Touching the Visualized Invisible: Wearable AR with a Multimodal Interface

MATHIAS KÖLSCH, RYAN BANE, TOBIAS HÖLLERER, MATTHEW TURK

Computer Science Department

University of California

Santa Barbara, CA 93109

Abstract

Wearable computers and their novel applications demand more context-specific user interfaces than traditional desktop paradigms can offer. We describe a multimodal interface to a wearable computing system and explain how it enhances a mobile user's situational awareness and provides new functionality. Our mobile augmented reality system visualizes otherwise "invisible" information encountered in urban environments. A versatile filtering tool allows interactive display of occluded infrastructure and of dense data distributions such as room temperature or wireless network strength, with applications for building maintenance, emergency response, and reconnaissance missions. Multiple input modalities – vision-based hand gesture recognition, a one-dimensional tool, and speech recognition – are combined with three late integration styles to provide intuitive and effective means to operate the complex application functionality. Users can select visualizations for various semantic information types with spoken commands and manipulate virtual objects through gestural input. The system is demonstrated and evaluated in a realistic indoor and outdoor task environment.

Keywords: augmented reality, wearable computing, multimodal interface, hand gesture recognition, information visualization

Introduction

Wearable computers – computers that can be worn or carried easily – have already evolved into tremendously powerful and versatile devices: PDAs, cell phones with integrated video recorders, wrist-watches – and soon there will be garments with built-in computational power. Unfortunately, their human interface capabilities have not evolved as rapidly and are in fact severely limited by the devices' continuously shrinking form factors. Traditional interfaces, such as keyboards and LCD screens, can only be as big as the device's surface.

Fortunately, the adverse goals of device size and interface area can both be accommodated for by expanding the interaction area beyond the device's dimensions: Augmenting the reality through head-worn displays allows for information visualization in the entire field of view, extending beyond the physical size of the display. Equally, hand gestures performed in free space, recognized with a head-worn camera, are not constrained to the hardware unit's dimensions. Speech is similar to vision-based gesture recognition as it also travels through space without devices as media. It thus also expands the user interface (UI) beyond the device. Combining these advantageous input and output modalities harbors the promise to allow for more complex interaction.

In this article, we detail our experiences with various input devices and modalities and discuss their advantages and drawbacks in the context of interaction tasks in mobile environments. We show how to integrate the input channels in such a way as to use each involved modality beneficially, as well as to enhance the interface's overall usability.

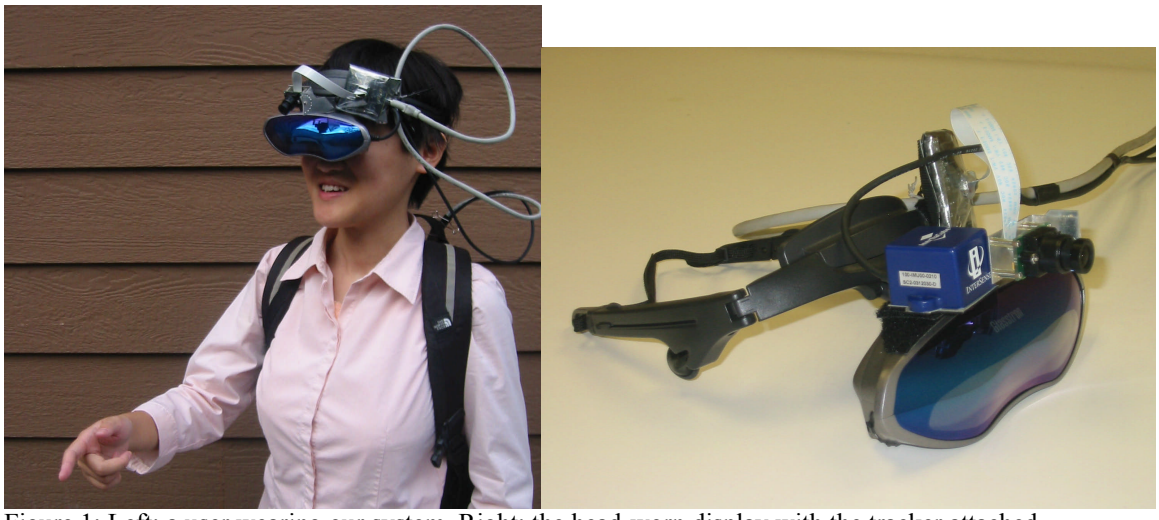


Figure 1: Left: a user wearing our system. Right: the head-worn display with the tracker attached.

Augmented reality (AR), the placement of computer-generated information directly on top of the user's perception of the physical world, is a powerful UI paradigm and our wearable output modality of choice. We use a head-worn display and a small camera mounted on the glasses to provide a video see-through mode (see Figure 1). The camera image is first processed with vision-based gesture recognition algorithms. Then it is

passed on to the AR module which generates hand tracking feedback, 3D graphics that are registered with the current camera view, and a screen-stabilized UI for conventional mouse-interaction functionality as a fallback solution. Our wearable computer is simulated by a bulky prototype backpack system based on two medium-performance laptops.

The motivation behind this work is to show how multi-modal interface techniques can stretch the boundaries of the types of tasks that can be performed on a wearable platform. We will discuss the lessons learned while transitioning from a keyboard-and-mouse interface (with all components spread out on a bench) to a multimodal interface. Our contributions fall mainly in two categories: interactive information visualization techniques and the multimodal interfaces that control these.

- A visualization aid is presented, called the Tunnel Tool. It allows a user to interactively inspect complex information while in the field and immersed in AR. It can "see through walls" and visualize occluded objects. It can also filter dense, volumetric data sets and present them in the form of visually more informative data slices.
- Various one-dimensional input devices and methods are evaluated informally for their suitability to our task requirements.
- A vision-based hand gesture recognition module and the input functionality it can provide are described.
- The integration of the above modalities plus speech recognition is shown to be capable of controlling the entire AR system, including in particular the mentioned Tunnel Tool.

Providing users with more powerful applications and the means to control these applications often fails because of limitations of the interaction devices. On the other hand, many non-traditional interaction paradigms seem unnecessary for many desktop and other conventional applications. Especially in a largely unexplored field such as wearable computing it is thus important to develop novel interaction metaphors, such as our Tunnel Tool, concurrently with the user interfaces that control them, such as our speech and gesture modalities working in concert. We followed this approach: after a review of related work, the interaction metaphors are presented in the next section, followed by a description of the means of interaction – the multimodal user input. Before drawing conclusions from our work, we describe the hardware and software structure of our experimental system.

Related Work

This section looks at related work in mobile AR and multimodal UIs. Related work pertaining to computer vision methods is discussed in the Gesture Recognition section.

Feiner et al. [FWK+95] explored the possibilities of AR for visualizing hidden architectural information as direct overlays on wall structures. The Touring Machine [FMHW97] was the first outdoor mobile AR system that overlaid 3D computer graphics

on top of the physical world. Follow-up research explored a series of mobile AR UIs, [HFT+99], [BHF02], including world- and screen-stabilized navigational aids. None of these mobile UIs made use of vision-based gestural input or focused particularly on multimodal interaction techniques. We present interfaces that interactively control such visualizations using multi-modal interface techniques. An advantage of our gesture-based selection over the head-orientation–based selection mechanism employed in the above work is that the user's hand can operate in the world reference frame, such that rotating the head does not get in the way of fine-grain selection tasks.

Researchers at the University of South Australia have implemented a series of MARS systems for terrestrial navigation [TDP+98], [PT01]. Their more recent prototypes employ a pinch-glove-based interface to creating and manipulating augmented material in the outdoors [TP02]. Visual markers allow 3D location and orientation detection with a camera. They show many compelling interface applications with these gloves, but the input device is bulky and prevents the user from using their hands for other tasks. Our gesture interaction does not require a glove to be worn, and our speech recognition component takes over tasks that are not naturally spatially arranged, such as command selections.

Julier et al. [JLB+00] present a framework for information filtering in mobile AR UIs. Their focus is on automatically determining from task information and user context what information should be displayed and what not. In contrast, our focus lies on multimodal interactive techniques to probe large amounts of information and simplify visualizations.

