RC24065 (W0609-158) September 29, 2006
Computer Science

IBM Research Report

Improvements in Vision-based Pointer Control

Rick Kjeldsen
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Improvements in Vision-based Pointer Control

Rick Kjeldsen
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
(914) 784-7558

fcmk@us.ibm.com

ABSTRACT
Vision-based head trackers have been around for some years and are even beginning to be commercialized, but problems remain with respect to usability. Users without the ability to use traditional pointing devices – the intended audience of such systems – have no alternative if the automatic boot strapping process fails, there is room for improvement in face tracking, and the pointer movement dynamics do not support accurate and efficient pointing. This paper describes a novel head tracking pointer that addresses these problems.

Categories & Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces -- *input devices and strategies*

General Terms
Human factors

Keywords
Human-computer interaction, Vision-based user interfaces, Accessible user interfaces

1. INTRODUCTION
Vision-based interaction, computer input techniques based on a camera and computer vision software, have matured significantly in recent years (see e.g. [11]). The use of vision, a non-contact sensing modality, makes these systems more flexible than input based on fixed hardware. This flexibility can be used to allow these systems to adapt to the needs of the user, giving them great potential as an input method for people with physical disabilities that make standard input devices difficult or impossible to use.

Head tracking is an alternative interaction mode which has received a great deal of attention. Users who have reasonably good head control, but are otherwise limited in their use of pointing devices, have had success using head tracking for complete pointer control. For users who are able to use conventional pointing devices, though perhaps with difficulty, head tracking can reduce the need to resort to the mouse for basic tasks such as large object selection, symbolic interactions (e.g. Yes / No dialog boxes) or to simplify tasks that require higher levels of dexterity, such as scrolling.

Hardware-based head tracking systems using laser and active IR (e.g. www.madentec.com) have been around for some time, but the hardware is expensive and many users seem to prefer an approach that does not require them to wear the headsets or markers these systems require. Recently vision-based head trackers have become available that address that need (e.g. www.eyetwig.com, the QualiEye at www.qualilife.com).

This paper describes the HeadTracking Pointer (HTP), which attempts to address some of the problems we have encountered in other systems. We provide improved end user control over the system by providing a simple way for them to signal the system when tracking is not working well, so corrective action can be taken. We also describe an improved method of converting head movement to pointer movement, one which is better suited to the needs of human body tracking applications than the approaches used to date. The result is a camera-based head tracking pointer which is more accurate and pleasing to use.

2. OTHER WORK
People have been exploring vision-based head tracking for some time. Most of that work has focused on the technical aspects of how to efficiently locate and track the user's head. Early approaches were hardware intensive lab prototypes using multiple cameras, often dedicated hardware, and complex feature extraction algorithms e.g. [9].

More recently the idea of using head tracking for pointer control has caught on. This work has generally had more modest hardware requirements, usually a single camera and one or two personal computers. In [12] Toyama illustrates one approach, where the 3D orientation of the head is estimated from the view of a single camera. This, together with assumptions about the relative position of

the user and the monitor, is used to determine where a line extended from the nose would intersect the screen, and so position the pointer.

The systems described in [1] and [3] use a somewhat different approach, more similar to the one taken here. The absolute position of the head is never estimated, rather a feature on the face is located and tracked. The location and motion of that feature relative to the image is used to locate and move the pointer on the screen. Nouse ([3]) relies on tracking the nose, which, since it extends in front of the face and ends with a somewhat universal rounded shape, is relatively easy to identify and track as the user's moves about. The pointer is moved using the offset of the nose from a center point, presumably acquired during training. Nose offset is converted to rate of movement of the pointer, but this process is not described in detail.

CameraMouse ([1]) locates visible features on the user's face, then tracks face movement by searching for similar looking regions in subsequent frames. This approach is similar to, though simpler than the one taken in HTP. The literature describes a system where the user must manually select a feature to track, but the most recent CameraMouse downloads have implemented a relatively robust automatic face finding process. The process of converting the face location/motion into pointer location/motion is not described in any detail, but we infer that the facial location is filtered to remove noise, then mapped to pointer motion by some algorithm that allows the user's face to point roughly in the direction of the pointer location.

