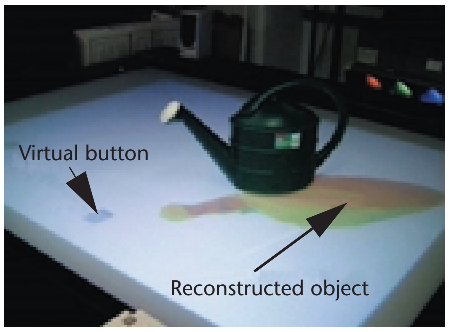
This setup provides the elements of a perceptual interface—operating without wires and without restrictions on the objects. For example, we constructed a simple application where the system detects objects placed on the desk, reconstructs them, then places them in a template set, displayed as slowly rotating objects on the workbench display's left border. Users can grab these objects, which could act as new physical icons that they attach to selection or manipulation modes. Or the shapes themselves could be used in model building or other applications.

An augmented reality game

We created a more elaborate collaborative interface using the Perceptive Workbench. The workbench communicates with a person in a separate space wearing an augmented reality headset. All interaction occurs via image-based gesture tracking without attached sensors. We patterned the game after a martial arts fighting game. The user in the augmented reality headset is the player, and one or more people interacting with the workbench are the game masters. The workbench display surface acts as a top-down view of the player's space. The game masters place different objects on the surface, which appear to the player as distinct monsters at different vertical levels in the game space. The game masters move the objects around the display surface, toward and away from the player. This motion is replicated in the player's view by the monsters, which move

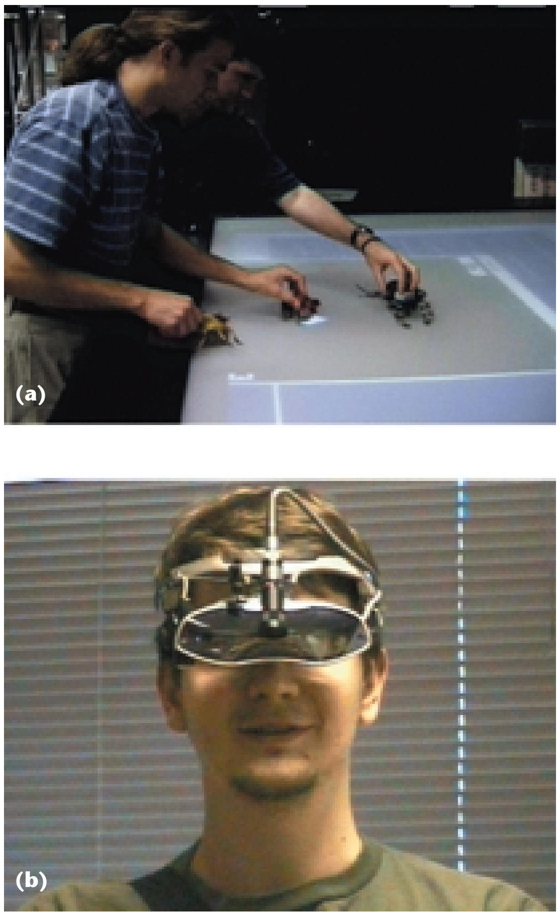


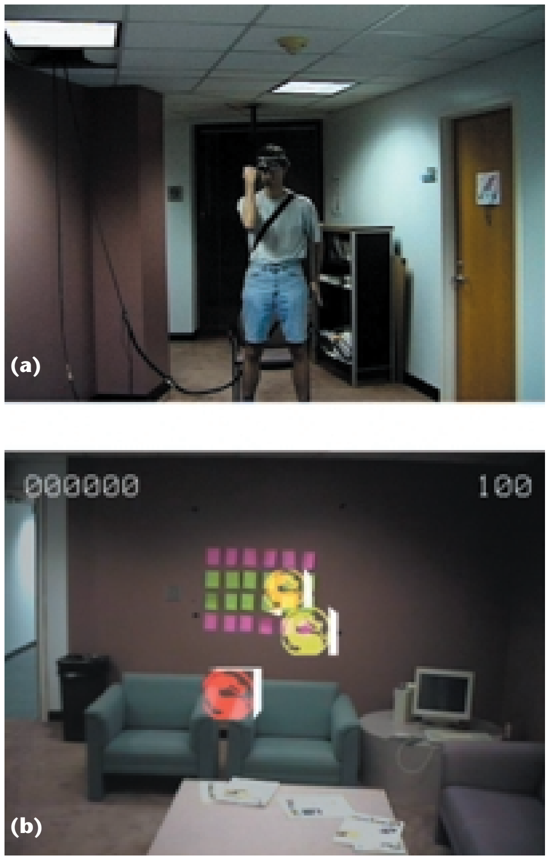
(a)



(b)

11 (a) Pointing gesture with hand icon and selection ray. (b) Virtual button rendered on the screen when object is detected on the surface. Figure 10 shows the reconstruction of this watering can.