Reitmayr and Schmalstieg at the Technical University Vienna adapted their Studierstube AR environment for use with a mobile system, and explore collaborative user interface issues [RS01]. Researchers at the University of Munich developed a component-based MARS and explore its use for the maintenance of power plants [KCD+01].
The Nokia Research Center in Tampere, Finland, has experimented with several augmented reality interfaces for palmtop- and wearable computer based navigational guidance [SL00], [SLS01].

Sawhney [SS98] designed and evaluated speech-controlled audio interfaces for wearable computers. We use speech to let the user issue commands that would be more difficult to express in other media, but we provide backup interaction techniques that could be employed when speech input would fail or is impossible due to situational constraints (noisy environments, or imposed silence).

Some functionality of our 3D tunnel tool is reminiscent of concepts pioneered by Bier et al in their work on magic lenses [BSP+93]. We extend the idea of different layers that can be visualized by 2D lenses to a physical 3D space and allow for the layers to be controlled by voice commands.

Multimodal UIs that combine speech with gestural input date back to Bolt's "Put-That-There" system [Bol80], in which coordinated speech and magnetically tracked pointing gestures controlled a computer graphics world. A number of researchers have tackled the

problem of multimodal integration for speech, gestures, gaze, and 3D tracking information [WG89], [KST93], [PSOS98], [KOR+02] but none of this work has been implemented on a mobile platform.

Cohen and colleagues [CJM+98] present the 2D multimodal Quickset architecture, which was implemented on mobile hand-held tablet computers. In this work, speech input is coordinated with 2D pen gestures to mutually disambiguate modalities. In follow-up work, Kaiser and colleagues [KOM+03] applied similar multimodal integration to 3D gesture, spoken language, and focus areas to control VR and AR interfaces. This interface, however, was not implemented or tested on a mobile platform.

If the device can be folded to a smaller form factor for transportation and stowing, the wearable computer loses one of its important properties: interactional constancy, the property of being always accessible [Man97].

Kurata et al.'s Hand Mouse [KOKS01, KKS02] is another vision-based interface to a wearable computer that leverages a high-speed LAN connection to a computing cluster to facilitate real-time hand tracking. In comparison, our system does not require stationary infrastructure and it uses multiple image cues for more robust hand detection and tracking. Starner et al. have also demonstrated wearable vision-based interfaces: multiple hand gestures can be recognized in combination with an active illumination source [SAAG00]. That system requires hands to be close to a chest-worn "Gesture Pendant". Less constrained sign language recognition was shown in combination with three wired acceleration sensors, worn on the hands and shoulder [BSLJ03].

New Interaction Metaphors

This section explains the visualization aids that our wearable system offers its users. We describe the display techniques that allow visualization of multiple overlays of "invisible" information over the physical world. We describe the visualization tools that allow for data filtering and ultimately enable the user to manage the wealth of available information. From the complexity of filter control parameters it will become obvious that traditional interaction methods are flawed and unsuitable for the mobile use. The multimodal interface that we built to overcome this problem will be explained in the next section. Phrases in "quotes" are voice commands that are to be processed by a speech recognition system.

Visualization Environments

Geometric primitives are grouped semantically into structural walls, furniture, temperature clouds, building wireframes, and so on. The AR module renders these virtual augmentations to the visible world differently depending on their momentary screen position. For example, the building wireframes can be shown to the edge of the screen, while they are not rendered at the screen center. There are three screen environments: the outermost, largest, and always-on environment is called the Surroundings environment and contains all augmentations in view that are not part of one of the other environments. The Tunnel environment is specific to the screen area covered by the Tunnel Tool. The

Finger environment represents an area of close proximity to the user's hand when in pointing mode and can, for example, be used to select objects or to add annotations to objects. At system startup, no augmentation is displayed. The user gives simple voice commands to add semantic visualization groups to the environments (see Speech Recognition section). Figure 2 has only the Surroundings environment turned on; activated are building wireframes and (in the right picture) "insufficient wireless" signal strength.

Figure 2: Left: AR-view of one of the two buildings that we built models for. Right: visualization of spots with "insufficient wireless" signal strength. The density of the data prohibits making out differences in the data intensity.

Tunnel Tool

The voice command, "open tunnel," brings up our main visualization aid, called the Tunnel Tool. It enables the user to mask out items that would obstruct the view of otherwise hidden objects, while providing orientation and navigation cues as context in the environment around the tunnel tool.

The tool is apparent to the viewer as a bluish plane that occludes the real world and occupies part of the screen. In front of the plane those items are displayed that the user has added to the Tunnel environment, while the Surroundings environment displays its items independently. The three dimensional layout of the Tunnel Tool that creates this impression is shown in Figure 3.
The Tunnel defines a frustum shape into the virtual representation of the world, which contains a series of three vertical planes, defining four regions within the tunnel. Moving forward from the user's position, the first of these regions starts at the user and extends to the first plane. Objects currently selected into the Tunnel environment that fall within this first Transparent Region are not rendered. The second region, called the Context Region, is intended to give the user some context on the objects he is viewing, so objects within this region are rendered in wireframe. The region between the second and third planes is the Focus Region. This region represents the area the user is currently interested in, and objects falling within the Focus Region are fully rendered. To avoid signal mixing, the Transparent Region behind the Focus Region is not rendered.

The user can slide the Focus Region forward and backward in the tunnel using a trackball, allowing him to view objects at any distance from his current position, even when other objects would normally occlude them. This functionality can be used to interactively explore complex three dimensional information, such as the structure of a building or volumetric visualization of wireless network strength. To give more control over the view, the user can also adjust the length of the context region and the focus region.

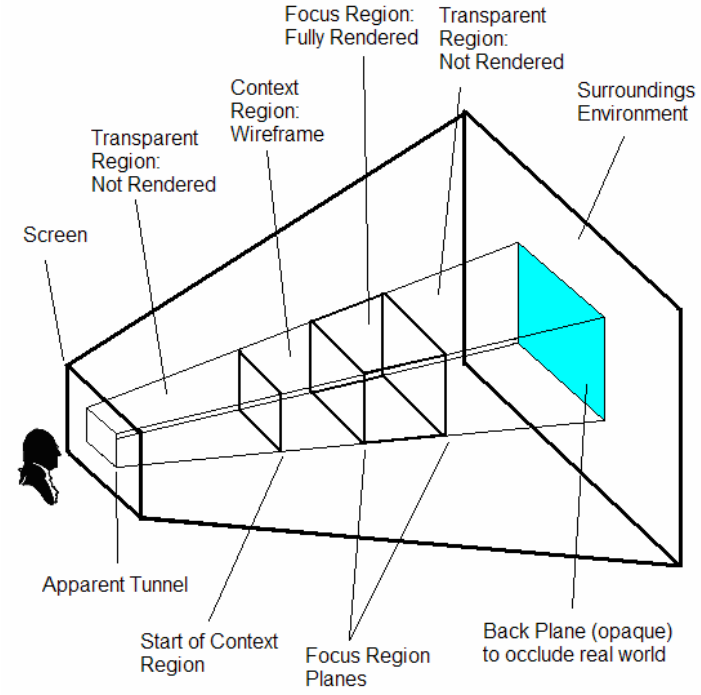


Figure 3: Schematic view of the Tunnel Tool. Only objects inside the Focus Region are rendered in full. Objects that fall within the Context Region are rendered as wireframes. To either side of the apparent tunnel the objects are rendered as specified in the Surroundings environment.

Tunnel Tool Functionality: Slicing

The tunnel tool presents volumetric data in short segments, or slices. This is done by adding the item in question to the Tunnel environment and sliding the focus region through the data cloud. This masks out data in front of the focus region and data behind it, simplifying the user's view and allowing him to explore the data. Figure 4 illustrates how this technique simplifies localization of a hotspot – a higher density are in a data cloud.