The work presented here differs from these others in several ways, two of which are especially significant for this audience. First, we give the end user the ability to control system operation more directly, using head gestures that are available even when pointer control fails. Second we use a pointer control function ("transfer function") that takes into account the dynamics of human motion to give a smoother and more responsive pointer motion. Other differences also contribute to the overall usability of the system, most importantly the specifics of the face tracking algorithm. This will be discussed only briefly in this paper.

Moving a pointer by tracking a body part requires three distinct processes, bootstrapping the system by locating the body part, then tracking it accurately, and finally turning that tracking information into pointer movement on the screen.

Existing systems bootstrap in one of two ways. Early systems required someone to identify the body part to be tracked using the standard interface devices (e.g. mouse) before vision-based pointer control could begin. This required that users unable to use a standard pointing device have someone help them initially and again whenever tracking begins to fail because of lighting or environmental changes.

More recent head tracking systems rely on automatic face finding. While automatic face finding has improved significantly, it is still prone to failure in difficult lighting or other unusual situations. For example, automatic face finding often relies on the use of color for identifying skin, which can be unreliable because of its variability in the face of lighting changes, the variability of human skin color, and the presence of similar colors in the environment. While automatic face finding can be very robust if the environment is kept within certain bounds (by adjusting lighting or providing a contrasting background), it is preferable to have systems which do not depend on environmental conditions.

We feel that the user should not be at the mercy of either an automatic process or another person. If face finding fails or head tracking is not working properly, it is important that the end user should have some method of bootstrapping the system without outside help. Our approach has been to provide an alternate channel, independent of face tracking, by which the end user can signal the system to reset.

In our review of the literature, little attention has been paid to the details of the transfer function between the head movement and pointer movement. Tests on the downloadable version of various systems show there is a significant amount of noise in the pointer location, and small smooth positioning motions are not possible. Ideally, a user should be able to position the pointer to within about one on-screen character in order to perform all the tasks needed in an unmodified GUI. In our experience the pointer motion in most systems too rough to support positioning of this accuracy.

To address this, HTP uses a novel transfer function that takes into account the kinematics of human movement, the needs of pointer control in a modern GUI, and the more subjective characteristics that make a pointer feel responsive and pleasing to use.

3. SYSTEM DESCRIPTION

3.1 User's view

Figure 1 shows the HeadTracking Pointer from the user's point of view. The right window shows the camera's view, and the left window is the control panel. These windows initially appear in the top left 3" x 2" of the screen, but can be resized, moved, or minimize independently to suit the user's needs.

Figure 1: HeadTracking Pointer windows. On the left is the Click Control dialog and on the right the video mirror. These windows initially appear in the upper left corner of the screen.

When it starts, the system is running, but not tracking the user. To begin tracking, the user tips their head three times (left/right/left, or right/left/right) (figure 2) and then pauses. One thread of the program is always watching for this signal. When a possible tipping head is observed, a box is drawn around it to tell users their actions have been seen. After HTP observes the three head tips and a pause, it draws a large window centered at the middle of the screen and rapidly shrinks it to a small window saying "Aim Here". This catches the user's attention, and encourages them to aim their head at the center of the screen. After a brief pause, the system captures the image of the user's face, and begins to track it, moving the pointer in such a way that it moves to approximately where their nose is pointing.

The system may be retrained in this way at any time. Thus, if the system is not operating as the user would like – say the pointer is off to one side of where they aim their nose, or tracking is unsteady because the lighting conditions have changed - they need only tip their head to correct the problem.

To the left of the video window (figure 1) is the Click Control dialog. It contains buttons to enable the various click actions. Floating the pointer within one of these buttons for about 1 second (adjustable) enables the selected click and turns the button green. The user then moves the pointer to a location on the screen and keeps the pointer still (dwells) for about 1 second (again adjustable) to trigger the click.

