GWindows: Robust Stereo Vision for Gesture-Based Control of Windows

Andrew Wilson Nuria Oliver
Microsoft Research
One Microsoft Way
Redmond, WA
awilson@microsoft.com nuria@microsoft.com

ABSTRACT

Perceptual user interfaces promise modes of fluid computer-human interaction that complement the mouse and keyboard, and have been especially motivated in non-desktop scenarios, such as kiosks or smart rooms. Such interfaces, however, have been slow to see use for a variety of reasons, including the computational burden they impose, a lack of robustness outside the laboratory, unreasonable calibration demands, and a shortage of sufficiently compelling applications. We address these difficulties by using a fast stereo vision algorithm for recognizing hand positions and gestures. Our system uses two inexpensive video cameras to extract depth information. This depth information enhances automatic object detection and tracking robustness, and may also be used in applications. We demonstrate the algorithm in combination with speech recognition to perform several basic window management tasks, report on a user study probing the ease of using the system, and discuss the implications of such a system for future user interfaces.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Input devices and strategies*; I.4.9 [**Image Processing and Computer Vision**]: Applications

General Terms

Algorithms, Design

Keywords

computer vision, gesture recognition, speech recognition, computer human interaction

1. INTRODUCTION

Perceptual user interfaces (PUIs) use alternate sensing modalities to replace or complement traditional mouse and keyboard input. For example, video cameras may be used to sense the presence of a user, track the user's hands to control a cursor or perform commands with gestures, in concert with speech recognition processes. Often the goal of such research is for the system to simulate natural modes of interaction, as in conversational interfaces [10]. At the same time and in the near term, we are faced with a variety of rather more mundane, specialized devices and applications that do not have the traditional mouse and keyboard interface, including TabletPCs, media-center PCs, kiosks, hand-held computers, home appliances, video-games, and wall-sized displays. In these scenarios, PUIs offer to replace the more traditional interaction modalities. PUIs may also add value by complementing traditional interfaces, by providing an alternate channel for interaction, such as using voice to communicate with an intelligent assistant [9] while working on a project or dismissing a notification while working on a primary task. Perceptual modalities can also be valuable in scenarios in which the mouse and keyboard are clumsy and require more effort than they should (e.g., adjusting the volume on the media player). Finally, perception-based interaction can be leveraged to assist disabled users who have lost the fine control of hand musculature.

Unfortunately most examples of PUIs are still quite fragile; these systems often are based on techniques sensitive to unique environmental circumstances (e.g. color models that highly depend on the lighting conditions), rely on the use of multiple CPUs or specialized hardware, are usually installed and maintained in very limited quantities, and require laborious calibration. We believe that for these novel interfaces to be adopted, they must perform robustly outside of the laboratory, be computationally inexpensive, rely on common hardware, and be easy to set up and calibrate. Also, they cannot rely on intrusive devices such as gloves, headsets or close-talk microphones.

In this paper, we propose a real-time stereo vision algorithm for PUIs that is designed with these constraints in mind. We review an application of the algorithm in a multimodal system, named GWINDOWS, that allows users to manipulate on-screen objects with gestures and voice.

2. RELATED WORK

Work on PUIs draws on wide variety of fields, including signal processing, user interface design, computer vision, speech processing and behavior modeling. Here we limit ourselves to considering PUIs used to interact with on-screen interfaces.

ICMI'03, November 5–7, 2003, Vancouver, British Columbia, Canada.

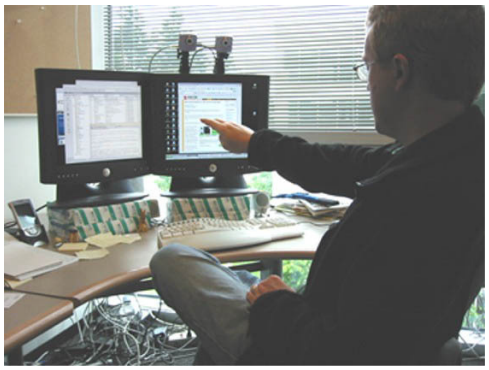


Figure 1: Perceptual interfaces enable "casual" and "10 foot" interfaces in scenarios where mice and keyboards are not appropriate or available.

Many perceptual interface systems have been developed for intelligent room systems. For example, the ALIVE system [17] used computer vision to track the users as they moved about the room. The system had limited gesture recognition abilities, which allowed the user to interact with a character on a large wall display.

In [6] Freeman and Weissman used computer vision techniques to find the user's open hand from across the room. Their system was then applied to controlling a television. Seated on a couch, the user could manipulate a graphical icon of a hand on-screen. To change the volume, the user moved the hand onto an on-screen slider. The authors found the feedback of the hand to be very effective in assisting the user.