In initial trials, the speech command "tunnel follow finger" attached the tool's main axis to the location of the tracked hand, allowing 2-dimensional relocation with the hand. The idea was to enable the user to interactively sample the space with the tunnel tool, leaving arbitrary amounts of context space to either side of it. However, we found it more convenient to leave the tunnel tool fixed at the center of the view since we felt that head orientation changes were a more natural interface to moving the tunnel around than

pointing gestures. In hindsight it makes sense to keep the tool at the center of the user's field of view, because it is likely to be the center of attention as long as it is being displayed.

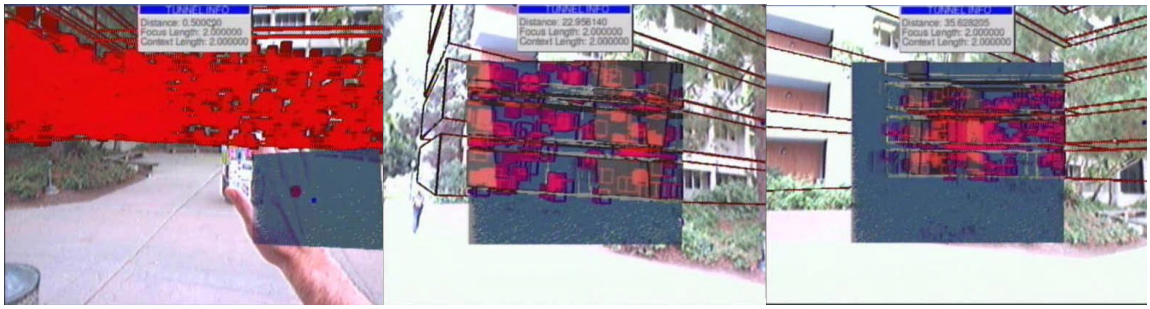


Figure 4: The Tunnel Tool (the bluish rectangular area) in action. Left: A first version of the Tunnel Tool had the functionality to attach it to the hand location and move it across the display. Middle: As discussed below, the current version keeps it centered in the field of view. In this picture, a slice of "insufficient wireless" data is cut out of the entire data cloud seen on the left. Now the density (that is, the number of reports of low signal strength) can be judged. The right picture shows a hotspot area where many reports are clustered close together.

Two of us recently extended the Tunnel Tools functionality [BH04]. It can now be spatially confined by "snapping" to semantic objects such as building floors, walls, or individual rooms. This avoids artifacts caused by partially displayed walls. The volume slicer can also operate on selective rooms only, permitting an exploration style that more closely resembles human building space exploration.

Tunnel Tool Functionality: X-ray Vision

The Tunnel Tool allows a user to have "virtual superman-style X-ray vision", letting him see through physically present walls to the virtual representations of the objects beyond. This is conceptually different from the Tunnel Tool's slicing functionality that helps exploring volumetric data since the view of the objects of interest is obstructed by real, physical objects instead of purely virtual data.

Manipulation of Virtual Objects

Semantic virtual objects such as a desk can be selected and manipulated. Simple geometric objects (boxes, spheres, lines) can be inserted into the virtual scene and subsequently manipulated. The object manipulation occurs in a three-step process: the object is first positioned, then resized, and lastly rotated. We use a mix of hand gestures, trackball input, and voice commands to perform these tasks. Figure 5 shows a schematic view of the associated state machine.

After issuing the voice command "select picking tool for finger," the user can select an object by pointing at it with his finger and saying "select object". The 'relocate' mode is then activated by saying "move object." Or he can insert a new virtual object with "insert *insertable,*" where *insertable* is one of box, sphere, or line. After insertion, the 'relocate' mode is entered automatically with the object initially placed at the screen center.

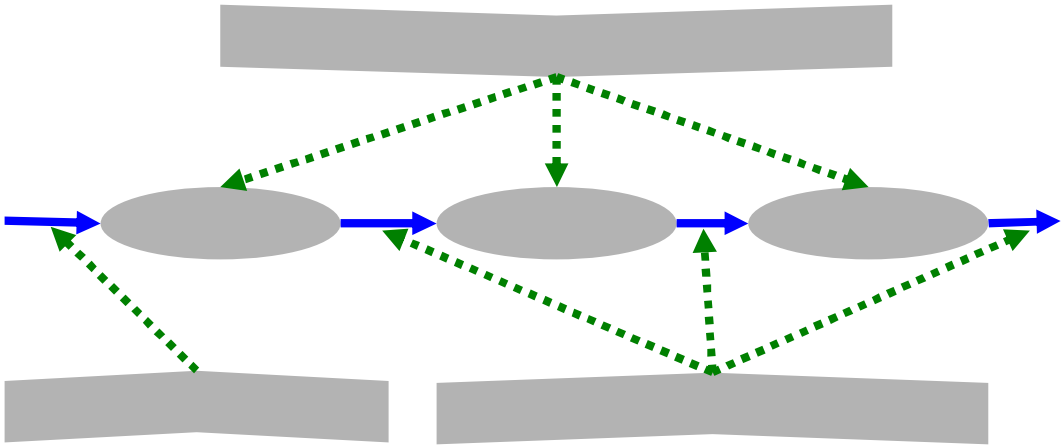


Figure 5: State machine showing the three-step process for manipulating virtual objects.
* We initially used the head pose as input for the object's orientation as well, but found it inconvenient that it rotated while attempting to look at it from multiple angles.

User input is performed with a combination of three signals. Without hand gestures, the object stays at the screen-stabilized location at which it was when the mode was entered. When hand tracking is active, the object follows the hand location instead of being screen-stabilized. Its depth in the scene is always modified with trackball input. This allows concurrent input of three dimensions: head pose and hand gestures for (x, y) and trackball input for (z). The gesture commands work as follows: the user first makes the 'lock' hand gesture, which sets the system to track the motions of his hand and apply them to the object's position (left image in Figure 6). The user then moves his hand left and right to move the object on the x axis, and up and down to move the object along the y axis. When he is satisfied with the object's position (actually its far, left, top corner), he makes the 'release' hand gesture which stops the system from applying his hand motions to the object (center image in Figure 6). Note that the input range is not limited to the camera's or HMD's field of view since the head orientation is tracked. Clicking with the trackball, the voice command "finished," or a particular hand gesture prompt the transition into the 'resize' mode. There, the same input modalities allow 3-dimensional object resizing by dragging it on its proximal, right, bottom corner. Again, a clicking, gesturing, or "finished" input commands each exit this mode and prompt transition to the next in which the object can be rotated around each axis, again with the same input modalities. In the last manipulation stage, the orientation of the object can be modified in an equivalent manner, however, no head pose information is used. The voice commands "move object", "resize object", and "orient object" enter the respective manipulation modes immediately.

Figure 6: Resizing an object with gestures. The left picture shows the user's hand in the 'lock' posture from which on hand location changes rescale the virtual object (the gray box with a green selection frame around it). In the middle picture, the user performs the 'release' posture, which decouples hand movements and object scale again. The right picture shows the placement of the virtual object in the left upper corner and representations for power and networking equipment near the floor.

Selection and manipulation with hand gestures has advantages over the frequently-used selection by gaze direction (e.g. [FMHW97]). The experience with those systems was that the head had to be held very still in order to make a selection. We can take advantage of a two-step approach where the head determines the main direction and the hand facilitates fine selection. This is also important for explorative probing of the environment, that is, selection preceded by a change in visualization style (labels popping up). On the other hand, head tracking extends the input range over that of hand tracking with only a static camera to the entire 4pi steradians.

Paths

While not the central focus for this particular scenario, a path finding and navigational guidance capability is critical to many emergency response situations, as for example building evacuation. To allow for multi-purpose application of our system, we implemented a modification of Dijkstra's shortest path algorithm and a visualization that displays the result. The user can set a path to either an object he has picked, or to the center of the focus region in the tunnel tool. To initiate path computation, the user uses the voice command "set path from here to tunnel." The visual result is shown in Figure 7.