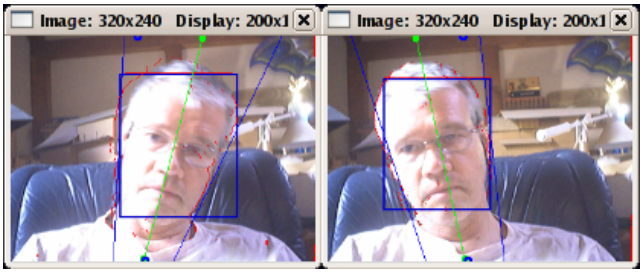


Figure 2: Head tipping motion used to train the HeadTracking Pointer. The green line denotes the approximate centerline of the face, which HTP uses to identify the tip movements.

The top two buttons control click generation. If the "Click Once" button is selected, a click will be generated only on the next dwell. When the "Turn Click ON" button is selected, it stays on and changes its label to "Turn Click OFF" (and the "Click Once" button disappears) until it is selected again. When Turn Click ON is activated, a click will be generated any time the user dwells at a point on the screen.

The remainder of the buttons control the type of click generated. Dwelling over "Right" or "Double" modifies the next click to that type. Note that the user need not select Click Once AND a modifier. If a modifier is selected, Click Once will turn on automatically (unless Turn Click ON is active). The "Drag and Drop" modifier button makes the subsequent dwell a mouse-down event, and the dwell after that a mouse-up event, so that drag-and-drop operations can be executed.

The Click Control dialog can be resized, which will also resize the buttons within it, so that the user can customize the window to suit their pointer dexterity. The video window can be resized or hidden. Both windows can also be moved around the screen as needed.

If another window covers the Click Control dialog, it will automatically pop to the top when the pointer moves over it, so that the user always has access to the buttons. To generate a click in a window at a point under the Click Control window, the user must move one of the windows first.

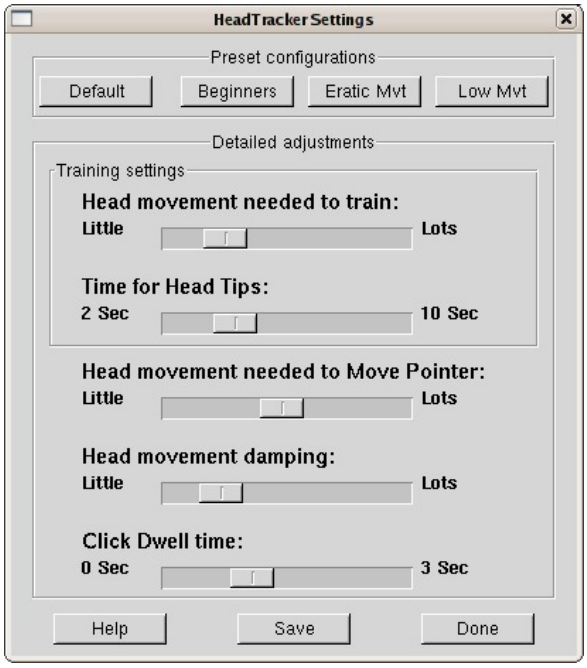


Figure 3: Settings adjustment dialog of HTP.

From the Click Control dialog the user can access a dialog to adjust various systems settings (figure 3). The settings are divided into groups for Training and Tracking. For training, the user can adjust the amount of head movement required to be considered a head tip, and the amount of time needed to complete the head tip training. Reducing these parameters makes the system easier to train for users with limited or difficult head movement, but makes inadvertent training more likely. Inadvertent training is not a significant problem – the "Look Here" window appears, the user aims their head, pauses, then can resume work – but it can be annoying if it occurs unexpectedly.

The user can also adjust the behavior of the system during tracking. The amount of head movement needed to move the pointer from one side of the screen to the other can be adjusted, as can the "damping" of the pointer, the exact meaning of which will be described later in this paper.

At the top of the dialog are buttons that allow the user to initialize the sliders to one of several preset configurations.

The remainder of this section describes how the main components of HTP are implemented.

3.2 Training

The head tip training signal is always available to the user. It is implemented independently from the head tracking for pointer control, using algorithms designed to be robust to almost any background or lighting conditions. In essence there are two head tracking processes going on in parallel, one that is very robust and one that is very accurate. The first is used to bootstrap the second.

