GWINDOWS: Towards Robust Perception-Based UI

Andrew Wilson and Nuria Oliver
Microsoft Research
One Microsoft Way
Redmond WA 98052
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 have tackled some of 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.**

I. INTRODUCTION

Perceptual user interfaces 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 [1]. In the near term, however, 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, videogames, and wall-sized displays. In these scenarios, perceptual interfaces offer to replace the more traditional interaction modalities. Perceptual user interfaces 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 [2] 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 perceptual user interfaces are still quite fragile; these systems often are based on 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.

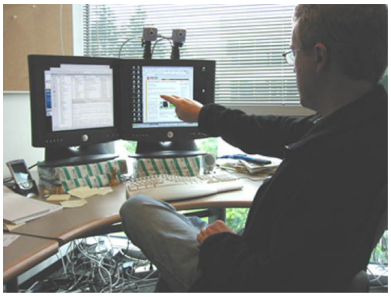


Fig. 1. Perceptual interfaces enable "casual" and "10 foot" interfaces in scenarios where mice and keyboards are not appropriate or available.

In this paper, we propose a real-time stereo vision algorithm for perceptual user interfaces 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.

II. RELATED WORK

Work on perceptual user interfaces draws on wide variety of fields, including signal processing, user interface design, computer vision, speech processing and behavior modeling [3]. Here we limit ourselves to considering perceptual user interfaces used to interact with on-screen objects.

Many perceptual interface systems have been developed for intelligent room systems. For example, the ALIVE system [4] 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. Though the user was free to move about the room, the need to attend to the display tended to limit the user's movements, which also limited the problem of detecting the user's gestures.

Perceptual user interfaces have been used to control household appliances. Freeman [5] used computer vision techniques to find the user's open hand from across the room. This was 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. Freeman found the feedback of the hand to be very effective in assisting the user.

Kjeldsen [6] demonstrates the potential of manipulating on-screen objects with hand gestures. In his system, movement of the hand was sensed with computer vision techniques. Gross movement was used for pointing, and the hand shape was used to select commands. After an initial learning curve, an experienced user could manipulate objects with speed and comfort comparable to other popular devices, while new users could be trained to select and move items accurately within minutes. The ability to interact directly with on-screen objects seemed to be more comfortable for some users than the indirect pointing used in a mouse or joystick. 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 [7] 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 users hand positions. Stereo vision has a long history in the field of computer vision, and has been applied in various perceptual user interfaces. For example, the stereo system presented in [8] used two precisely calibrated 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. Jojic [9] used dedicated stereo hardware 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. Yoda and Sakaue propose in [10] a system that simultaneously utilizes stereo disparity and optical flow information 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 that gestures and speech are two complementary modalities: Mignot et al. observe in [11] that gestures were normally used to indicate objects and spots in the screen, as well as simple moves, whereas speech was used for specifying more abstract notions, actions or relations. They also note 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 III, we describe our approach to sensing the user's hand position. Section IV shows this system applied to the task of manipulating a GUI and examine the performance of this system in a user study in section V. Lastly, we discuss in section VI various extensions of this system, including implications for gesture analysis, two-handed interaction, and accessible interaction for peopled with disabilities.

III. OUR APPROACH

An important challenge in using computer vision for perceptual user interfaces is the automatic real-time detection and tracking of objects in the scene that are relevant to the application, as well as their patterns of behavior. In many scenarios, 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 objects like 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.

In this section we describe our novel approach for computing lightweight sparse stereo vision from images captured by two inexpensive off-the-shelf cameras. Our framework will let us build interfaces that achieve smooth, natural interaction between the computer and the user.

A. Multiple Hypothesis Tracking

We use a multiple hypothesis tracking approach, in which simple, fast techniques are used to track multiple objects moving in the scene. We rely on domain specific constraints to determine the actual object of interest. 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 user is facing the cameras it is often the case that the objects closest to the cameras are the hands. Another application may focus on objects that behave in a particular way over time. One advantage of this multiple hypothesis approach is that we may use a simple, fast, and imperfect tracking algorithm and rely on the fact that if one tracker fails, another may still be following the object of interest.