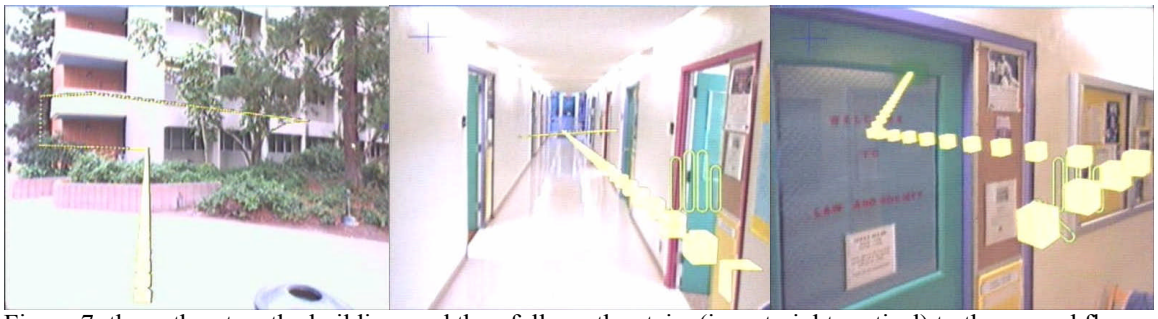


Figure 7: the path enters the building, and then follows the stairs (in a straight vertical) to the second floor.

Interacting with the Visualized Invisible

In this section we describe both the realization of the multiple modalities as well as how they are put to use to control the visualization tools. Over time, we experimented with a number of different input modalities for the various tasks. This section details our experiences, the lessons we learned for future implementations, and the choices we eventually made.

Design Choices

Our development and fallback interface of a keyboard- and mouse-controlled screen-stabilized GUI was our starting configuration. We spread out the components of the wearable computer on a park bench and operated the HMD/tracker unit independently of the input devices. The inconvenience of conventional devices for controlling complex interaction metaphors while being immersed in AR became apparent very quickly. Since not every input interface is well-suited to every task, we experimented with a pool of interface devices: said keyboard and mouse, a wireless handheld trackball, a ring-style trackball, hand gesture recognition, speech recognition, and head orientation tracking. With informal experimentation we gathered first-hand results on the devices' suitability for a number of diverse tasks.

We felt that discrete, "binary" parameters – previously mapped to one key each – are best accessed by equally binary speech commands. Speech also allows for natural extension to allow commands with modifiers. The same approach does not work well for keys (that would have to be pressed in succession) and we had mapped a combination of action and modifier onto a single key. For example, the "add" speech commands are parameterized by the names of bundles and items, yet we had chosen context-free, dedicated keys for each combination of actions and parameters. Other typical commands that we preferred to be mapped to voice input are: "take snapshot", "save", "discard", "open/float/close tunnel", "add bundle temperature to tunnel".

The other multi-dimensional control that our visualization interface requires is for positioning, sizing, and orienting objects. This could be done with multiple sequential 1D and/or 2D input steps, but at least for repositioning this is very awkward and differs starkly from the direct interaction employed for positioning a physical object. To achieve concurrent input of three dimensions of data, hand tracking provided two dimensions and an auxiliary device was to supply the third. Many choices of input modalities are available for quasi-continuous 1-dimensional input, for example, to control the distance to the Tunnel Tool's Focus Region. The traditional methods include mice, trackballs, trackpoints, touchpads, and key presses for extended-periods of time. We discarded our initial plans to use one dimension of a touchpad's output (or a special device like Synaptics' ScrollStrip) after realizing that it would necessitate mounting to some clothing, requiring fairly inflexible hand positions during operation. Holding it with the other hand or attaching it to the other arm would restrict the independence of both arms. As expected, regular mice and keyboards turned out unsuitable to the environment. The mouse wheel by itself can also supply one-dimensional input. However, the most common hardware implementation of mouse wheels does not permit input of parameters

from a continuous domain. Instead, they send discrete stepping commands generated by optical switches and are thus unsuited for quasi-continuous input.

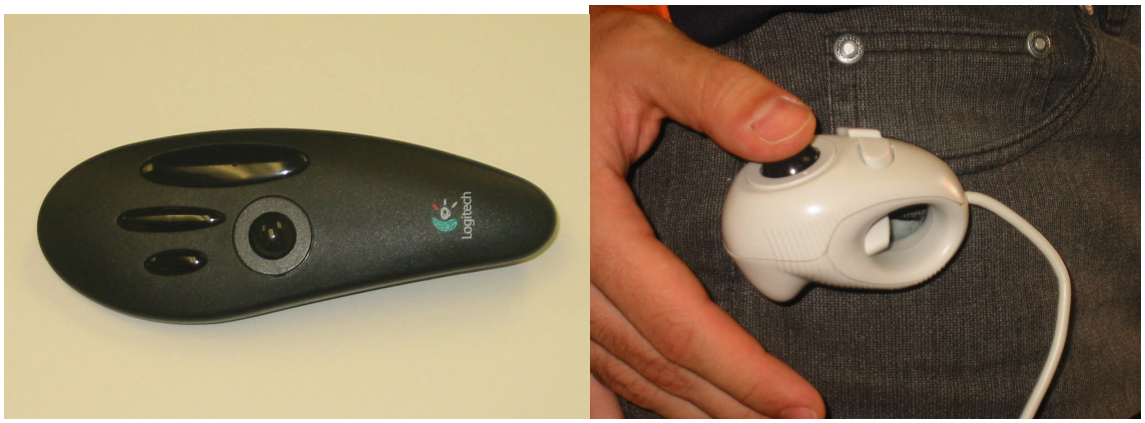


Figure 8: Our two favorite devices to provide one-dimensional input to our system, a wireless handheld trackball and a trackball that can be worn similar to a ring. Here it is attached to the user's pants with a Velcro strip.

Our favorite candidates were a wireless, handheld trackball and a "ring trackball" (see the pictures in Figure 8). We eventually chose the second device because it allows multi-finger input and because it is easier to retrieve and stow away, especially after we provided for attachment to the user's pants with a Velcro strip. In addition, the device can be left dangling from the index finger if the user chooses the use the same hand for gesturing and trackball operation, thus permitting single-handed and less encumbering hand gesture input.

Only one dimension of the 2D ball motions was needed for our system, but the device has the full functionality of a 3-button mouse and thus allows for UI extensibility. In some system states (during object manipulation, for example), button clicks are interpreted as one of a set of redundant input modalities. Another advantage of trackballs and similar concepts is that they deliver information about unbound relative movement, that is, their input domain is infinite. This is important if the interaction range is also not limited – as in the case of moving the Focus Regions of Tunnel Tool.

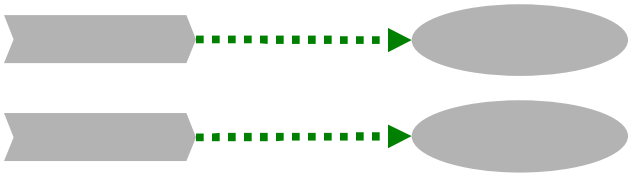
Mapping the trackball's two continuous dimensions to the planar interaction parameters in our interface would prevent the advantages of registered interaction of the hand with virtual objects would have been lost. The drawing interface also suffers from mouse and trackball input because it is hard to create smooth two-dimensional curves with these handheld devices.

Multimodal Integration

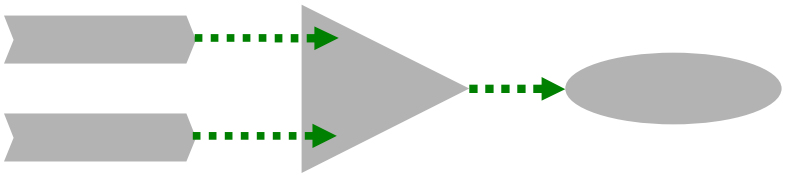
The system integrates three different modalities of interaction: hand gesture input, voice commands, unidirectional trackball motions, and head orientation. Feature extraction and interpretation happens independently on all channels, that is, we combine the modalities with late integration after grammatically correct sentences have been extracted and the location and posture of the hand is determined. The style of high-level interpretation

differs according to input commands and system state. Three styles of late integration are blended in a way to maximize the overall usability while choosing input from the best-suited modality for a task.