A tipping head is defined as a moving blob of the correct shape (oval, taller vertically than horizontally), where the vertical axis moves back and forth within the right range of angles and within a range of frequencies expected of a tipping head.

The head tip thread first computes a frame-to-frame difference between images, labeling all the pixels which have changed in appearance, to create a motion mask as shown in figure 4. Some basic image processing eliminates spurious dots and highlights the remainder. The bounding rectangle of the result is computed and

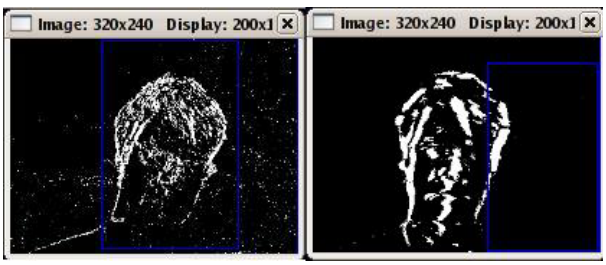


Figure 4: Motion mask found during a head tip signal. On the left is the raw data, on the right the data after some simple processing.

analyzed to ensure the aspect ratio is what is expected for a tipping head. A Hough Transform [5] is used to identify approximately vertical lines in this data. Obvious outliers are eliminated, and the lines with the most support are combined to determine the vertical axis of the head (blue line in figure 2). The angle of this line is monitored to identify head tips.

Since this process relies on moving objects, when the head pauses at the end of the tips it disappears, however we can assume it is still at the same location. The last observed head location is recorded, along with the size and shape of the head blob, which were estimated during the tips. From this information, the location of the center of the face (eyes/nose area) is estimated.

At this point the user is asked to aim their head the center of the screen. After a brief pause, allowing them to orient their head, a new image is recorded. The estimated eye/nose area is checked to ensure there is sufficient contrast in both the horizontal and vertical directions. If not, the region around it is searched till a suitable box is found. The image data in this box is recorded as our original tracking template.

3.3 Tracking

Once a tracking template has been recorded, the system uses it to follow the face in subsequent images of the video stream. The tracking algorithm must be able to locate the face as accurately as possible with little or no drift over time. Unlike CameraMouse (as described in [1]), which uses the relative movement of the face to move the pointer, we use the absolute location of the face, as in our experience this helps make pointing more stable and predictable, but it requires a stable estimate of the location of the face. If the estimated location drifts to different parts of the face over time, tracking is adversely affected.

There are two general approaches to tracking an object like the face with a template. One is to search for image regions that match the original tracking template as closely as possible. This leads to problems when the appearance of the face changes as it moves throughout its range. Appearance can change both due to uneven lighting, as you would get when sitting next to a window, and because the face is a 3D object which is rotating as it moves. The end result is that methods which track only the original template tend to fail catastrophically when the face turns to certain orientations, though they recover well when it turns back.

The second tracking approach is to search for a region that most closely resembles the appearance of the template region from the previous image. Thus, as the appearance of the face changes gradually from frame to frame the tracking template adapts. The main problem here is that the location being tracked tends to drift across the face over time, with the result is that the pointer eventually

ends up not tracking where the user is aiming their nose, but off to the side.

We have adopted a combined approach that uses both the original template and the template from the previous image to track the face accurately through appearance changes, but also resist drift.

Our approach is to first search for the regions which match the appearance of the face tracking region in the previous image, then take the best candidates (those with the highest match score) and compare them to the original template. This results in only a small increase in computational complexity, but significantly improves performance. Tracking remains "locked on" through a much larger range of rotation and lighting changes, but drift is essentially eliminated.

As template matching is the most computationally expensive aspect of this system, we constrain the search to parts of the image where we can expect to find the face. The size and location of the head, estimated during training, are used to determine the region of the image where the head could possibly move to. We also take advantage of the location of the face in the previous frame, and an estimate for the maximum speed a head can move.

