Combining Head Tracking and Mouse Input for a GUI on Multiple Monitors

Mark Ashdown, Kenji Oka, Yoichi Sato
Institute of Industrial Science, The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505, Japan
mark@ashdown.name, oka@iis.u-tokyo.ac.jp, ysato@iis.u-tokyo.ac.jp

ABSTRACT

The use of multiple LCD monitors is becoming popular as prices are reduced, but this creates problems for window management and switching between applications. For a single monitor, eye tracking can be combined with the mouse to reduce the amount of mouse movement, but with several monitors the head is moved through a large range of positions and angles which makes eye tracking difficult. We thus use head tracking to switch the mouse pointer between monitors and use the mouse to move within each monitor. In our experiment users required significantly less mouse movement with the tracking system, and preferred using it, although task time actually increased. A graphical prompt (flashing star) prevented the user losing the pointer when switching monitors. We present discussions on our results and ideas for further developments.

Author Keywords

Gaze-contingent display, attentive user interface, head tracking, multiple monitors.

ACM Classification Keywords

H.5.2 [Information interfaces and presentation]: User Interfaces - Input devices and strategies.

INTRODUCTION

The falling price of flat-panel monitors is encouraging the trend of multi-monitor use for PCs. Multiple-monitor systems allow people to be more productive [3], but also cause changes in the way they use graphical user interfaces, for instance people will arrange windows within monitors rather than across the boundaries between monitors.

Use of multiple monitors increases multitasking and puts strains on window management. Czerwinski *et al.* [1] found that users of a large display were surprised when the completely unobscured window they were looking at did not receive keyboard events—actually a different window was active but was positioned outside the user's field of view. Users of such systems employ strategies to alleviate the problem of transferring the mouse pointer and keyboard focus between monitors, such as dedicating one monitor to a display-only task like monitoring email [3]. We would like to allow the interactive potential of all monitors to be fulfilled.

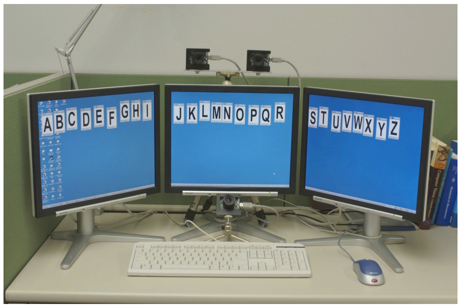

Figure 1. Experimental participants arranged a series of windows across three monitors. Two cameras are mounted above the centre monitor, and there is a spare one below.

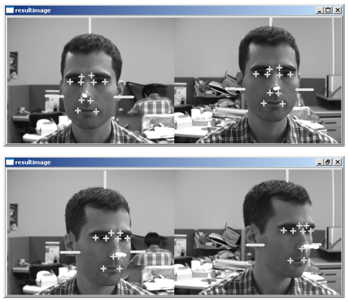

Figure 2. Feature points on the user's face are tracked by two cameras, and the head pose, consisting of position and orientation, is estimated.

A person's explicit actions can be combined with implicit information like eye gaze. Also, a user usually looks at a target before selecting it [4] so the direction they are looking in could be used together with the conventional mouse and keyboard devices for a display that is large enough that moving around it is cumbersome.

The inherent accuracy limitations of any attentive interface will probably also require explicit inputs if it is to be used for precise tasks of the type performed on conventional GUIs. Also, one must consider the 'Midas Touch' problem: what seems like a fast and accurate response in a constrained test may become an annoyance and source of error in a general user interface where many tasks are interleaved. MAGIC

CHI 2005, April 2–7, 2005, Portland, Oregon, USA.
ACM 1-59593-002-7/05/0004.

Pointing [10] combines eye tracking and fixation detection with mouse input to move the mouse pointer close to the position at which the user is looking on a single monitor. Users were able to make use of the prototype system, and showed some improvement in task times.

Eye tracking does not work well in the presence of large changes in head position and orientation. Movements of this type occur in a standard multi-monitor setup: the head must be rotated through a considerable angle to view three monitors (Figure 1). Rather than using eye tracking, we have used a vision-based head tracking system developed in our lab [6]. Other technologies such as electromagnetic and ultrasonic trackers exist but if the system is to be used in an everyday setting it must be non-intrusive and automatically initializing. That is, it must not require the user to wear any device and it must allow her to freely move away from the computer then return and continue working immediately. Our system uses the head tracking data to move the mouse pointer between monitors and switch the active application.