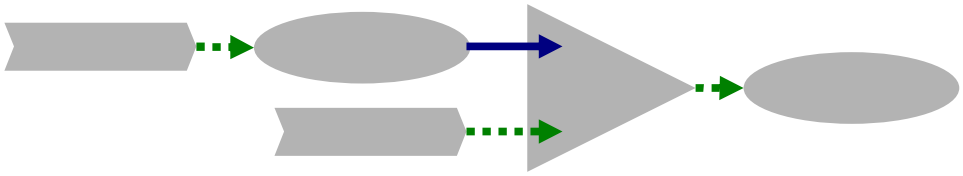
- *independent, concurrent interpretation:*
  Input of this style is interpreted immediately and as atomic commands. In our system, most speech can be given at any time and have the same effect at any time. For example, the speech directive "add networking to surroundings" can occur simultaneously with gesture or trackball commands and it is interpreted independent of their state. Another example is the 2D hand tracking and1D handheld trackball input that combine into 3D input.



- *singular interpretation of redundant commands:*
  Redundant commands, that is, commands from one channel that can substitute for commands from a different input channel, are useful for giving the user a choice to pick the momentarily most convenient way to give an instruction. They are interpreted in exactly the same way, and in the case that multiple, mutually redundant commands are given they are to be treated as a single instruction. We currently have two cases of this style: "select picking tool for finger" achieves the same result as performing a dedicated hand posture while the hand is being tracked, and the 'release' gesture during object manipulation is equivalent to the "finished" speech command. For the second case we chose two seconds as an appropriate interval in which the mutually redundant commands are to be considered as one. In the first case we avoid such an arbitrary threshold implicitly by entering the picking mode, in which the two commands are not associated with a meaning.



- *sequential, moded interpretation:*
  This style does the opposite of redundant commands: it requires users to provide input first in one modality, then in another. This is a common style within the desktop metaphor – first a mouse click to give focus to a window, then keyboard interaction with that window – and has the drawback of an associated switching time between mouse and keyboard. In our system however there is no such switching time since the two involved modalities do not both use hands: The drawing and virtual object manipulation modes use gestures for spatial input and voice commands for mode selection. In fact, we chose this style because it makes the best use of each modality without creating a conflict.



Overall, the modalities work together seamlessly, allowing for interaction that has almost conversational character. Voice commands allow the user to easily switch features or tools on and off, and enter non-spatial input such as adding Items to Environments. Hand gestures make for a very natural input interface for spatial data and a few key hand postures enable their use to perform simple action sequences entirely with gestures. Finally, the trackball provides for exact, continuous 1-dimensional input for situations where hand gestures are less convenient or 3-dimensional input is desired. Table 1 summarizes how the four modalities are combined to control various application parameters. Multiple simple checkmarks in a row mean that redundant modalities can supply the input. Multiple encircled checkmarks stand for concurrent interpretation of the input channels.

control parameter	*speech*	*gesture*	*trackball*	*head pose*
non-dimensional:				
• take/save/discard snapshot	✓			
• tunnel mode (float, open, close…)	✓			
• add/remove viz to/from env.	✓			
• enter relocate mode	✓			
• end relocate/resize/orient modes	✓	✓	✓	
one-dimensional:				
• adjust focus region distance			✓	
• adjust focus region depth			✓	
two-dimensional:				
• pencil tool for finger		✓	✓	
• select virtual objects		⊘		⊘
three-dimensional:				
• view direction				✓
• position virtual objects		⊘	⊘	⊘
• resize virtual objects		⊘	⊘	⊘
• orient virtual objects		⊘	⊘	

Table 1: Various application parameters and which modalities control them. A certain application parameter can be controlled with one of the input modalities if the respective entry has a checkmark. In the case of multiple plain checkmarks in a row any *one* of many modalities can supply this input. If multiple encircled checkmarks are present, *every* modality's input contributes to the application parameter.

The benefit of multiple modalities is in fact three-fold: first, more diverse input modalities increase the chance that one of them allows a very natural way to perform a

certain action, such as the registered interaction with hand tracking. Second, redundancy allows the user to select the momentarily most convenient interface. Third, more input dimensions can be controlled simultaneously. The first two points are of particular importance in wearable computing where the computer is usually not in the foreground and the sole object of focus of attention. The third point is increasingly important as application complexities rise beyond what conventional input modalities can control.

System Description

Figure 9 shows a diagram of the structure of our system implementation, overlaid over pictures of the actual devices we use. We need two laptops simply for performance reasons: The gesture recognizer's performance drops dramatically if it has to compete with other compute- or bus communication-intensive applications on the same machine (a 1.1GHz PIII running Windows XP and DirectShow as the video processing infrastructure). The other laptop runs a custom-built OpenGL-based visualization engine which is described below. Sony Glasstron LDI-A55 glasses serve as the base for our head-worn unit, delivering monocular NTSC resolution in color. Mounted atop are a Point Grey FireFly color camera and an InterSense InertiaCube^2 orientation tracker. The COTS "newscaster-style" microphone is clipped to the side of the glasses. A tv-one CS-450 Eclipse scan converter overlays the luminance-keyed output from the rendering computer over the mostly unmodified video feed-through from the first computer, providing the input to the head-worn display and to a DV camera that we used to record the images shown in this article.

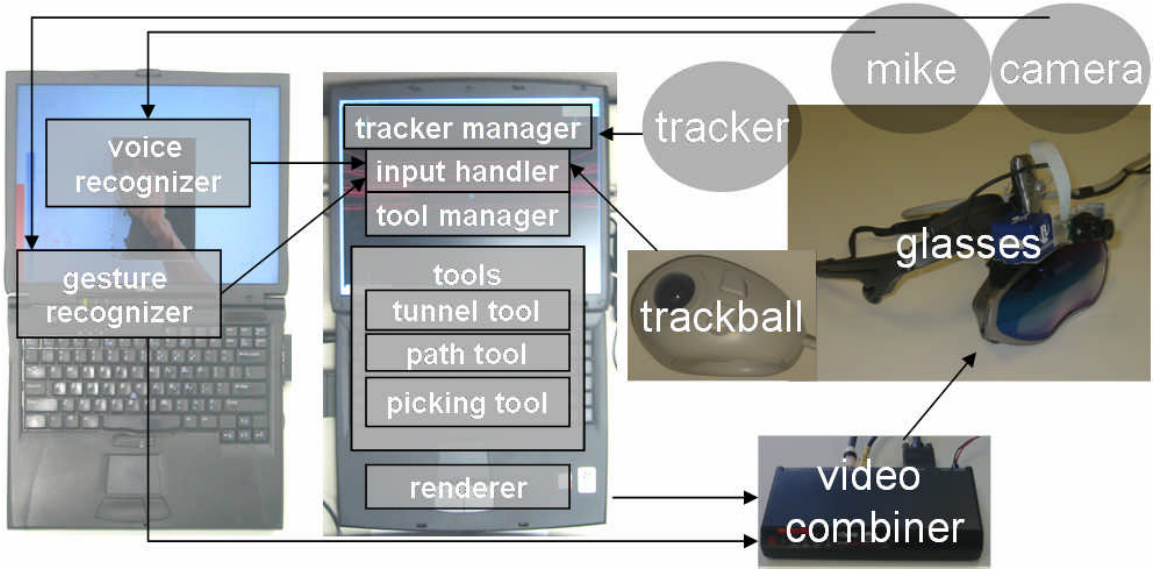


Figure 9: An overview of the hardware components we used to build the wearable augmented reality system and a functional diagram of the main software modules.

Visualization Engine

The visualization engine uses a simple scene graph to store data about each object. This data includes a tag specifying the Item or Items the object belongs to. Each Environment stores a record of its currently active Items that is used to determine whether an object should be rendered.

When the Tunnel Tool is not currently running, rendering is done by making a single pass over the scene graph to render the items currently selected into the Surroundings environment. Once the Tunnel Tool has been activated, rendering becomes a series of passes over the scene graph: first the items in the Surroundings Environment are rendered outside the tunnel, then the objects in the Tunnel Environment are fully rendered within the Focus Region, and finally objects within the Context Region are rendered in wireframe. Restricting rendering to objects in the tunnel is done with clipping planes defining the four outer walls of the tunnel and the front and back planes of the current region.