The potential of manipulating on-screen objects with hand gestures sensed with computer vision is explored in [15] by Kjeldsen. Gross movement was used for pointing, and the hand shape was used to select commands. Kjeldsen highlighted the difficulties in constructing systems that meet users' expectations for responsiveness, particularly in pointing tasks. Users found arbitrary mappings between gestures and commands difficult to learn and remember. Difficulties with responsiveness and accuracy lead to the conclusion that such interfaces are more appropriate for selecting and manipulating large on-screen objects. Finally, new users became fatigued easily.

A variety of systems have been created to explore the use of head motion to control on-screen interaction. For example, Bérard [2] used a lightweight tracking system to detect fine head movement, and used the system to control the viewpoint as well as moving on-screen objects. As in the present work, Bérard used a very simple, fast and robust technique that resulted in very responsive system.

In our work on GWindows, we have implemented a real-time stereo vision system with the ability to sense the user's hand positions. Stereo vision has a long history in the field of computer vision, and has been applied in various PUIs. For example, the stereo system presented in [1] used two cameras and a skin color model to find the position of the user's head and hands. This was applied to a variety of interaction scenarios where the user, seated in front of a large display, manipulated objects on screen. In [11] dedicated stereo hardware is used to match a three dimensional articulated model of the user. This model was then used to recover broad pointing motions, which could be used to point at on-screen objects. Stereo disparity and optical flow information are combined in [23] to follow the head and hands of the user. The authors claim that the system can discriminate the face of the user, monitor the basic movements, and smoothly learn an object presented by the user, and communicate with users from hand signs learned in advance.

We also integrate speech commands into GWindows. There has been much interest in developing human-computer interfaces that allow the use of speech and gesture. It has been shown by Mignot et al. that gestures and speech are two complementary modalities: gestures are normally used to indicate objects and spots in the screen, as well as simple moves, whereas speech is used for specifying more abstract notions, actions or relations [18]. The authors also noted that multimodal commands are less ambiguous than purely oral or gestural ones. They conclude that spoken natural language associated with unconstrained 2D gestures or direct manipulations is a promising communication mode for users interacting either with standard software or with 'intelligent' systems. The paper is structured as follows: In section 3, we describe our approach to sensing the user's hand position. Section 4 shows this system applied to the task of manipulating a GUI and examine the performance of this system in a user study in section 5. Lastly, we discuss in section 6 various extensions of this system, including implications for gesture analysis and two-handed interaction.

3. COMPUTER VISION ALGORITHM

An important problem in using computer vision for PUIs is the automatic real-time detection and tracking of relevant objects in the scene. In many applications we would like to be aware of the presence of the user, the user's location, and the position of the user's head and hands.

For ease of deployment and robustness of operation we prefer detection and recognition methods that make as few assumptions as possible about the environment and the specific appearance of the objects of interest, e.g. hands. Secondly, we would like to use computationally inexpensive techniques so that the system does not prohibit the user from performing other tasks on the same CPU. Lastly, we require that the system be sufficiently responsive so that user's experience is fluid.

Our algorithm uses simple, fast techniques to track multiple objects moving in the scene, and relies on domain specific constraints to determine the true object of interest. One advantage of this multiple hypothesis approach to tracking is that we may use a simple, fast, and imperfect tracking algorithm and rely on the fact that if a tracker fails, another may still be following the object of interest.

3.1 Image Motion to Focus Attention

To initially find potential objects of interest, our algorithm finds regions of the image which exhibit motion. This exploits the observation that our own attention is often drawn to moving objects [12, 5]. Motion in the image is detected by comparing a patch of the current image centered about a given location to a patch at the same location in the previous image. To compare image patches, we use sum of absolute differences (SAD) over square patches in two images. For a patch from image I_1 centered about image location (u_1, v_1) and a patch in I_2 centered about (u_2, v_2), we define the im-

age comparison function $SAD(I_1, u_1, v_1, I_2, u_2, v_2)$ as

$$\sum_{-\frac{D}{2} \leq i,j \leq \frac{D}{2}} |I_1(u_1 + i, v_1 + j) - I_2(u_2 + i, v_2 + j)| \quad (1)$$

where $I(u, v)$ refers to the pixel at (u, v), D is the patch width, and the absolute difference between two pixels is the sum of the absolute differences taken over all available color channels. To find regions in the image with movement we find points (u, v) such that

$$SAD_{motion} = SAD(I_{t-1}, u, v, I_t, u, v) > \tau \quad (2)$$

where τ is a threshold. An object hypothesis is initiated for each such region. To limit computation, this test for image motion may be conducted on a sparse, regular grid on the image (e.g. every 16 pixels).