Head motion is different to eye motion. The eye is constantly moving, its motion characterized by fixations interspersed with rapid saccades. Head motion is more stable, but less accurately indicates the user's focus of attention because the eyes can move independently of the head. The head movement has been shown to closely follow the eye movement in response to stimuli [2], and the head generally moves so as to keep the eye near the centre of its orbit and away from the extremes. Head motion has previously been shown to be useful for switching between windows on a single-monitor system, and for zooming and scrolling a map [5].

HEAD TRACKER

Robust real-time tracking of users suitable for human interaction is an active research area. Vacchetti *et al.* [8] provide a good example of object tracking, in which a mesh model of the object is provided, then then the object is tracked using a single camera. Our head tracking system [6] differs in various ways: it uses multiple cameras, it automatically acquires the model of the user's face rather than requiring a mesh, and it uses stochastic filtering which generally leads to more robust algorithms. Our algorithm has two stages: automatic initialization of a 3D model of the user's head with multiple feature points, and tracking the head pose in consecutive image frames using a particle filter. For the experiment described here we used two cameras, although the software supports more.

The initialization process starts automatically when a user appears: their face is found by a face detector from the OKAO vision library by Omron Corporation. A set of feature points on the face is identified (see Figure 2—currently we use ten points) and for each one the algorithm obtains the 3D location, via stereo triangulation, and an image template from each camera. During tracking the six-dimensional head pose, consisting of position and orientation, is continually estimated. A particle filter follows multiple hypotheses in the 6D state space, these are compared with the actual image frames, and finally an estimation of the pose is obtained. A key component of the method is adaptive control of diffusion factors in the particle filter, which makes it robust against abrupt motion while also providing high accuracy.

The head tracker works at the full video rate of 30 frames per second. Accuracy depends on the camera configuration which is set to encompass the space in which the head will move, but when the head is stable the accuracy is around 1 mm for position and 1 degree for orientation. Latency is minimal—around one frame, or one thirtieth of a second—and in choosing a monitor we do not perform any filtering, such as averaging measurements from several frames, so no lag is introduced. During testing we found that when typing the user will often look down at the keyboard, so we have mounted a third camera below the monitor to ensure that the feature points remain visible when the head is tilted downwards.

APPLICATION TO MULTIPLE MONITORS

We use our head tracker to make the mouse pointer jump between monitors. Our system runs on Microsoft Windows XP which is otherwise unaltered. In a simple calibration procedure the user looks at a point on each of the monitors in turn. During tracking the scalar product of the current head direction with the stored direction for each monitor is computed to determine how closely the user is looking at each monitor. These values are used to select one monitor, after application of a hysteresis threshold.

Allowing the pointer to be moved from one monitor to an adjacent one with the mouse, in addition to having it jump in response to a head movement, would cause conflicts between the two methods. We therefore restrict it to a single monitor: the pointer is moved within each monitor by the mouse, and between monitors by the head. Because head direction is not as accurate as eye direction at indicating focus of attention, we only use the head direction to pick a monitor, thus the pointer appears on a new monitor in the same position that it was in on the old monitor. If we wanted the new pointer position to be determined by the head direction we would need to wait until the head had settled on a particular direction before choosing the new point. That is, we would need an algorithm to detect fixations in head movements, like the fixations and saccades that are detected by eye tracking systems. This possibility is discussed below.

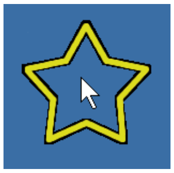

Figure 3. When the mouse pointer jumps between monitors a flashing star helps the user to find its new position.

Jacob [4] states that for eye tracking there is a dilemma as to whether to display a cursor. If the tracking has zero error it will not be necessary, and will probably be distracting, but the system will probably exhibit some error so feedback on the estimated position will be useful. For our head tracking system, because the pointer may not be at the current point of regard when the user looks at a new monitor, a star appears around new position of the pointer to make it more prominent (Figure 3). The star flashes five times over one second, and a short beep indicates that the pointer has moved.

A user can move windows around within a single monitor as usual. To move them between monitors she grabs the title bar as usual then looks at a different monitor. The mouse pointer jumps to the new monitor and the window goes with it. When the user switches monitors without dragging a window the top window on the new monitor is activated, so she can immediately start interacting with it, by typing for example. This is to avoid the problem mentioned above of the user failing to realize that the active window is on another monitor [1], and to make it faster to switch between applications—no explicit action is necessary to do so.

The automatic jumping of the mouse pointer means that less work is required from the hand when moving between windows, particularly those far apart on a multi-monitor system. As more monitors are used multitasking increases, so switching between windows will become more frequent, and as monitors become larger and more numerous the distance traversed to do that switching will increase.