We exploit the observation that our own attention is often drawn to objects that start moving or exhibit some kind of motion. Our algorithm initiates tracking of any objects in the scene that move. These objects correspond to regions in the image which undergo an amount of motion greater than some threshold. Motion is detected by comparing a patch of the current image centered about a given location to a patch at the same location from the previous image. Throughout, we use a simple image comparison function, the sum of absolute

differences 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 image 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 motion we simply find points (u, v) such that $SAD_{motion} = SAD(I_{t-1}, u, v, I_t, u, v) > \tau$. To limit computation, this test for image motion may be conducted on a sparse, regular grid on the image (e.g. every 16 pixels).

Once a given tracker has been seeded, the algorithm updates the tracker's position over time by finding the patch in the current image which best matches the patch centered on the object at the previous image. We define $SAD_{movement} = SAD(I_{t-1}, u_{t-1}, v_{t-1}, I_t, u_t, v_t)$ where (u_t, v_t) refers to the object's location at time t. A simple frame to frame tracker finds (u_t, v_t) that minimizes $SAD_{movement}$. We instead optimize a weighted sum of $SAD_{movement}$ and SAD_{motion}. This combination combats drift problems common to simple block-matching based trackers, where over time the tracker may begin following some part of the image other than that corresponding to the intended object. Intuitively, the 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 after some time (typically less than one second), a tracked object exhibits very little movement, it is removed from the set of 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 removed from consideration.

B. Stereo Disparity Computation

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 a simple matter to compute the depth of the object. 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

$$u = u_r = f\frac{x}{z} \quad (2)$$

$$v = v_r = f\frac{y}{z} \quad (3)$$

$$d = u_r - u_l = f\frac{b}{z} \quad (4)$$

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 [12].

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. Most often such region-based approaches are used to compute a depth map which gives the depth in the scene at every location in the image. Computing a depth map is a very computationally intensive process, and often requires dedicated hardware to run in real-time [13, 14].

In fact, most applications do not require a complete depth map. Our approach consists of using the region-based approach to find the disparity at only locations in the image corresponding 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 centered on (u, v) and a corresponding patch in the left image centered over $(u - d, v)$ is minimal, i.e. d that minimizes $SAD_{disparity} = SAD(I_l, u - d, v, I_r, u, v)$. Furthermore, if we already have 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 dealt with automatically by fixing the cameras on a prefabricated mounting bracket, or semiautomatically by the user presenting objects at known depth in a calibration routine that requires a few minutes at most. 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 [15]. This camera calibration procedure also gives the value of the focal length f.

C. Summary of the Vision Algorithm

Figure 2 illustrates the 3-d tracking and 3-d depth computations. In this process, each object hypothesis is supported only by consistency of the object's movement in 3-d. Unlike many other computer vision algorithms, this does not rely on fragile appearance models such as skin color models or hand image templates, which are prone to fail when environmental conditions change or when the system is confronted with a new user. This robustness comes at a cost of relying on application constraints to determine which object to follow. We believe that this is a valuable trade-off in many circumstances.

In some cases there is a natural criterion to adopt. For example, in an application where the cameras are placed in

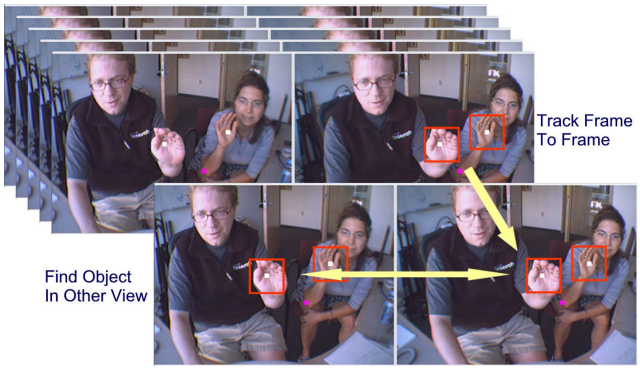
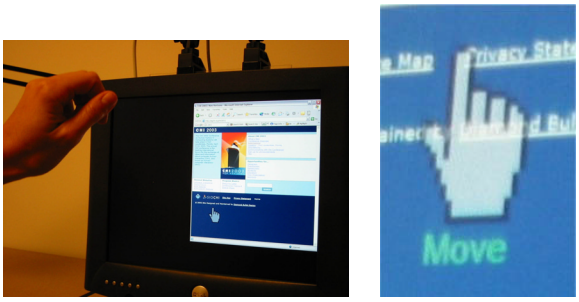