3.2 Frame to Frame Tracking

Once an object hypothesis has been initiated, the position of the object is updated at each time step by finding the patch in the current image which best matches the patch centered on the object at the previous image. We define

$$SAD_{movement} = SAD(I_{t-1}, u_{t-1}, v_{t-1}, I_t, u_t, v_t) \quad (3)$$

where (u_t, v_t) refers to the image location at time t. A simple frame to frame tracker finds (u_t, v_t) that minimizes $SAD_{movement}$. This simple block matching technique suffers from drift problems, where over time the tracker may begin following some part of the image off the intended object. To combat drift, we optimize both $SAD_{movement}$ and SAD_{motion} as a weighted sum. Intuitively, this combination uses *motion* to coarsely track the object as it moves, while using the frame-to-frame tracking to precisely "stick" on a given part of the moving object, as well as maintain the tracker when the object is not moving.

The tracking search is conducted over a small window (typically 10 pixels) around the predicted location of the object, assuming a linear dynamics model with noise (Kalman filter). Note that we use the term *movement* to imply a representation based on a discrete object and its location over time, while we use *motion* to refer to change in image intensity values in a given region of the image due to the movement of one or more objects.

If the average movement of an object falls below some threshold, it is eliminated as an object hypotheses. Furthermore, if the distance between a given object hypothesis and any other object hypothesis falls below a threshold (say, five inches in world coordinates), it is supposed that that the two hypotheses are redundant, and one of the two hypotheses is eliminated.

3.3 Object Depth

Binocular disparity is a primary means for recovering depth information from two or more images taken from different viewpoints. Given the 2D position of an object in two views, it is straightforward to triangulate to find the depth of the object [8].

If two cameras of focal length f are parallel to one another, the 3-d position (x, y, z) of the object may be computed from the positions of the object in images from both cameras, (u_l, v_l) and (u_r, v_r), by the perspective projection equations

$$u = u_r = f\frac{x}{z} \quad (4)$$

$$v = v_r = f\frac{y}{z} \quad (5)$$

$$d = u_r - u_l = f\frac{b}{z} \quad (6)$$

where the disparity d, or shift in location of the object in one view to the other, is related to the baseline b, the distance between the two cameras [8].

Typically disparity is computed by matching an image intensity pattern (patch) at a given location in the first image to its pair in the second image. Often this approach is used to compute a map which gives the depth in the scene at every location in the image. Computing such a depth map is very computationally intensive, and often requires dedicated hardware to run in real-time [14, 4].

To limit computation, we only compute binocular disparity at points within the image that correspond to object hypotheses. For a given point in the image, (u, v), we find the value of disparity d such that the sum of absolute differences over a patch in the right image I_r centered on (u, v) and a corresponding patch in the left image I_l centered over $(u - d, v)$ is minimal, i.e. d that minimizes

$$SAD_{disparity} = SAD(I_l, u - d, v, I_r, u, v). \quad (7)$$

Furthermore, with an estimate of the depth of the point from a previous time step, we may limit the search over values of d corresponding to a range of depth around the last known depth. This search may be further narrowed by computing a prediction of the object's new location from a Kalman filter.

Note that in this stereo matching process, we assume that both cameras are parallel (that is, their rasters are parallel). If we wish to recover the depth in real world coordinates, we must also know the distance between the pair of cameras (baseline). In practice, both calibration issues may be addressed automatically by fixing the cameras on a prefabricated mounting bracket, or semiautomatically by the user presenting objects at known depth in a short calibration routine. Lastly, we improve the accuracy of the transform to world coordinates by accounting for lens distortion effects with a static, pre-computed calibration for a given camera [24].

3.4 Selective Attention

Unlike many other computer vision algorithms, the algorithm does not rely on fragile appearance models such as skin color models or hand image classification schemes which are prone to break when environmental conditions change or when the system is confronted with a new user. Because the system does not rely on color models, for example, large changes in illumination conditions may be accommodated. The present system has been used successfully in extremely low light situations, including in an office with the lights turned off, where the only illumination is due to the display.

This robustness comes at a cost of relying on application constraints to determine which of multiple object hypotheses to select as the true object of interest. We believe that this is a valuable trade-off in many circumstances.