12 (a) Two game masters controlling virtual monsters. (b) Hardware outfit worn by mobile player.



13 (a) Mobile player performing Kung-Fu gestures to ward off monsters. (b) Virtual monsters overlaid on the real background as seen by the mobile player.

in their individual planes. Figure 12a shows the game masters moving objects, and Figure 13b displays the moving monsters in the virtual space.

The mobile player wears a "see-through" Sony Glasstron (Figure 12b) equipped with two cameras. The forward-facing camera tracks fiducials or natural features in the player's space to recover head orientation. This allows graphics (such as monsters) to be rendered roughly registered with the physical world. The second camera looks down at the player's hands to recognize martial arts gestures.[12]

While we're developing a more sophisticated hidden Markov model system, a simple template-matching method suffices for recognizing a small set of martial arts gestures. To effect attacks on the monsters, the user accompanies the appropriate attack gesture (Figure 13a) with a Kung Fu yell ("heee-YAH"). Each foe requires a different gesture. Foes not destroyed enter the player's personal space and injure the player. Enough injuries will cause the player's defeat.

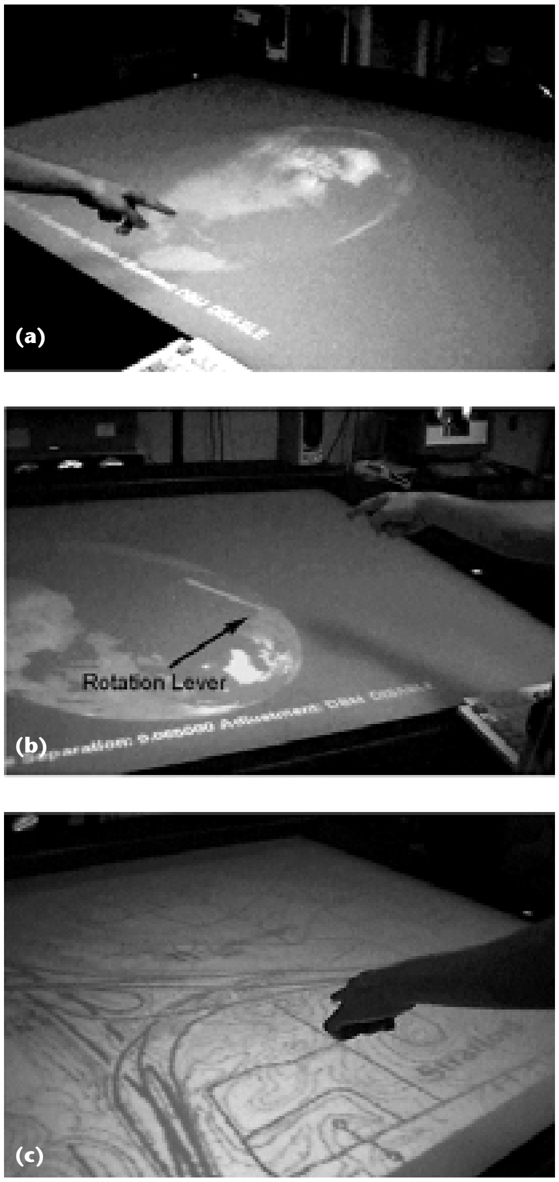
Faculty and graduate students in the Graphics, Visualization, and Usability lab have used the system. They found the experience compelling and balanced. Since it's difficult for the game master to keep pace with the player, two or more game masters may participate (Figure 12a). The Perceptive Workbench's object tracker scales naturally to handle multiple, simultaneous users. For a more detailed description of this application, see Starner et al.[22]

3D terrain navigation

We developed a global terrain navigation system on the virtual workbench that lets you fly continuously from outer space to terrain or buildings with features at 1 foot or better resolution.[19] Since features are displayed stereoscopically, the navigation is both compelling and detailed. In our third-person navigation interface, users interact with the terrain as if it were an extended relief map laid out below them on a curved surface. Main interactions include zooming, panning, and rotating. Since users are head-tracked, they can move their head to look at the 3D objects from different angles. Previously, interaction took place by using button sticks with 6-DOF electromagnetic trackers attached.