EXPERIMENT

To test if people could use our system and to get some feedback we devised a window management task to be performed on multiple monitors. We used a within-subjects design with eight members of our research group, of whom all used Windows regularly but none used multiple monitors. We used three 17-inch 1280×1024-pixel LCD monitors arranged on a desk about 50cm from the user's eyes. We used Windows XP with default settings and compared two conditions: with our head tracking system, and without. We used a Microsoft IntelliMouse, and the default Windows mouse setting which includes acceleration so it takes a mouse movement of about 12cm traverse a monitor moving slowly, or 3cm moving very quickly.

The task was to arrange a set of 26 windows containing the letters of the alphabet from left to right across three monitors (Figure 1). Initially the windows were scattered randomly over the three displays. After receiving an explanation of the system and practising with the two conditions, participants performed the task four times, twice with and twice without the tracker, with the ordering being balanced. The time in seconds and total mouse movement in pixels were measured for each instance of the task. We discarded the data from the first two tasks for each user, treating them as extra practice. At the end of the tasks we asked the participants which condition they preferred, and asked them for further comments.

We used a paired t test to analyse the distance and time data. With the head tracker mean distance moved by the mouse in pixels reduced by 32% (Figure 4) from 82,844 to 56,142 ($t(7) = -5.71, p < 0.001$), indicating that less effort was expended in moving the mouse when the head tracker was used. Seven out of the eight participants preferred the head tracker condition. In the comments participants said that the flashing star was very helpful for finding the pointer after a change of monitor.

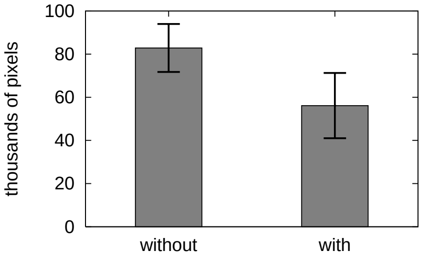

Figure 4. Total distance moved by the mouse during the task without the head tracker, and with it. Error bars show standard deviations.

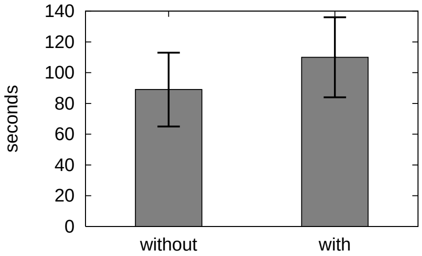

Figure 5. Time to complete the task without the head tracker, and with it. Error bars show standard deviations.

Unfortunately, mean time to complete the task increased by 24% (Figure 5) from 89 to 110 seconds ($t(7) = 5.78, p < 0.001$). The main complaint was the awkwardness of moving the pointer between points that are quite close together but on different monitors. Moving within one monitor was easy, and moving from, for instance, monitor 1 to monitor 3 was aided by the head tracker, but moving from the edge of one monitor to the nearest edge of an adjacent monitor was easier in the condition without the tracker (Figure 6).

The problem of moving the pointer between close points on adjacent monitors would be alleviated if the pointer jumped to the user's point of regard on the new monitor rather than to the position from the old monitor. We did not implement this because of the requirement for a model of head movements and an algorithm to detect fixations. Implementing such an algorithm is an option because head motion, being slower and less erratic, is much simpler than eye motion. Specifying the link between eye and head movement is an active area of research in neurophysiology. Similar to eye tracking, head fixation detection could use velocity or dispersion statistics, and a time threshold [7]. The simplest method is a velocity threshold: whenever the velocity of the head is low throughout a certain window of measurement points, generate a fixation event beginning at the start of the window.

Several participants commented that performance with the head tracking system would improve with practice. The participants had obviously had extensive practice with the conventional mouse interface, but needed to get used to the head tracker, thus we omitted the first two results from each per-

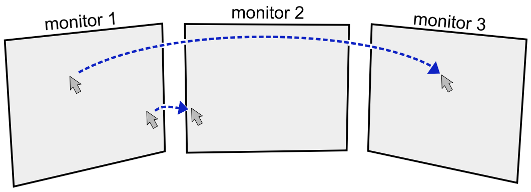

Figure 6. Moving large distances with the head tracker is easy, but moving a small distance across a monitor boundary can be awkward.

son to reduce the learning effect. Another result was that users looked at a target before selecting it allowing the system to select the correct monitor, but some users occasionally did not look at the destination. This was when they were casually placing a window in a rough temporary position.

CONCLUSIONS & FUTURE WORK

We have presented a system that combines head tracking and mouse input to allow a user to control a conventional GUI that spans multiple monitors by moving the head to select a monitor. In our experiment participants were able to use the system for a window management task across three monitors, they required significantly less mouse movement with the tracking system, and they preferred using it, although task time actually increased. The main problem was moving between nearby points on adjacent monitors. A graphical prompt (flashing star) prevented the user losing the pointer when switching monitors.