Fig. 2. Object hypotheses (indicated by square solid colored dots on the image) are supported by frame to frame tracking through time in one view and stereo matching across both views.



(a) (b)

Fig. 3. The GWINDOWS system allows the user 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.

a common configuration above the display, and the goal is to interact with objects on the display, the application may simply focus its attention on the nearest objects. If by the given application context the user is predisposed to use hand gestures towards the display, in practice, the nearest object hypotheses will fall on the hands. In other scenarios we may wish to design more elaborate criteria for object selection. For example, an application may select a particular object based on its 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.

Note that all the image operations use the same sum of absolute difference function on the image patches. This detail allows easy SIMD optimization of the algorithm's implementation, which in turn allows it to run with sufficiently many trackers while still leaving the user CPU time.

IV. 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 interface enabling users to "grab" a window with their hands and move it across their desktop, close, minimize, maximize windows as well as scroll the foreground window.

GWINDOWS was designed with the view that perceptual user interfaces may be applied to everyday 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 [16] 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 [17]. 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 users hand. This is illustrated in Figure 3. A similar scheme for hand position feedback was used in [5].

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 can be found in [18].

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 labelling offers valuable feedback to indicate the success of the speech recognition process.

In addition to using speech to initiate modes of interaction such as moving or raising windows, the user may additionally operate GWINDOWS by using gestures. In this case, the user may select the above 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.

A. Implementation Details

Our implementation of GWINDOWS uses the computer vision system described section III, 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. Many more trackers may used, but in our experience, 6 trackers is 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. Our sparse stereo implementation uses our own MMX implementation of the sum of absolute differences image function (Equation 1). The current system takes less than 20% of the CPU time on a 1GHz Pentium III.

The engagement and acquisition of the hand is implemented in the sparse stereo 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.

V. USER STUDY

A. Introduction

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 [6]. 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.

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

B. Participants

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.

C. Apparatus

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). The participants had no access to 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. Each session was videotaped.

The early version of GWINDOWS used in this experiment allowed the user to trigger commands only with speech, and not the dwell time-based technique or gesture recognition previously described. Additionally, the tracking algorithm used in the experiment has been made more reliable since the user study.

D. 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 previously mentioned commands.

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 experiment 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.

Finally, 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.

E. Discussion

After three to five minutes of interaction with the system, all but one of the users were comfortable managing windows using GWINDOWS. 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 about 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, this change in behavior in the extreme tends to degrade performance. After realizing that such a strategy was not helpful, they would come back to the "normal" way of operation.

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 (due to a change of trackers in the multiple hypothesis tracking) 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 such a problem. One user suggested different hand icons for each of the commands. Currently, GWINDOWS displays the current command in green lettering under the virtual hand. Many users thought that GWINDOWS would be a good system for kiosk environments or at home, *i.e.* in 10ft interfaces.

We will briefly summarize the results of the survey as average ratings $\pm$ standard deviation in the Likert scale of the answers of all participants. From the survey, we may conclude that the participants enjoyed (1.3 ± 0.1) interacting with the computer using GWINDOWS. They also wished there were more systems available that used vision for interaction (2.1 ± 0.1) and they were not particularly concerned about the privacy implications of having cameras in their environment (3.5 ± 0.31). Participants imagined GWINDOWS being used in accessibility scenarios first (1.5 ± 0.15), 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). As a curiosity, they all had very positive reaction to the video user interface that appears in the movie *Minority Report* (1.6 average rating). 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 their arm getting tired after a while (2 ± 0.26).