In some cases there is a natural criterion to adopt. For example, for a given application it may be reasonable to monitor only the objects closest to the cameras, while ignoring all others. In our hand-tracking application, if the

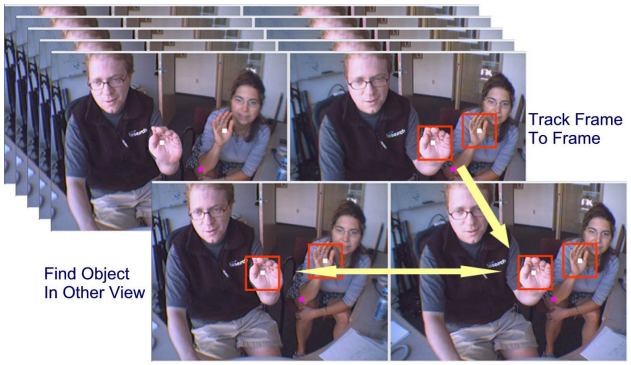
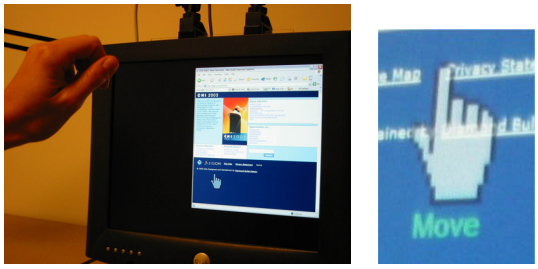


Figure 2: Object hypotheses (indicated by square solid colored dots on the image) are supported by frame to frame tracking through time in one view and stereo matching across both views



(a) (b)

Figure 3: The GWINDOWS system allows the user to select windows on the display. (a) Feedback regarding the user's hand position is provided by a hand icon which moves to follow the user's hand. (b) Any command mode in effect is indicated by drawing the name of the mode below the hand.

user is facing the cameras it is often the case that the object closest to the cameras is the hand. Another application may focus on objects that exhibit a particular quality of movement over time. Or a two-handed interaction application may select an object to the left of the dominant hand (for right-handers) as the non-dominant hand.

Figure 2 illustrates the 3-d tracking and 3-d depth computations. Note that all three steps, motion detection, tracking and depth computation, use the same sum of absolute difference function on image patches. This computation is easily optimized for single-instruction-multiple-data (SIMD) instructions, permitting a very fast implementation.

4. GWINDOWS

We have developed GWINDOWS, an application that allows users to conduct various window management tasks without the mouse and keyboard. The GWINDOWS interface extends the usual WIMP (windows, icons, mouse, pointer) interface, enabling users to "grab" a window with their hands and move it across their desktop, close, minimize, maximize windows and scroll the foreground window.

GWINDOWS was designed with the view that PUIs may be applied to everyday Graphical User Interface (GUI)-based computing tasks, and thereby the system may serve to introduce and evangelize perceptual interfaces to people otherwise unfamiliar with the notion that their computer is capable of sensing their activities and responding appropriately. Another motivation is to offer an alternative user interface to applications in which a keyboard and mouse are either undesirable or unavailable. For example, in the so-called "10 foot" user experience offered by media center PCs GWINDOWS-like systems may obviate or complement the IR remote control. Although GWINDOWS is rather conservative in its extension of the user experience (especially compared to conversational or agent-based interfaces), it is interesting to note that the recent sci-fi thriller *Minority Report*, set in the year 2054, shows the main character using a very elegant two-handed interface which relies on a similar sensing and interaction paradigm, particularly in how objects are picked up and moved on-screen.

Users explicitly initiate an interaction with GWINDOWS by moving their hand across a predefined "engagement plane", an invisible plane about twenty inches in front of the display, and parallel to the plane of the display. When the hand crosses the engagement plane, feedback is given to the user by drawing a large alpha-blended hand icon on the usual Windows desktop. This icon is distinct from the usual Windows cursor and can be viewed as an area cursor [22]. The engagement plane is placed such that the user's hands do not enter it during the usual use of the mouse and keyboard. When the system is "engaged", the hand icon is moved to reflect the position of the user's hand. This is illustrated in Figure 3. A similar scheme for hand position feedback was used in [6].

An open microphone used for speech recognition is placed in front of and below the display. The user may invoke one of a small set of verbal commands in order to act upon the current window under the hand icon. When an utterance is understood by the system, the token phrase is drawn along with the icon to give feedback that the speech recognition system understood the utterance.