3.4 Transfer function

Once the location of the face is identified, its location and/or movement must be turned into a location or movement of the pointer. For face tracking, one of two techniques is appropriate. In the first, the absolute position of the face with respect to some reference frame determines the absolute position of the pointer on the screen (Position Control). In the second, the movement of the face is translated into a corresponding motion of the pointer (Rate Control). Cross control methods, such as when the location of the face controls the motion of the pointer, are generally not suitable for facial pointing because of limits on the range of head movement and because head position affects where the user can comfortably look.

From empirical evaluation of several current vision-based head tracking systems, most appear to use a rate control approach. This may be because rate control places fewer demands on the tracking algorithm. With rate control, the tracker only need identify motion in some portion of the image. If the exact region being tracked drifts from one part of the face to another, there is little effect on performance. With position control the tracker must remain accurately locked onto the same features on the face.

The disadvantage of rate control is that the apparent location of the pointer with respect to head position "slips" quite easily, often with just a few seconds of use. Systems using rate control usually provide a method to re-center the pointer, often by "pushing" against the edge of the screen, but this action must be performed very regularly by the user to keep the pointer in a usable position with respect to the head. Position control systems generally do not suffer from this problem, the pointer will stay in the same position with respect to the face as long as the tracking algorithm does not drift.

With either rate or position control, there must be a Transfer Function that determines exactly how the facial input is converted to pointer movement. This function can have a large effect on the usability of the pointer. In previous work, the transfer function is generally a linear mapping from head input to pointer output. A filter, often a predictive filter such as that found in [7], is applied to either the head or the pointer position to reduce the effect of noise and effectively increase pointer resolution. Although these filters can reduce noise, simple filtering does nothing to accommodate the needs of the domain. The following list attempts to specify the attributes of vision-based body tracking pointer control that the transfer function must take into account.

First are the constraints imposed by the physical system:

- The required pointer accuracy is higher than either the resolution of the camera, or the accuracy with which users can position their heads.
- Estimates of the user's head position are noisy by the very nature of computer vision. For users with erratic head control this problem is exacerbated.

Some additional constraints must be addressed in order to ensure the pointer is responsive to both large and small movements. Pointing movements tend to have two phases, an initial ballistic phase that puts the pointer in the vicinity of the target, followed by a series of short movements performed in a visual feedback loop to fine position the pointer [10]. Therefore:

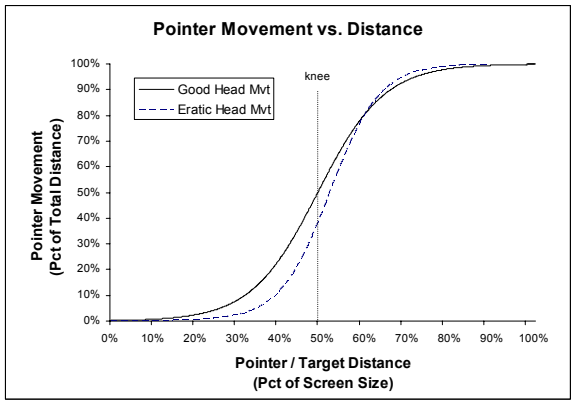
- On long movements the pointer must track the head quickly with little lag. This tight coupling between head and pointer is important so the user can begin the fine tuning phase as soon as their head is aimed at the target. Lag has been shown to slow down pointing significantly [8]. Positional accuracy is not important because long movements are followed by the fine tuning phase.
- On short movements the pointer movement must be slower and more stable so that the user can fine tune the position in a tight feedback loop with the system. Head movements must result in relatively smaller pointer movements so that the user need not use impossibly small head movements to move the pointer a character or two. Smoothness and stability are important

because jitter has been shown to slow down accurate positioning [6].

Finally, some constraints should be imposed to make the pointer movement more pleasing to the user.

- When the user stops moving their head, the pointer should smoothly come to a complete stop. This is important to support accurate dwell clicking, as well as for user comfort. If the pointer jumps around when the user's head is still, it can be very disconcerting.
- The user must be able to look as directly as possible at the pointer while they are positioning it. While this sounds obvious, it is an easy condition to violate.