With respect to the experiment, participants were generally satisfied with their performance $(2.3 \pm 0.24,\ 2$ median) and they thought they could improve their performance with more practice (1.5 ± 0.2). Participants found it easy to understand what the computer was doing in response to their actions (2 ± 0.19) and had no trouble remembering how to perform each of the commands (2 ± 0.17). They all understood how to "engage" and "disengage" with the system (1.3 ± 0.11) and found they could reliably engage and disengage when they wanted to (2.2 ± 0.27). They found the control of the virtual hand on the screen more responsive $(2.5 \pm 0.24,\ 2$ median) than accurate $(2.7 \pm 0.23,\ 3$ median). 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), showing a slight preference for gestures instead of speech (2.7 ± 0.26), even though the present system provides no support for using gestures to invoke commands. Finally, participants were confident that they could CLOSE, MOVE, RAISE, SEND TO BACK any window with average scores between 1.9 ± 0.09 and 1.6 ± 02. The SCROLL command was the most difficult with an average score of 2.1 ± 0.28. We believe that the main reasons for the difficulty in scrolling came from the rate-control mechanism employed in the scrolling mode.

VI. EXTENDING GWINDOWS

GWINDOWS provides an auxiliary cursor which may be used to select windows on the screen. The user study shows that, although people found the interface very easy to learn, fatigue and speech recognition errors were problematic. We believe 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 with some degree of precision. Fatigue may be addressed partly by (per one of our subject's suggestions), scaling the movement of the hand, thus requiring a smaller range of movement to reach all parts of the screen, or changing the configuration of the cameras such that they track object motion just above the keyboard. We have addressed the speech recognition problems in part by using the dwelling time and flick gestures to perform all the tasks except for scrolling.

In scenarios where a mouse is available, there may be more value in considering how GWINDOWS can *complement* the mouse and keyboard. Here we consider a number of promising ideas.

A. More Complex Gestures

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.

In the simplest view, gestures play a symbolic communication role similar to speech, suggesting that for simple tasks gesture may enhance or replace speech recognition. Secondly, we believe that small gestures near the keyboard or mouse do not induce fatigue as quickly as sustained whole arm postures. Long et al [19] observe that users find gesture-based systems highly desirable, but that users are also dissatisfied with the recognition accuracy of gesture recognizers. Furthermore, experimental results that users' difficulty with gestures is in part due to a lack of understanding of how gesture recognition works. Finally, they 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 to the one side or the other, as in shooing a fly. Another example is gestures for "next" and "back" functionality found in web browsers, PowerPoint and other applications. In [20], Moyle and Cockburn show that adding a simple gesture-based navigation facility to web browsers can 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. Users' subjective ratings in their experiments 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.

Even when a 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 coffee might be accomplished by mapping simple gestures to "next message" and "delete message".

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, the foreground object may be "bumped" to another display by gesturing in the direction of the target display. In our experience, our flick gestures let the user quickly send windows from one monitor to the next without the need of large mouse movements.

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.

B. Two-Handed, Mouse and Hand Interfaces

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 [21] 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. Theoretical 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 [22]. 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) [23].

C. Accessibility

Users with a number of motion impairment conditions cannot cope with most current computer access systems. Such conditions include Cerebral Palsy, Muscular Dystrophy, Friedrich's Ataxia and spinal injuries or disorders. Frequent symptoms include tremor, spasm, poor coordination, restricted movement, and reduced muscle strength. Similar symptoms are also seen amongst the elderly able-bodied population. Older adults are the fastest growing segment of the population in the United States and Europe. It is known that as people age, their cognitive, perceptual and motor skills decline, with negative effects in their ability to perform many tasks [24]. Computers can play an increasingly important role in helping older adults function well in society.