The full functionality of GWINDOWS is as follows: (1) MOVE: By uttering "move" the user initiates the continuous movement of the window under the hand to follow the movement of their hand. Movement of the window is terminated when the user's hand is disengaged by moving behind the engagement plane, or when the user utters "release". (2) CLOSE, MINIMIZE, MAXIMIZE: By uttering "close", "minimize", or "maximize" the currently selected window is acted upon appropriately. (3) RAISE, SEND TO BACK: By uttering "raise", the selected window is popped to the foreground, while uttering "send to back" sends the selected window behind all other windows. (4) SCROLL: By uttering "scroll", the user initiates a scrolling mode on the current window, in which the rate of scrolling up and down on the window is proportional to how far above or below the hand is in relation to its position when scrolling mode was initiated, similar to functionality often obtained with mouse wheels. Scrolling is terminated when the user's hand is disengaged by moving behind the engagement plane, or when the user utters "release". A video figure of the system is available at `http://research.microsoft.com/~nuria/GWindows/GWindows.htm`.

When the user switches modes as described above, the user is given feedback by the appearance of the mode name displayed in green lettering under the hand icon, as Figure 3

(b) illustrates. In the case of using speech recognition, this mode label offers valuable feedback to indicate the success of the speech recognition process.

4.1 Implementation Details

Our implementation of GWindows uses the computer vision system described section 3, with two Firewire webcams acquiring 320x240 color images at a frame rate of 30Hz. The multiple hypothesis tracking system is configured to handle at most 6 trackers simultaneously. The system may use any number of simultaneous trackers. However, in our experience, 6 trackers seem to be sufficient for tracking the user's hands. Speech recognition is performed using Microsoft SAPI 5.1, with a simple command and control grammar and an inexpensive open microphone placed in front of and below the display. The computer vision module uses an MMX [21] implementation of the sum of absolute differences image function (Equation 1). The current system takes less than 15% of the CPU time on a 1GHz Pentium III.

The engagement and acquisition of the hand is implemented in the stereo vision system by simply looking for any object hypothesis with depth less than 20 inches. Any such hypothesis is considered the active hand in GWindows until it is moved behind the engagement plane, or when it is removed from the set of tracked object hypotheses, in which case the nearest remaining object hypothesis is selected.

5. PILOT USER STUDY

We performed a preliminary, qualitative user study to determine how everyday users of GUIs find using GWindows. In this study, we confirm Kjeldsen's observations on a related system in which he found users become adept at selecting and moving items on the display very quickly, while new users tend to tire easily holding up their hand [15]. Unlike Kjeldsen's system, however, we rely on a small set of speech commands rather than requiring the user to put their hands in specific configurations to change application function.

Eighteen people (eight women and ten men) participated in the experiment. They ranged in age from late 20s to early 40s; all were experienced computer users. Whereas all men worked in computer-science related fields, the women worked in the administrative or library-related fields.

The experiment was conducted on the implementation of GWindows described in the previous section, in which the keyboard and mouse were removed (see Figure 4). With no keyboard and mouse, they could only interact with the computer by hand motions and speech. The experimenter was seated behind the participants, with access to a second display, keyboard and mouse to open some Internet Explorer windows on the participant's display.

5.1 Procedure and design

Participants were tested individually in a single session that lasted ten to fifteen minutes. Each participant performed two kinds of tasks. After explaining the GWindows system, the experimenter demonstrated the engage/disengage interaction with the computer using GWindows. Finally she explained verbally and with examples each of the following commands: Close, Move, Raise, Send to Back and Scroll. The participant was then invited to freely interact with the computer using GWindows and to practice each of the commands.

Figure 4: Experimental Setup. Study participants were seated in front of a GWindows-enabled display and an open microphone for speech recognition.

After acknowledging proficiency with the system, the participant was asked three questions to be answered using the GWindows interface by manipulating five Internet Explorer windows, some of which contained the answers to the questions. These windows were placed on the display by the experimenter. The participants were asked to answer the questions in any order and without any time constraints. The questions were: *(1) What is the weather forecast for tomorrow? (2) What is playing at the* Crossroads 8 *cinemas? (3) What is the top story on the New York Times?*

The trial was considered successful if the participant was able to correctly find the answers to the three questions in the Internet Explorer windows. To do so, the participants had to Raise, Send to back, Move, Close and Scroll the windows. After completing this task, the participants were asked to answer a questionnaire with 31 Likert scale questions (where 1 corresponds to strongly agree and 5 to strongly disagree) about their experience using GWindows and their general attitude regarding perceptual interfaces.

5.2 Discussion

After three to five minutes of interaction with the system, all but one of the users were comfortable managing windows using GWindows. In this latter case, the participant had some difficulties adjusting to the new way of interacting with the desktop computer. In the question-answering task, some participants used a strategy of reordering the windows with Send to Back and Raise commands, while others preferred to move the windows to reveal the information they were looking for. All the participants successfully finished this task (i.e. correctly answered the three questions) in a time period of three to seven minutes. On rare occasions, if the system was not performing as expected, participants tended to move even closer to the display, or move their hands faster. Analogous to the Lombard effect in speech recognition [13], this change in behavior in the extreme tends to degrade performance.