We employed the deictic gestures of the Perceptive Workbench to remove this constraint. Users choose the direction of navigation by pointing and can change the direction continuously (Figure 14c). Moving the hand toward the display increases the speed toward the earth and moving it away increases the speed away from the earth. Users accomplish panning by making lateral gestures in the direction to be panned (Figure 14a). They rotate views by making a rotating gesture with their arm (Figure 14b). Currently, users choose these three modes by keys on a keyboard attached to the workbench. In the future we expect to use gestures entirely (for example, pointing will indicate zooming).

Although some problems exist with latency and accuracy (both of which will lessen in the future), users can successfully employ gestures for navigation. In addition, the gestures are natural to use. Further, we find that the vision system can distinguish hand articulation and ori-

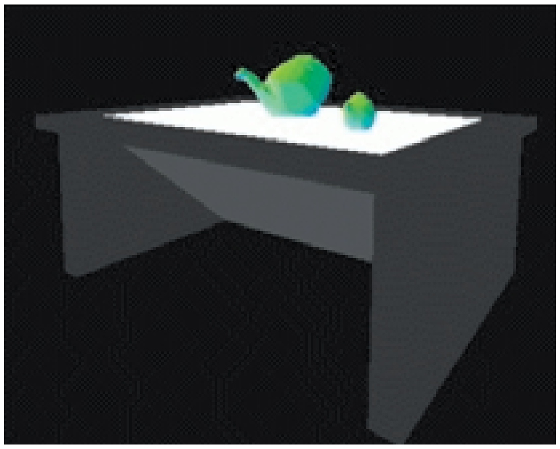


14 Terrain navigation using deictic gestures: (a) panning, (b) rotating (about an axis perpendicular to and through the end of the rotation lever), and (c) zooming.

entation quite well. Thus we'll be able to attach interactions to hand movements (even without the larger arm movements). At the time of this writing, we developed a hidden Markov model framework that lets users train their own gestures for recognition. This system, in association with the terrain navigation database, should permit more sophisticated interactions in the future.

Telepresence, CSCW

As another application of the Perceptive Workbench, we built a small 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. Every one of their actions is immediately applied to a VR model of the workbench mirroring the current state of the real desk. 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 (Figure 15). 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.



15 A virtual instantiation of the workbench that mirrors, in real time, the type and position of objects placed on the real desk.

Future work

We propose several improvements to the Perceptive Workbench. Use an active pan/tilt/zoom camera mounted underneath the desk for higher resolution reconstruction and improved recognition for small objects. Deploy the color side camera to improve 3D reconstruction and construct texture maps for the digitized object. Modify the reconstruction code to handle holes in objects and to correct errors caused by non-square pixels. Improve the latency of the gesture-rendering loop through code refinement and the application of Kalman filters. With a difficult object, recognition from the reflections from the light source underneath can be successively improved by using cast shadows from the light sources above or the 3D reconstructed model directly. Employ hidden Markov models to recognize symbolic hand gestures[12] 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.

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. These elements combine seamlessly into the same interface and can be used in various applications. In addition, the sensing system is relatively inexpensive, retailing for approximately $1,000 for the cameras and lighting equipment plus the cost of a computer with one or two video digitizers, depending on the functions desired. As seen from the multiplayer gaming and terrain navigation applications, the Perceptive Workbench provides an untethered and spontaneous interface that encourages the inclusion of physical objects in the virtual environment. ■

Acknowledgments

This work is supported in part by a contract from the Army Research Lab, an NSF grant, an ONR AASert grant, and funding from Georgia Institute of Technology's Broadband Institute. We thank Brygg Ullmer, Jun Rekimoto, and Jim Davis for their discussions and assistance. 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.