Graphical interfaces contribute to the ease of use of computers. WIMP interfaces allow fairly non-trivial operations to be performed with a few mouse motions and clicks. However, at the same time, this shift in the user's interaction from a primarily text-oriented experience to a point-and-click experience has erected new barriers between people with disabilities and the computer. For example, for older adults, there is evidence that using the mouse can be quite challenging. There is extensive literature demonstrating that the ability to make small movements decreases with age [25]. This decreased ability can have a major effect on the ability of older adults to use a pointing device on a computer. It has been shown [26] that even experienced older computer users move a cursor much more slowly and less accurately than their younger counterparts. In addition, older adults seem to have increased difficulty (as compared to younger users) when targets become smaller. For older computer users, positioning a cursor can be a severe limitation.

One solution to the problem of decreased ability to position the cursor with a mouse is to simply increase the size of

the targets in computer displays, which can often be counterproductive since less information is being displayed, requiring more navigation. Another approach is to constrain the movement of the mouse to follow on-screen objects, as with sticky icons [17] or solid borders that do not allow cursors to overshoot the target. There is evidence that performance with area cursors (possibly translucent) is better than performance with regular cursors for some target acquisition tasks [27, 28]. GWINDOWS combines area cursors with gesture-based manipulation of on-screen objects. Currently GWINDOWS may be configured to be driven by gross or fine movements, and may be helpful to people with limited manual dexterity.

VII. 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.

GWINDOWS demonstrates the use of our sparse stereo framework in everyday GUI tasks. Users were able to pick up the system very quickly, and many were pleasantly surprised to find how responsive the system is. We look forward to exploring extensions to GWINDOWS that highlight interaction scenarios in which perceptual user interfaces add value beyond traditional interfaces.

VIII. ACKNOWLEDGEMENTS

We thank the subjects participating in our experiments.

REFERENCES

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

[2] E. Horvitz, "Principles of mixed-initiative user interfaces," in *Proc. of CHI '99*, 1999.

[3] *Perceptual User Interfaces Workshop Proceedings*, 1997, 1999, 2001.

[4] 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.

[5] W. T. Freeman and C. Weissman, "Television control by hand gestures," in *Intl. Workshop on Automatic Face and Gesture Recognition*, 1995, pp. 179–183.

[6] F. Kjeldsen, "Visual interpretation for hand gestures as a practical interface modality," Ph.D. dissertation, Department of Computer Science, Columbia University, 1997.

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

[8] 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: IEEE Computer Society Press, August 1996.

[9] 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.

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

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

[12] B. Horn, *Robot Vision*, M. Press, Ed., 1988.

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

[14] T. Darrell, G. Gordon, M. Harville, and J. Woodfill, "Integrated person tracking using stereo, color, and pattern detection," pp. 601–608, 1998.

[15] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[16] "http://www.microsoft.com/windows/ehome/default.asp."

[17] 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*, 1997, pp. 266–271.

[18] "http://research.microsoft.com/∼nuria/gwindows/gwindows.htm."

[19] J. A. Long, J. Landay, and L. Rowe, "Implications for a gesture design tool," in *Proc. CHI'99*, 1999, pp. 40–47.

[20] M. Moyle and A. Cockburn, "Gesture navigation: An alternative 'back' for the future," in *Proc. of CHI02*, 2002, pp. 822–823.

[21] 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*, 2002, pp. 650–651.

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

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

[24] T. Salthouse, *Theoretical Perspectives on Cognitive Aging*. Hillsdale, NJ: Erlbaum, 1991.

[25] A. Welford, "Signal, noise, performance and age," *Human Factors*, vol. 23, pp. 97–109, 1981.

[26] N. Walker, D. Philbin, and D. Fisk, "Age-related differences in movement control: Adjusting submovement structure to optimize performance," *Journal of Gerontology: Psychological Science*, 1997.

[27] P. Kabbash and W. Buxton, "The prince technique: Fitts' law and selection using area cursors," in *Proc. of CHI'95*, 1995, pp. 273–279.

[28] S. Zhai, W. Buxton, and P. Milgram, "The "silk cursor": Investigating transparency for 3d target acquisition," in *Proc. of CHI'94*, 1994, pp. 273–279.