The Finger Environment is a special case, adding objects to the scene graph that will be rendered when the Surroundings or Tunnel environments are rendered, as appropriate.

Hand Gesture Recognition

This subsection briefly introduces the computer vision module *HandVu* that we used to implement hand gesture recognition. The overall architecture of HandVu is described to more detail in [KHT04], the robust hand detection in [KT04a,KT04c], and the *Flock of Features* tracking in [KT04b].
We use a combination of recently developed methods with novel algorithms to achieve real-time performance, accuracy, and robustness – all imperative to a user interface's quality and usability. A careful orchestration in multiple successive processing stages and automatic parameterization are largely responsible for the high speed performance. Multi-modal image cue integration (grey-level texture and local color information) alleviates interdependency problems with staged approaches, improves the robustness, and increases confidence in the results. The final output of the vision system is the hand's location in 2D image coordinates and possibly its posture – a classification into a set of predefined, recognizable hand configurations. It thus delivers zero-dimensional (postures) through two-dimensional (x-y location) output. The diagram in Figure 10 details the components of our vision system and their interactions.

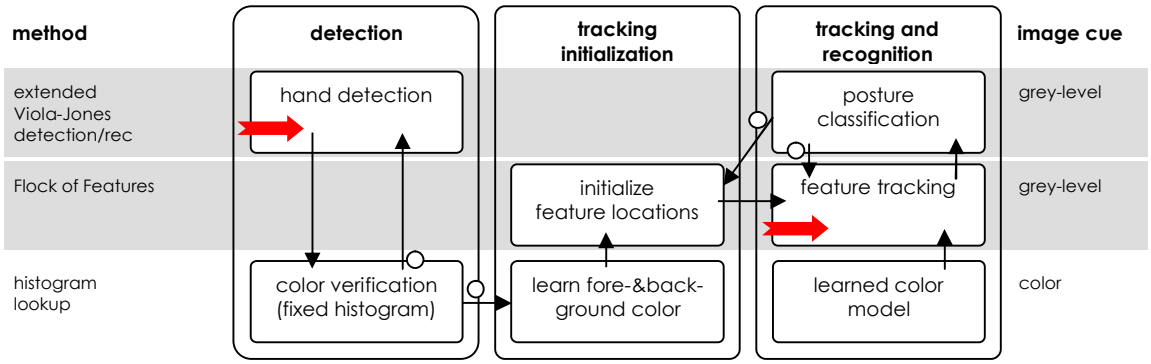


Figure 10: Components of our vision-based gesture recognition system. Until a hand was detected, new video frames are first processed in the "hand detection" module (left big arrow). It then makes its way through the system along the arrows, where + and – signal a successful and unsuccessful operation, respectively. One the hand is tracked, new video frames start out at the "feature tracking" module (right big arrow) and traverse the module graph from there.

For **hand detection**, we customized an object detection method recently proposed by Viola and Jones [VJ02] to detect a single hand posture, seen from a single view direction – a restriction that allows much improved speed and accuracy. The detection runs with about 10fps on a 200x300pixel sized area (in a 640x480 video stream) on a 1GHz laptop. The initial hand pose is a top-down view of the flat hand with the fingers touching each other (see Figure 11). The head-worn display shows an iconic hand to signal its readiness for gesture input (see for example in Figure 3). We chose this posture/view combination due to its highly identifiable nature against background noise and therefore its good success rate as a fail-safe detection condition [KT04a,KT04c]. This appearance based method yields a detection rate of 92.23% on a validation set.
Upon detection of a hand area, it is tested for the amount of skin-colored pixels it contains. We employ a histogram-based statistical model in HSV space, built from a large collection of hand-segmented pictures from many imaging sources (similar to Jones and Rehg's approach [JR02]). Only if the amount of skin pixels within the area exceeds a threshold is the hand detection considered successful. In practice, the two image cues combined yielded only a few false detections occurred per hour video.

The **tracking initialization** stage serves as an initialization to the third stage and is repeatedly executed after every hand detection and successful posture recognition. It first refines the user-independent statistical model of skin color by learning the observed hand color from the match area. This color histogram is contrasted to a reference area in the image that is assumed to not contain skin areas (a horseshoe-shaped area around and in front of the detected hand. The left image in Figure 11 shows the result of backprojecting the color model into parts of the image; that is, drawing in black every pixel that has a less than 50% chance of being of skin color. Next, approximately twenty KLT features (named after Kanade, Lucas, and Tomasi, see [ST94]) are placed randomly on skin-colored spots in the detected area. These features' image areas can be matched efficiently to the most similar areas, respectively, in the following video frame (see [LK81]).

Before a hand has been detected in the video stream, all incoming image frames are first processed in the detection module. Once the hand is being tracked, new frames arrive at the "feature tracking" module, part of the **tracking and recognition** stage. There, a *Flock of Features,* a method that we recently proposed for tracking articulated objects such as hands [KT04b], is employed to locate the hand in the current frame: since KLT features do not encode object-level knowledge or other global information, a global constraint on the features' relative locations is applied. This helps to achieve consistency among the features, to improve tracking across changing backgrounds, and to better deal with hand articulations and lighting changes. The Flock of Features algorithm removes features for which no good match between the frames can be established from the set and re-initializes them at positions that have a high skin color probability and are within a certain range of the other features' centroid. In the course of this, each feature location is weighted with its probability to be skin-color as predicted by the learned foreground/background color model. This method leads to a very natural multi-cue integration – grey-level texture with texture-less color information – to improve feature movement. The hand location is defined as the feature centroid. The accuracy of the centroid with respect to some fixpoint on the hand can not be guaranteed because of the entirely object-independent tracking method. While centroid might drift over time, this measure allows for a high precision; even minute movements of the hand are tracked and evaluated correctly. This technique is responsible for HandVu's ability to not loose tracking despite the hand being a highly articulate object whose appearance can change vastly and rapidly. Two images with active hand tracking and verbose output turned on are shown in Figure 11.

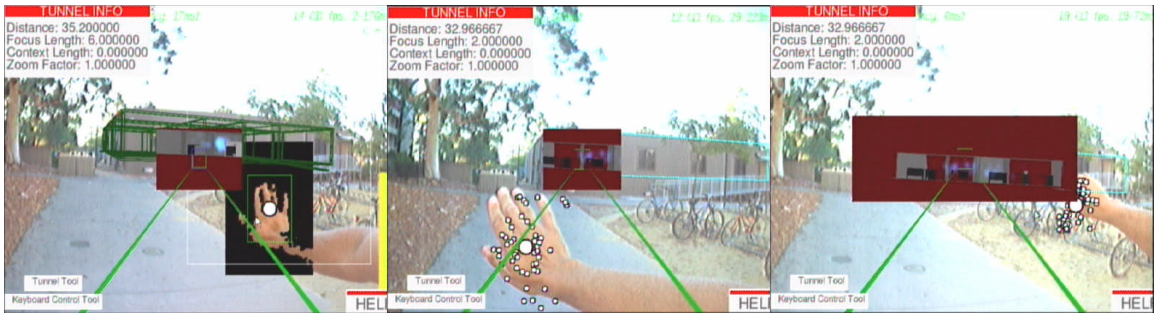


Figure 11: Verbose output from HandVu: a faint dark-green box around the hand in the left image indicates the area in which the hand was detected (not to be confused with the wireframe of the trailer). In the center image, each feature of a Flock of Features is shown as a little dot, their average is the big dot. In the right image, the user is resizing the Tunnel Tool's area with his tracked hand.

We currently **recognize** three postures (see Figure 6, plus a pointing gesture) for input signals as this is sufficient for our intentions. However, we can distinguish a total of six postures and we are considering increasing the substitutability in input modalities by designating more discrete postures to perform the same tasks as some voice commands. A two-level hierarchical Viola-Jones style detector first finds images areas that might show any of the six postures. Only those candidates are tested for each of the postures individually. A first experiment was conducted and showed that a recognized posture was correct in 93.76%. A formal evaluation is in preparation.