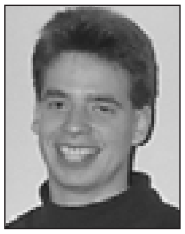
References

1. A.F. Seay et al., "Multimodal Interaction Techniques for the Virtual Workbench," *Proc. CHI 99 Extended Abstracts*, ACM Press, New York, 1999, pp. 282-283.
2. B. Ullmer and H. Ishii, "The metaDESK: Models and Prototypes for Tangible User Interfaces," *Proc. ACM Symp. User Interface Software and Technology (UIST 97)*, ACM Press, New York, 1997, pp. 223-232.
3. O. Bimber, "Gesture Controlled Object Interaction: A Virtual Table Case Study," *Proc. 7th Int'l Conf. in Central Europe on Computer Graphics, Visualization, and Interactive Media (WSCG 99),* Vol. 1, Univ. of West Bohemia, Plzen, Czech Republic, 1999.
4. M. Krueger, *Artificial Reality II*, Addison-Wesley, Reading, Mass., 1991.
5. W. Krueger et al., "The Responsive Workbench: A Virtual Work Environment, *Computer*, Vol. 28, No. 7, July 1995, pp. 42-48.
6. N. Matsushita and J. Rekimoto, "Holo Wall: Designing a Finger, Hand, Body, and Object Sensitive Wall," *Proc. ACM Symp. on User Interface Software and Technology (UIST 97)*, ACM Press, New York, 1997, pp. 209-210.
7. R. van de Pol et al., "Interaction in Semi-Immersive Large Display Environments," *Proc. Virtual Environments 99*, Springer-Verlag Wien, Wien, Austria, 1999, pp. 157-168.
8. R. Bolt and E. Herranz, "Two-handed Gesture in Multimodal Natural Dialog," *Proc. UIST 92*, ACM Press, New York, 1992, pp. 7-14.
9. D. Sturman, *Whole-hand Input*, PhD thesis, Massachusetts Inst. of Tech. Media Lab, Cambridge, Mass., 1992.
10. C. Wren et al., "Pfinder: Real-Time Tracking of the Human Body," *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 19, No. 7, 1997, pp. 780-785.
11. J.M. Rehg and T. Kanade, "Visual Tracking of High DOF Articulated Structures: An Application to Human Hand Tracking," *Proc. European Conf. on Computer Vision (ECCV 94)*, Lecture Notes in Computer Science, Vol. 800, Springer-Verlag Berlin, Berlin, May 1994, pp. 35-46.
12. T. Starner, J. Weaver, and A. Pentland, "Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video," *IEEE Trans. PAMI*, Vol. 20, No. 12, 1998, pp. 1371-1375.
13. R. Sharma and J. Molineros, "Computer Vision-Based Augmented Reality for Guiding Manual Assembly," *Presence*, Vol. 6, No. 3, 1997, pp. 292-317.
14. J.W. Davis and A.F. Bobick, *SideShow: A Silhouette-Based Interactive Dual-Screen Environment*, Tech Report No. 457, MIT Media Lab, Cambridge, Mass., 1998.
15. J. Underkoffler and H. Ishii, "Illuminating Light: An Optical Design Tool with a Luminous-Tangible Interface," *Proc. of CHI 98*, ACM Press, New York, 1998, pp. 542-549.
16. P. Wellner, "Interacting with Paper on the Digital Desk," *Comm. ACM*, Vol. 36, No. 7, 1993, pp. 86-89.
17. G.D. Kessler and L.F. Hodges, "The Simple Virtual Environment Library: An Extensible Framework for Building VE Applications," *Presence*, Vol. 9, No. 2, April 2000, pp. 187-208.
18. P. Lindstrom et al., "Real-Time, Continuous Level of Detail Rendering of Height Fields," *Proc. Siggraph 96*, ACM Press, New York, 1996, pp. 109-118.
19. Z. Wartell, W. Ribarsky, and L.F. Hodges, "Third Person Navigation of Whole-Planet Terrain in a Head-Tracked Stereoscopic Environment," *Proc. IEEE Virtual Reality 99*, IEEE CS Press, Los Alamitos, Calif., 1999, pp. 141-149.
20. G.W. Fitzmaurice et al., "Bricks: Laying the Foundations for Graspable User Interfaces," *Proc. CHI 95*, ACM Press, New York, 1995, pp. 442-449.
21. H. Ishii and B. Ullmer, "Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms," *Proc. CHI 97*, ACM Press, New York, 1997, pp. 234-241.
22. T. Starner et al., "Mind-Warping: Towards Creating a Compelling Collaborative Augmented Reality Gaming Interface through Wearable Computers and Multimodal Input and Output," *Proc. Int'l Conf. on Intelligent User Interfaces (IUI 2000)*, ACM Press, New York, 2000, pp. 292-317.
23. 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,* Vol. 49, No. 1, 1990, pp. 68-84.
24. S. Sullivan and J. Ponce, "Automatic Model Construction, Pose Estimation, and Object Recognition from Photographs Using Triangular Splines," *IEEE Trans. PAMI*, Vol. 20, No. 10, 1998, pp. 1091-1097.
25. D. Daum and G. Dudek, "On 3D Surface Reconstruction Using Shape from Shadows," *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR 98)*, IEEE CS Press, Los Alamitos, Calif., 1998, pp. 461-468.
26. B.G. Baumgart, *Modeling for Computer Vision*, PhD thesis, Dept. of Computer Science, Stanford University, Palo Alto, Calif., 1974.
27. P.L. Rosin and G.A.W. West, "Non-Parametric Segmentation of Curves into Various Representations," *IEEE Trans. PAMI*, Vol. 17, No. 12, 1995, pp. 1140-1153.
28. *Twin Solid Modeling Package Reference Manual*, Computer Aided Design and Graphics Laboratory (CADLab), School of Mechanical Engineering, Purdue University, West Lafayette, Ind., 1995, http://cadlab.www.ecn.purdue.edu/cadlab/twin/.
29. A. Laurentini, "How Many 2D Silhouettes Does It Take to Reconstruct a 3D Object?" *Computer Vision and Image Understanding*, Vol. 67, No. 1, July 1997, pp. 81-87.
30. B. Watson et al, "The Effects of Variation of System Responsiveness on User Performance in Virtual Environments," *Human Factors*, Vol. 40, No. 3, Sept. 1998, pp. 403-414.
31. P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring Error on Simplified Surfaces," *Computer Graphics Forum*, Vol. 17, No. 2, June 1998, pp. 167-174.