Users tended to have difficulty with the speech recognition, which gave some errors, probably because it was not tuned to individual users. Many used "Stop" or "No" instead of "Release" to finish a Move or Scroll command. Some users found occasional jittering of the virtual hand troublesome. Other users were impressed by the tracking

ability. Some users occluded the computer screen with their hand. They found relatively quickly that they could change their body position to avoid this problem. Many users thought that GWINDOWS would be a good system for kiosk environments or at home, *i.e.* in 10 foot interfaces.

From the written survey (Likert scale 1 agree 5 disagree), users indicated that they enjoyed interacting with the computer using GWINDOWS and they were generally satisfied with their own performance. Participants imagined GWINDOWS being used in accessibility scenarios first (1.5 ± 0.15)[1], then to control their TV from across the room (2.1 ± 0.24) and finally to interact with a kiosk in a public place (2.2 ± 0.23). All had a very positive reaction to the video user interface that appears in the movie *Minority Report* (1.6 ± 0.1). GWINDOWS was rated as an intuitive way to manage windows (2.3 ± 0.15), but not particularly comfortable (3.2 ± 0.23). In particular, the participants found that their arm got tired after a while (2 ± 0.26). Participants rated the SCROLL command as the most difficult, possibly due to to the rate-control mechanism used in scrolling mode. Even though the speech recognition system was not found to be particularly reliable (2.7 ± 0.25), the participants enjoyed being able to use speech commands in the experiment (2.1 ± 0.17). They showed a slight preference for gestures instead of speech (2.7 ± 0.26), even though the version of GWINDOWS that they used provided no support for using gestures to invoke commands.

6. EXTENDING GWINDOWS

From the previous pilot user study we realized that, although people found GWINDOWS very easy to learn, fatigue and speech recognition errors were problematic. We believe that fatigue is due primarily to the fact that subjects had to raise their arm and maintain their arm position for some seconds in order to reach many regions of the screen and move the GWINDOWS hand icon with some degree of precision. Fatigue may be addressed partly by scaling the movement of the hand so that smaller movements are required to reach all parts of the screen, and by changing the configuration of the cameras such that they track object motion just above the keyboard, and therefore holding the hand in front of the display is no longer required.

To address the difficulties users had with speech recognition during the user study, we added functionality to GWINDOWS that permits most operations to be performed without the use of speech: in addition to using speech to initiate modes of interaction such as moving or raising windows, the user may operate GWINDOWS by using gestures. In this case, the user may select interaction modes by pausing or dwelling the hand over the target window. By dwelling a short amount of time (about 0.5 seconds), the target window is raised if its not already the topmost window. If the hand dwells a longer amount of time (about 1 to 1.5 seconds), the hand icon text then changes to "gesture", and the user may move the hand quickly left or right (a flick gesture) to send a window to the adjacent (left or right) monitor in a multi-monitor system. The system smoothly animates the movement of the window with a "swish" sound. If there is no adjacent monitor, then the window is minimized. If the user dwells even longer (about 2 seconds), the mode changes to the "Move" mode described previously. The user may exit the "Move" mode by pausing again. This change of interaction mode by dwelling relies on the continuous feedback of the mode label under the icon: a user simply pauses and dwells long enough until the desired mode is displayed. When the user then moves the hand, the system effects the mode's associated action (e.g., moving windows) and also exits the selection of modes.

Here we consider further ideas in extending GWINDOWS functionality.

6.1 Gesture Recognition

We have augmented the GWINDOWS interface with the ability to understand gesturing beyond simple pointing and movement. Our initial gesture recognition system recognizes dwell and left/right flick gestures. However, this is certainly a rich and somewhat unexplored research area that we are further investigating via more complex gestures.

Our initial experiments with dwell time and left/right flick gestures suggest that gesture recognition may be useful but requires careful design. Long et al. report in [16] that users often find gesture-based systems highly desirable, but they are also dissatisfied with the recognition accuracy of gesture recognizers. Furthermore, their experimental results show that users' difficulty with gestures is in part due to a lack of understanding of how the gesture system works. Long et al. highlight the ability of users to learn and remember gestures as an important design consideration. In light of these findings, we believe that one general approach is to standardize a small set of easily learned gestures, the semantics of which are determined by application context.

A small set of very simple gestures may offer significant bits of functionality where they are needed most. For example, dismissing a notification window may be accomplished by a quick gesture moving the hand from one side to the other, as in shooing a fly. Another example is gestures for "next" and "back" functionality found in web browsers, PowerPoint and other applications. A simple gesture-based navigation facility to web browsers may significantly reduce the time taken to carry out one of the most common actions in computer use: using the "back" button to return to previously visited pages, as reported by Moyle et al. in [19]. In their experiments users' subjective ratings showed a strong preference for the "flick" system, where the users would flick the mouse left or right to go back or forward in the web browser.