Because of the problems with rate control, we chose to use a position control algorithm for pointer movement. To address the constraints identified above, we have developed a transfer function based on a modified sigmoid. Initially absolute head position determines a target location on the screen, then the sigmoidal function is used to move the pointer toward the target.



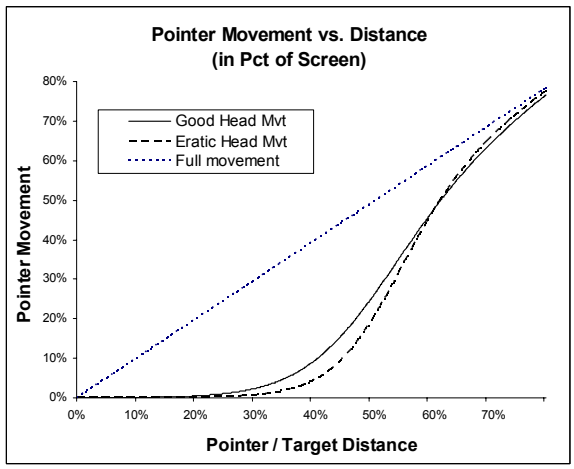


Figure 5: Two views of the modified sigmoid function used to smooth the pointer motion. The solid line shows the function for users with good quality head movement. The dashed line shows the function modified for users with erratic head movements.

The initial target location on the screen is determined using the displacement of the user's face from the reference face location. The reference location is determined during training when the user is asked to aim at the center of the screen. The current displacement from that location is computed in "face width" units (face width, again, is determined during training), and multiplied by a ScreenWidthRatio parameter to determine the screen location. The ScreenWidthRatio specifies the number of "screen widths" of pointer movement for each "face width" of head movement. The user has access to this parameter to tune how much face movement is required to move the pointer.

Once a target location has been determined, the pointer is moved toward that target. The percentage of the distance the pointer moves each frame is determined from the distance between the current pointer location and the target. When the two are far apart, the pointer moves most of the way toward the target. As the two get closer, the pointer moves less of the total distance. The percentage of the distance moved is varied smoothly with distance using a modified sigmoid function as follows.

The distance D moved by the pointer toward the target is computed each frame using the relationship

$$D_i = \frac{\Delta_i}{1 + e^{\left(\frac{\kappa - \Delta_i}{\mu}\right)}}$$

where $i \in \{x, y\}$. $\Delta_i = T_i - P_i$ is the distance between the current pointer location P and the target location T. $0 \le \kappa \le 1$ and $0 \le \mu \le 1$, called the knee and slope, respectively, of the sigmoid, control the shape of the relationship.

Figure 5 shows two views of the sigmoidal relationship between $\Delta = T - P$ and the pointer movement D. In the top view, notice that as the distance between the pointer and the target increases, the percentage of that distance traveled by the pointer increases as well. The bottom view shows the same relationship plotted slightly differently. Here, the fine dotted line represents what would happen if the pointer always went the full distance to the target. You can see that for small displacements the pointer only goes a small fraction of that distance each cycle, while at

large displacements it goes to the target almost immediately.

The shape of the sigmoid can be adjusted to adapt the pointer to the head movement of different users. For users with erratic head movements, both μ and κ have been increased to reduce the effect of their unintentional head movement on pointer position.

One advantage of this function is the smooth transitions between the different regions in the curve. Some systems simulate a similar relationship using "pointer acceleration", where the pointer is sped up when pointer speed passes a threshold. When these non-linear relationships are simulated using thresholds, the junctions are abrupt making pointer movement change suddenly. The smooth transitions provided by the sigmoid make it easier for the user to anticipate the pointer movement, which is important for accurate control.

The movement in x and y are computed separately because that gives the pointer movement a small amount of affinity for straight vertical and horizontal motions. This characteristic is handy for many interactions in a modern GUI, such as moving along the characters in a line or traversing nested sub-menus. It should be pointed out that this characteristic actually decreases the accuracy of pointer tracking, making free-form tasks, such as drawing a smooth curve, more difficult. It is a simple matter to use the same approach on the straight line distance between *T* and *P*, in effect creating a "drawing mode" for the pointer.