Bastian Leibe *is currently finishing his Diplom degree at the University of Stuttgart, Germany. He received his MS in computer science from Georgia Institute of Technology in 1999, where he was a member of the Graphics, Visualization and Usability (GVU) Center and the Contextual Computing Group. His research interests include computer vision, interaction techniques, visualization, and machine learning.*

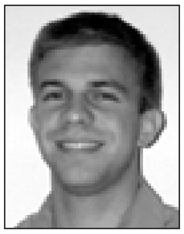
Thad Starner *is founder and director of the Contextual Computing Group at Georgia Institute of Technology's College of Computing. He graduated from the Massachusetts Institute of Technology in 1991 with BS degrees in computer science and brain and cognitive science. He later returned to the MIT Media Lab, where he earned his MS and PhD in computer science in 1995 and 1999, respectively. His current research explores on-body perception systems and intelligent agents for continuous-access wearable computers. Starner is also a founder of Charmed Technology.*

William Ribarsky *is Principal Research Scientist and leader of the Data Visualization group in the GVU Center, College of Computing, at Georgia Institute of Technology. He received a PhD in physics from the University of Cincinnati. His research interests include navigation, dynamical query, and analysis of large-scale information spaces using interactive visualization and hierarchical representations; 3D multimodal interaction; and virtual environments. He is a member of the Steering Committee for the IEEE Virtual Reality Conference and an associate editor of* IEEE Transactions on Visualization and Computer Graphics.



Zachary Wartell *is a PhD candidate in the College of Computing at Georgia Institute of Technology. He received his BS there in computer science in 1994. For several years he worked in the virtual environments group at the Human Interface Technology Center at NCR. His research interests are interactive 3D computer graphics and virtual reality. His dissertation topic is on algorithms for stereoscopic head-tracked displays.*

David Krum *is a PhD student in the College of Computing at Georgia Institute of Technology. His research interests are 3D interaction techniques and distributed collaborative virtual environments. He received his BS in engineering and applied science from California Institute of Technology in 1994 and his MS in computer science from the University of Alabama in Huntsville in 1998.*



Justin Weeks *is a graduate of the College of Computing of the Georgia Institute of Technology, where he specialized in graphics and computer vision. His research interests include vision-based human-computer interaction through gesture recognition and augmented reality. He is currently in the "real world," writing Java for a consulting firm while he ponders the rewards and consequences of going back to grad school.*

Bradley Singletary *is a PhD student in the College of Computing and the GVU Center at the Georgia Institute of Technology. He received a BA in computer and information sciences from East Tennessee State University in 1994. His research interest is in augmentation and improvement of the human experience through wearable and handheld computers. He is a member of the Contextual Computing Group at Georgia Institute of Technology.*

Larry Hodges *is Associate Professor in the College of Computing and head of the Virtual Environments Group in the Graphics, Visualization, and Usability Center at Georgia Institute of Technology. He received his PhD from North Carolina State University in computer engineering in 1988. He is a senior editor of the journal* Presence: Teleoperators and Virtual Environments*, is on the editorial board of* CyberPsychology & Behavior*, and is a member of the Organizing Committee for the annual IEEE Virtual Reality Conference.*

Readers may contact Thad Starner or William Ribarsky at the Graphics, Visualization, and Usability Center, College of Computing, Georgia Institute of Technology, 801 Atlantic Dr., Atlanta, GA, 30332-0280, e-mail {thad, ribarsky}@cc.gatech.edu.