Selecting objects by clicking the mouse is an important functionality in today's GUIs. Presently, GWINDOWS does not enable the user to simulate mouse clicks by moving the hand. We are currently exploring techniques to select and click on on-screen objects such as hyper-links in a web browser. For example, a second, deeper, depth plane beyond the "engagement plane" can be used to trigger a click event. One challenge with this technique is to provide the user with adequate feedback to indicate where this "click plane" lies. An alternative technique is to compute the change in depth over time, such that fast forward motion triggers a click. Lastly, we may use speech recognition to detect when the user says "click", "select" or "go".

Even when mouse and keyboard are available, users may find it attractive to manipulate often-used applications while away from the keyboard, in what we call a "casual interface" or "lean-back" posture. Browsing email over morning

[1]The results are provided as the average value $\pm$ the standard deviation

coffee might be accomplished by mapping simple gestures to "next message" and "delete message". There are other circumstances where the user's hands might be dirty and gestures could provide a practical interface to the computer (e.g. reading email or reading some online recipe while cooking).

Finally, gestures may compensate for the limitations of the mouse when the display is several times larger than today's typical displays or in a multiple monitor situation. In such a scenario, gestures can provide mechanisms to restore the ability to quickly reach any part of the display, where once a mouse was adequate with a small display. Similarly, in a multiple display scenario it is desirable to have a fast, comfortable way to indicate a particular display. For example, in the current GWINDOWS system, the foreground object may be "bumped" to another display by moving the hand in the direction of the target display.

Note that in many cases the surface forms of these various gestures may remain the same throughout these examples, while the semantics of the gestures depends on the application at hand. Providing a small set of standard gestures eases problems users have in recalling how gestures are performed, and also allows for simpler and more robust signal processing and recognition processes.

6.2 Two-Handed, Mouse and Hand UI

Mice are particularly suited to fine cursor control, and most users have much experience with them. GWINDOWS can provide a secondary, coarse control that may complement mice in some applications. For example, in a map application, the user might cause the viewpoint to change with GWINDOWS, while using the mouse to select and manipulate particular objects in the view. GWINDOWS may also provide a natural "push-to-talk" or "stop-listening" signal to speech recognition processes. In [20] users were shown to prefer using a perceptual user interface for push-to-talk.

Our multiple hypothesis tracking framework allows for the detection and tracking of multiple objects. Thus we may consider tracking both hands for a two-handed interface. Studies show that people naturally assign different tasks to each hand, and that the non-dominant hand can support the task of the dominant hand [7]. Two-handed interfaces are often used to specify spatial relationships that are otherwise more difficult to describe in speech. For example, it is natural to describe the relative sizes of objects by holding up two hands, or to specify how an object (dominant hand) is to be moved with respect to its environment (non-dominant hand) [3].

6.3 Accessibility

GWINDOWS may find application where the user is unable to use a traditional keyboard and mouse. This includes accessibility scenarios in which users with limited manual dexterity are unable grasp the mouse or strike keys on a keyboard. Furthermore, because the techniques used in GWINDOWS do not rely on precise hand appearance models but instead rely on properties of motion and depth, the system may track any available object of sufficient size, including the tip of a Coke bottle, rolled up papers, and even the user's head. This ability to exploit a variety of available signals may be advantageous in many accessibility applications, where no two users may be alike, and some users may not have control of their hands at all.

In some applications the mouse and keyboard are unavailable due to the nature of the task at hand. For example, a worker involved in cleaning a chemical spill may be wearing a hazardous materials (HAZMAT) suit which prevents the use of a mouse and keyboard. A surgeon may wish to consult various images such as CT scans during a procedure while observing sterilization rules which preclude the use of a standard mouse. While cooking in the kitchen you might want to view an incoming email without dirtying the keyboard, or from the comfort of your easy chair change TV channels without finding the remote. Museum kiosks, gym treadmills and other public interfaces may also benefit from operation without keyboards and mice. In each of these specialized applications, the interaction may be simple enough that a system like GWINDOWS may be used exclusively.

7. CONCLUSION

As computers become more integrated in our daily lives, we can expect to find users in a wide variety of contexts where traditional mouse and keyboard interfaces are awkward, too intrusive, or unavailable. Perceptual user interfaces have the potential to fill new roles in user experiences opened by these new scenarios, but there is presently an unfulfilled need for lightweight, robust and responsive sensing algorithms. The stereo vision technique proposed in this paper enables fast and robust sensing of the user in depth, and provides a useful resource for building future perceptual user interfaces. We note that by making few assumptions other than consistency of an object's movement in depth, the technique itself is general enough to be applicable to a wide range of scenarios.