Various techniques have recently been developed to address the difficulties of GUIs spread over large screens and multiple monitors. 'Drag-n-pop' reduces mouse movement by moving targets towards the cursor, conversely to our method which makes it faster to move the cursor to targets. 'Mouse ether' and the 'high-density cursor' avoid the mouse getting lost when moving quickly or between monitors—our flashing star performs this role, but should be unnecessary if jumping to the point of regard is implemented. 'Bumping windows' moves a window between monitors using a key press, whereas in our system the user grabs the window then moves the head. New window management techniques are also being developed—they could be supported by giving a window that jumps to a new monitor an automatically selected location and size.

Three important criteria for interfaces based on sensing techniques such as eye or head tracking are:

- the distinction between implicit and explicit inputs
- whether tracking data are treated as continuous or discrete
- the cost of mistakes

Implicit inputs are generated automatically in response to sensing of the user's inadvertent movements; explicit ones are consciously performed. The continuous stream of data generated by a tracking algorithm can either be used to continuously adjust parameters of the application, or it can be converted to a series of discrete events by, for example, applying a threshold, or using a hidden Markov model. The cost of mistakes should be kept low so that the user can easily recover from their own errors or errors of the tracking system.

To remove the problem of moving between close points on separate monitors we could combine tracking (implicit input) and button presses (explicit action.) The user could hold down a specially assigned mouse button to have the pointer follow his head movements, then release it to resume normal control. This would not require fixation detection, would avoid the problem with moving between adjacent monitors, and would avoid the Midas Touch problem, although it would require an explicit trigger.

Head or eye tracking could be used to create a system like GAZE-2 [9], a video-conferencing application in which a higher data rate is assigned to the video stream at which the user is currently looking. Input is implicit, and the cost of a tracking error is low because a lower rate video stream is still useful. We could implement something like the 'cocktail party' effect for standard application programs that are sharing a sound output channel, or some other resource. The volume of each sound stream would be weighted by a the distance between its application window and the user's point of regard. Again this would have the advantages of requiring only implicit input and having low cost of error.

ACKNOWLEDGEMENTS

We thank the Japan Society for the Promotion of Science for funding this work via a postdoctoral fellowship, and Omron Corporation for providing the OKAO vision library used in the initialization step of our head tracker.

REFERENCES

1. M. Czerwinski, G. Smith, T. Regan, B. Meyers, G. Robertson, and G. Starkweather. Toward Characterizing the Productivity Benefits of Very Large Displays. In *Proc. Interact 2003*, pages 9–16, 2003.
2. H. Goosssens and A. V. Opstal. Human Eye-Head Coordination in Two Dimensions Under Different Sensorimotor Conditions. *Exp. Brain Research*, 114:542–560, 1997.
3. J. Grudin. Partitioning Digital Worlds: Focal and Peripheral Awareness in Multiple Monitor Use. In *Proc. CHI 2001*, pages 458–465, 2001.
4. R. J. Jacob. *Advances in Human-Computer Interaction, Vol. 4, ed. by H.R. Hartson and D. Hix*, chapter Eye Movement-Based Human-Computer Interaction Techniques: Toward Non-Command Interfaces, pages 151–190. Ablex Publishing Co., 1993.
5. K. Kitajima, Y. Sato, and H. Koike. Vision-Based Face Tracking System for Window Interface: Prototype Application and Emperical Studies. In *Proc. CHI 2001*, pages 359–360, 2001.
6. K. Oka, Y. Sato, Y. Nakanishi, and H. Koike. Head Pose Estimation System Based on Particle Filtering with Adaptive Diffusion Control. In *Proc. MVA 2005*, 2005.
7. D. D. Salvucci and J. H. Goldberg. Indentifying Fixations and Saccades in Eye-Tracking Protocols. In *Proc. Symp. Eye Tracking Research & Applications*, pages 71–78, 2000.
8. L. Vacchetti, V. Lepetit, and P. Fua. Stable Real-Time 3D Tracking Using Online and Offline Information. *IEEE Trans. PAMI*, 26(10):1385–1391, 2004.
9. R. Vertegaal, I. Weevers, C. Sohn, and C. Cheung. GAZE-2: Conveying Eye Contact in Group Video Conferencing Using Eye-Controlled Camera Direction. In *Proc. CHI 2003*, pages 521–528, 2003.
10. S. Zhai, C. Morimoto, and S. Ihde. Manual And Gaze Input Cascaded (MAGIC) Pointing. In *Proc. CHI 99*, pages 246–253, 1999.