HandVu's vision processing methods typically require less than 100ms combined processing time per frame, plus about 20-30ms for lens distortion correction that might be desired for accurate registration. This is well below the threshold of 300ms for when interfaces start to feel sluggish, might provoke oscillations, and cause the "move and wait" symptom [SF63]. Hand tracking is performed for every frame with our 15Hz camera, while hand detection and posture recognition dynamically decide whether to process a frame or not. In comparison to other mobile VBIs, our method is more responsive and allows faster hand movements (for example, than the Hand Mouse [KOKS01], judging from a video available from their web site).

Robustness is the method's ability to deal with different environmental conditions. Our methods are mostly invariant to lighting and the automatic image quality adjustments of digital cameras, but, if it supports a certain common exposure interface, we automatically correct for extremely over- or under-exposed hand appearances. Shadows cast onto the hand and rapid color changes however impact detection and potentially tracking. Our methods are largely user independent and are not negatively influenced by different cameras or lenses. Hand occlusions are not detected or modeled explicitly. Yet, brief occlusions with foreign objects or the other hand do generally not cause all KLT features to be lost and tracking might continue successfully.

Results from HandVu are sent as **Gesture Events** to clients that are connected to the event server. This is a unidirectional stream of ASCII events in the following format:
1.2 tstamp id: t, r, "posture" (x, y) [s, a]\r\n
where

1.2	is the protocol version number,
tstamp	is a long integer timestamp of the respective image capture time, in milliseconds starting with the first seen frame,
id	is an identifier for which object this event belongs to, currently fixed to 0,
t	is 1 if the object is being tracked, 0 otherwise,
r	is 1 if one of the key postures was recognized, 0 otherwise,
posture	is a string identifier of one of the six recognized postures, or the empty string "",
x, y	are the tracked location in relative image coordinates, the image origin is in the top left,
s, a	are currently unused but will eventually contain a scale identifier and a rotation angle.

A DLL provides access to the captured frame through a memory-mapped file.

3D hand tracking would not be the same as interpreting the trackball's rolling motions as the third dimension: The actual distance of the hands would not allow interaction with far objects, scaling would trade off range for precision, and for all other mappings a "clutching" mechanism would have to be provided [Mac95].

Speech Recognition

We use a prototype automatic speech recognition library (ASRlib), provided to us by the Panasonic Speech Technology Laboratory. ASRlib is targeted towards computationally very efficient, speaker-independent recognition of simple grammars. We use the English dataset, about 70 keywords, and a grammar that allows around 300 distinct phrases with these keywords. Command phrases must be preceded and followed by a brief pause in the speech, but the words can be concatenated naturally. It performed well in our tests, not producing any false positives despite listening in on all of our conversation, but sometimes requiring the repetition of a command. A subset of the grammar is shown below; upper-case words are keywords.

<table>
<tr><td rowspan="8">command:
ADD viz TO environment
| REMOVE viz FROM environment
| SELECT tool TOOL FOR FINGER
| OPEN TUNNEL
| TUNNEL FLOAT
| TUNNEL FOLLOW FINGER
| TAKE SNAPSHOT
| INSERT insertable
| SAVE | DISCARD | FINISHED
| HIBERNATE | WAKE UP
| SET PATH FROM location TO location
| RESIZE OBJECT | MOVE OBJECT</td><td>viz:
BUNDLE bundle | bundle | ITEM item | item</td></tr>
<tr><td>bundle:
EMPTY | SKELETON | PATH
| NETWORKING | PHONE INFRASTRUCTURE</td></tr>
<tr><td>item:
INSUFFICIENT WIRELESS | TEMPERATURE
| BUILDING WIREFRAME | PHONE WIRES</td></tr>
<tr><td>environment:
SURROUNDINGS | TUNNEL | FINGER</td></tr>
<tr><td>location:
HERE | TUNNEL | POINTER</td></tr>
<tr><td>tool:
PICKING | PENCIL</td><td>insertable:
BOX | SPHERE | LINE</td></tr>
</table>

Table 2: The major parts of the recognized speech commands.

The speech recognition module consumes few enough resources to run aside the power-hungry computer vision application in its own process. The interface to the rendering application consists of unidirectional speech events: newline-terminated strings of recognized phrases, sent via a socket connection. The events are textually parsed again on the receiving side and converted into the rendering engine's internal event format. The socket detour and double parsing were chosen for improved modularization.

Conclusions

Wearable computers offer new applications to new fields of deployment. We have shown how a novel, versatile visualization aid for augmented reality environments can be controlled with a multimodal interface in a mobile scenario. Late modality integration in different styles permits every task and subtask to be operated by the most natural and

effective input modality. For atomic actions we employ mostly speech recognition. One-dimensional input is facilitated with a “ring trackball”, the device that was the most convenient to use out of a set of devices that we investigated.
Two-dimensional input, for example for picking virtual objects, is provided by registered hand pointing, a more natural interface than a mouse or trackball could offer. Three-dimensional input however is difficult with strictly registered hand input due to the depth of field of our rendered scene. Thus, the third dimension is controlled with the trackball, which is especially suitable because of its potentially infinite range (one can roll the ball forever forward). All the while, spoken commands control the system state through input concurrent with or in sequence with other modalities.
The Tunnel Tool enhances visual exploration of dense data distributions. Visualizing the insides of filled volumes is otherwise difficult, let alone in a wearable scenario. Yet its selective visualization paradigm should be of interest to a diverse group of people and professions. Furthermore, it shows that the combination of head tracking and speech input enables the use of effective visualization techniques without the need to hold any device in the hands.
We conclude that multimodal user interfaces for wearable computers enhance the devices’ applications and that they achieve respectable usability even for demanding application interfaces.

Acknowledgements

Panasonic Speech Technology Lab

References

[BH04] Ryan Bane, Tobias Höllerer. Interactive Tools for Virtual X-Ray Vision in Mobile Augmented Reality. Currently under review. http://www.cs.ucsb.edu/~holl/BaneHollerer04.pdf

[BHF02] Blaine Bell, Tobias Höllerer, and Steven Feiner. An annotated situation-awareness aid for augmented reality. In Proc. ACM UIST 2002 (Symp. on User Interface Software and Technology), pages 213-216, Paris, France, October 27-30 2002.

[Bol80] R. A. Bolt. “put-that-there”: Voice and gesture at the graphics interface. Computer Graphics (SIGGRAPH '80 Proceedings), 14(3):262-270, July 1980.

[BSLJ03] H. Brashear, T. Starner, P. Lukowicz, and H. Junker. Using Multiple Sensors for Mobile Sign Language Recognition. In IEEE Intl. Symposium on Wearable Computers (ISWC), Oct 2003.

[BSP+93] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony DeRose. Toolglass and Magic Lenses: The see-through interface. In James T. Kajiya, editor, Computer Graphics (SIGGRAPH '93 Proceedings), volume 27, pages 73-80, August 1993.

[CJM+98] P. Cohen, M. Johnston, D. McGee, S. Orviatt, J. Pittman, I. Smith, L. Chen, and J. Clow. QuickSet: Multimodal interaction for distributed applications. In Proceedings of The Fifth ACM International Multimedia Conference (MULTIMEDIA '97), pages 31-40, New York/Reading, November 1998. ACM Press/Addison-Wesley.

[CW99] Yuntao Cui and Junyang Weng. A Learning-Based Prediction and Verification Segmentation Scheme for Hand Sign Image Sequence. Transactions on Pattern Analysis and Machine Intelligence, pages 798-804, 1999.

[DUS01] Klaus Dorfmüller-Ulhaas and Dieter Schmalstieg. Finger Tracking for Interaction in Augmented Environments. In IFAR, 2001.

[FMHW97] S. Feiner, B. MacIntyre, T. Höllerer, and T. Webster. A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment. In Proc. First Int. Symp. on Wearable Computers, October 1997.

[FWK+95] S. Feiner, A. Webster, T. Krueger, B. MacIntyre, and E. Keller. Architectural anatomy. Presence, 4(3):318-325, Summer 1995.

[HFP99] Tobias Höllerer, Steven Feiner, and John Pavlik. Situated documentaries: Embedding multimedia presentations in the real world. In Proc. ISWC '99 (Third Int. Symp. on Wearable Computers), pages 79-86, San Francisco, CA, October 18-19 1999.