We have incorporated this technique into a prototype called GWINDOWS. GWINDOWS demonstrates the use of perceptual user interfaces (PUIs) in everyday GUI tasks. GWINDOWS has been shown to several hundred people at a number of demonstration venues, under varying lighting conditions and cluttered, moving backgrounds. Most users were able to pick up the system very quickly, and many were pleasantly surprised to find how responsive the system is. GWINDOWS was able to successfully detect and track the user's hand in nearly all cases.

We look forward to exploring extensions to GWINDOWS that highlight interaction scenarios in which perceptual user interfaces add value beyond traditional interfaces.

8. ACKNOWLEDGEMENTS

We thank the subjects participating in our experiments.

9. REFERENCES

[1] A. Azarbayejani and A. Pentland. Real-time self-calibrating stereo person tracking using 3-D shape estimation from blob features. In *Proceedings of 13th ICPR*, Vienna, Austria, August 1996. IEEE Computer Society Press.

[2] F. Berard. The pereptual window: head motion as a new input stream. In *IFIP conf. on human-computer interaction*, 1999.

[3] W. Buxton and B. Myers. A study in two-handed input. In *Proc. of CHI'86*, pages 321–326, 1986.

[4] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection, 1998.

[5] S. Franconeri and D. Simons. Moving and looming stimuli capture attention.

[6] W. T. Freeman and C. Weissman. Television control by hand gestures. In *Intl. Workshop on Automatic Face and Gesture Recognition*, pages 179–183, 1995.

[7] Y. Guiard. Assymetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behavior*, 19(4):486–517, 1987.

[8] B. Horn. *Robot Vision*. MIT Press, 1988.

[9] E. Horvitz. Principles of mixed-initiative user interfaces. In *Proc. of CHI '99*, 1999.

[10] E. Horvitz and T. Paek. A computational architecture for conversation. In *Proc. of the Seventh International Conference on User Modeling*, pages 201–210, 1999.

[11] N. Jojic, B. Brumitt, B. Meyers, and S. Harris. Detecting and estimating pointing gestures in dense disparity maps. In *Proceed. of IEEE Intl. Conf. on Automatic Face and Gesture Recognition*, 2000.

[12] J. Jonides and S. Yantis. Uniqueness of abrupt visual onset in capturing attention. *Perception and Psychophysics*, 43(4):346–354, 1988.

[13] J. Junqua. The lombard reflex and its role on human listeners and automatic speech recognizer. *J. Acoustic Soc. Amer.*, 93:510–524, 1993.

[14] T. Kanade. Development of a video-rate stereo machine. In *Proc. of ARPA Image Understanding Workshop (IUW'94)*, pages 549 – 558, 1994.

[15] F. Kjeldsen. *Visual Interpretation for Hand Gestures as a Practical Interface Modality*. PhD thesis, Department of Computer Science, Columbia University, 1997.

[16] J. A. Long, J. Landay, and L. Rowe. Implications for a gesture design tool. In *Proc. CHI'99*, pages 40–47, 1999.

[17] P. Maes, T. Darrell, B. Blumberg, and A. Pentland. The alive system: wireless, full-body interaction with autonomous agents. *ACM Multimedia Systems, Special Issue on Multimedia and Multisensory Virutal Worlds*, 1996.

[18] C. Mignot, C. Valot, and N. Carbonell. An experimental study of future 'natural' multimodal human-computer interaction. In *Proc. of INTERCHI93*, pages 67–68, 1993.

[19] M. Moyle and A. Cockburn. Gesture navigation: An alternative 'back' for the future. In *Proc. of CHI02*, pages 822–823, 2002.

[20] A. Oh, H. Fox, M. Van Kleek, A. Adler, K. Gajos, L. Morency, and T. Darrell. Evaluating look-to-talk: a gaze-aware interface in a collaborative environment. In *CHI'02*, pages 650–651, 2002.

[21] A. Peleg and U. Weiser. Mmx technology extension to the intel architecture. *IEEE Micro*, 16(4):42–50, 1996.

[22] A. Worden, N. Walker, K. Bharat, and S. Hudson. Making computers easier for older adults to use: area cursors and sticky icons. In *Proc. of Conf. on Human factors in computing systems*, pages 266–271, 1997.

[23] I. Yoda and K. Sakaue. Utilization of stereo disparity and optical flow information for human interaction. In *Proc. of ICCV'98*, 1998.

[24] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.