[HFT+99] Tobias Höllerer, Steven Feiner, T. Terauchi, G. Rashid, and Drexel Hallaway. Exploring MARS: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System. Computers and Graphics, 23(6):779-785, December 1999.

[JLB+00] Simon Julier, Marco Lanzagorta, Yohan Baillot, Lawrence Rosenblum, Steven Feiner, Tobias Höllerer, and Sabrina Sestito. Information filtering for mobile augmented reality. In Proc. ISAR '00 (Int. Symposium on Augmented Reality), pages 3-11, Munich, Germany, October 5-6 2000.

[JR02] M. J. Jones and J. M. Rehg. Statistical Color Models with Application to Skin Detection. Int. Journal of Computer Vision, 46(1):81-96, Jan 2002.

[KCD+01] Gudrun Klinker, Oliver Creighton, Allen H. Dutoit, Rafael Kobylinski, Christoph Vilsmeier, and Bernd Brügge. Augmented maintenance of powerplants: a prototyping case study of a mobile AR system. In Proc. ISAR '01 (Int. Symposium on Augmented Reality), pages 124-133, New York, NY, October 29-30 2001.

[KHT04] M. Kölsch, M. Turk, and T. Höllerer. Vision-Based Interfaces for Mobility. In Proc. Intl. Conference on Mobile and Ubiquitous Systems (Mobiquitous), August 2004.

[KKS02] Takekazu Kato, Takeshi Kurata, and Katsuhiko Sakaue. VizWear-Active: Towards a Functionally-Distributed Architecture for Real-Time Visual Tracking and Context-Aware UI. In Proc. IEEE Intl. Symposium on Wearable Computers, pages 162-163, 2002.

[KOKS01] Takeshi Kurata, Takashi Okuma, Masakatsu Kourogi, and Katsuhiko Sakaue. The Hand Mouse: GMM Hand-color Classification and Mean Shift Tracking. In Second Intl. Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems, July 2001.

[KOM+03] E. Kaiser, A. Olwal, D. McGee, H. Benko, A. Corradini, X. Li, S. Feiner, and P. Cohen. Mutual dissambiguation of 3d multimodal interaction in augmented and virtual reality. In Proc. ICMI 2003 (Fifth International Conference on Multimodal Interfaces), pages 12-19, Vancouver, BC, November 5-7 2003.

[KOR+02] D. M. Krum, O. Omoteso, W. Ribarsky, T. Starner, and L. F. Hodges. Speech and gesture multimodal control of a whole earth 3D visualization environment. In Proc. VISSYM 2002 (Joint Eurographics -- IEEE TCVG Symposium on Visualization), pages 195-200, 2002.

[KST93] David B. Koons, Carlton J. Sparrell, and Kristinn R. Thorisson. Integrating simultaneous input from speech, gaze, and hand gestures. In Mark T. Maybury, editor, Intelligent Multimedia Interfaces, pages 257-276. AAAI Press | MIT Press, Menlo Park, CA, 1993.

[KT04a] Mathias Kölsch and Matthew Turk. Analysis of Rotational Robustness of Hand Detection with a Viola-Jones Detector. In IAPR International Conference on Pattern Recognition, 2004.

[KT04b] Mathias Kölsch and Matthew Turk. Fast 2D Hand Tracking with Flocks of Features and Multi-Cue Integration. In IEEE Workshop on Real-Time Vision for Human-Computer Interaction (at CVPR), 2004.

[KT04c] Mathias Kölsch and Matthew Turk. Robust Hand Detection. In Proc. IEEE Intl. Conference on Automatic Face and Gesture Recognition, May 2004.

[LK81] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In Proc. Imaging Understanding Workshop, pages 121-130, 1981.

[Mac95] I. S. MacKenzie. Input devices and interaction techniques for advanced computing. In W. Barfield and T. A. Furness III, editors, Virtual environments and advanced interface design, pages 437-470. Oxford University Press, 1995.

[Man97] Steve Mann. Smart clothing: The wearable computer and wearcam. Personal Technologies, 1(1), March 1997.

[ME02] Tim Morris and Osama S. Elshehry. Hand segmentation from live video. In The 2002 Intl. Conference on Imaging Science, Systems, and Technology, UMIST, Manchester, UK, 2002.

[PSOS98] Indrajit Poddar, Yogesh Sethi, Ercan Ozyildiz, and Rajeev Sharma. Toward natural gesture/speech hci: A case study of weather narration. In Proc. PUI 1998 (Workshop on Perceptual User Interfaces), pages 1-6, November 1998.

[PT01] Wayne Piekarski and Bruce Thomas. Tinmith-metro: New outdoor techniques for creating city models with an augmented reality wearable computer. In Proceedings of the Fifth International Symposium on Wearable Computers, October 2001.

[RS01] Gerhard Reitmayr and Dieter Schmalstieg. Mobile collaborative augmented reality. In Proc. ISAR '01 (Int. Symposium on Augmented Reality), pages 114-123, New York, NY, October 29-30 2001.

[SAAG00] Thad Starner, Jake Auxier, Daniel Ashbrook, and Maribeth Gandy. The Gesture Pendant: A Self-illuminating, Wearable, Infrared Computer Vision System for Home Automation Control and Medical Monitoring. In International Symposium on Wearable Computers, 2000.

[SF63] T.B. Sheridan and W.R. Ferrell. Remote Manipulative Control with Transmission Delay. IEEE Transactions on Human Factors in Electronics, 4:25-29, 1963.

[SF96] David Saxe and Richard Foulds. Toward robust skin identification in video images. In Proc. IEEE Intl. Conference on Automatic Face and Gesture Recognition, pages 379-384, Sept. 1996.

[SK98] J. Segen and S. Kumar. GestureVR: Vision-Based 3D Hand Interface for Spatial Interaction. In The Sixth ACM Intl. Multimedia Conference, September 1998.

[SL00] Riku Suomela and Juha Lehikonen. Context compass. In Proc. ISWC '00 (Fourth Int. Symp. on Wearable Computers), pages 147-154, Atlanta, GA, October 16-17 2000.

[SLS01] Riku Suomela, Juha Lehikoinen, and Ilkka Salminen. A system for evaluating augmented reality user interfaces in wearable computers. In Proc. ISWC '01 (Fifth Int. Symp. on Wearable Computers), pages

77-84, Zürich, Switzerland, October 8-9 2001.

[SS98] Nitin Sawhney and Chris Schmandt. Speaking and listening on the run: Design for wearable audio computing. In Proc. ISWC '98 (Second Int. Symp. on Wearable Computers), pages 108-115, Cambridge, MA, October 19-20 1998.

[ST94] Jianbo Shi and Carlo Tomasi. Good features to track. In Proc. IEEE Conference on Computer Vision and Pattern Recognition, Seattle, June 1994.

[TDP+98] B. Thomas, V. Demczuk, W. Piekarski, D. Hepworth, and B. Gunther. A wearable computer system with augmented reality to support terrestrial navigation. In Proc. ISWC '98 (Second Int. Symp. on Wearable Computers), pages 168-171, Pittsburgh, PA, October 19-20 1998.

[TP02] B. H. Thomas and W. Piekarski. Glove Based User Interaction Techniques for Augmented Reality in an Outdoor Environment. Virtual Reality: Research, Development, and Applications, 6(3), 2002.

[TRT03] Carlo Tomasi, Abbas Rafii, and Ilhami Torunoglu. Full-Size Projection Keyboard for Handheld Devices. Communications of the ACM, 46(7):70-75, July 2003.

[TvdM96] Jochen Triesch and Christoph von der Malsburg. Robust Classification of Hand Postures against Complex Backgrounds. In Proc. IEEE Intl. Conference on Automatic Face and Gesture Recognition, October 1996.

[VJ02] Paul Viola and Michael Jones. Robust Real-time Object Detection. Int. Journal of Computer Vision, 2002.

[WG89] D. Weimer and S. K. Ganapathy. A synthetic visual environment with hand gesturing and voice input. In Proc. ACM CHI'89 Conference on Human Factors in Computing Systems, pages 235-240, 1989.