4 PERFORMANCE

The HeadTracking Pointer is currently undergoing formal field trials with two groups of users, one with Cerebral Palsy and one with spinal cord injuries. To this point it has been used informally for some time by both quadriplegic and fully-abled users. To date, one brief test with 5 fully-abled users has been done, comparing HTP and the demonstration version of CameraMouse[1]. CameraMouse was chosen because it uses standard filtering techniques to turn head movement into pointer movement, and so make a good comparison to the sigmoidal filter used in HTP.

The users were asked to use a standard on-screen keyboard to type phrases of from 12 to 15 characters long using each system. The time to complete each phrase was measured. The typing task was chosen as it required accurate selection of small screen regions at various distances and locations. The users were allowed to use a system until they felt comfortable with it before the timing was begun. The systems were presented to the subjects in random order. One subject had prior experience with both systems, the other subjects had never used head tracking pointers before.

Using CameraMouse the average length of time to type the phrases was 72 seconds. Using HTP, the average length of time was 56 seconds. Asked their opinions, the users indicated that the pointer motion was less erratic with HTP than with CameraMouse. They also commented that the pointer was easier to keep still with HTP. While in general they liked the quality of the HTP pointer motion better, they reported that it took a little while to get used to controlling the pointer accurately because it tended to drift slowly to a stop after they had stopped moving their head. Stopping the pointer immediately took a slight head motion in the opposite direction, which took some time to learn.

HTP is currently in field test with several users with Cerebral Palsy. The results are not yet available, but they have begun using the system as part of a trial of a web browsing system [4], and there have been some interesting observations during our efforts to adapt the system to their needs.

The initial problem was to enable the user to successfully train the system with the head tip motion. With spinal cord injury patients, it was not uncommon for a user to be unable to use a physical pointing device, but still have good head and neck control. With the cerebral palsy population, often a user who is severely enough affected to not be able to use a mouse or track ball also has limited or irregular head movement. The original training algorithm assumed that the users' head movement would be symmetrical and relatively smooth during the head tipping motion. With the CP users, this assumption was violated. To be successful with the CP patients, head tip recognition had to be adjusted to be more forgiving of asymmetrical and irregular head tip motions.

A second problem came about because the CP population seemed to be more prone to erratic head motions while using the pointer than we expected. This was addressed by tuning the shape of the sigmoid filter function. As shown in figure 5, for users with erratic head movements, both μ and κ can be increased so that moderate sized head motions have less effect on pointer motion. The low gain region at the left of figure 5 is extended so that the pointer is motionless, or very slowly moving for relatively larger head movements. The upper region is similar to the original function, so that large head movements are still tracked immediately. The key difference is in the important center section of the function, where the user is trying to position the pointer slightly to the left or right to select an icon or other object. Here the pointer motion is lower compared to the original curve so that more of the erratic head movement is damped out, while still allowing the pointer to follow trends in head movement. The pointer becomes less responsive to the small spastic head movements, while still responding smoothly to trends in head position. Importantly, this is achieved while large

pointer movements are still tracked quickly, so the user does not have to wait for the pointer to travel long distances across the screen.

We make this control available to the user with a single parameter that varies μ and κ together. This "sensitivity knob" allows the user a single, simple to understand adjustment to tune the system to their level of head control. In future work it may be interesting to automatically tune this parameter based on an analysis of the steadiness of the user's head movements.

5 CONCLUSION

This paper has described a vision-based head tracking pointer which addresses several of the limitations of similar systems. Specifically, we have given the user increased control over the operation of the system by providing symbolic head gesture recognition. The path for recognizing these gestures is independent of the standard head tracking algorithm, so that end users signal the system that head tracking is not working properly without asking for help from other individuals. We believe this extra level of autonomy is a valuable addition. In future work it would be interesting to extend these symbolic head signals to other control tasks.

HTP also demonstrates a novel head tracking algorithm that is robust to lighting changes, and resistant to the drift often associated with head tracking. This allows users to get consistent behavior from the system.

Finally we have described in some detail a novel transfer function for converting head motion into pointer motion. This algorithm has several improvements over standard filtering approaches that make it easier for the user to control the pointer. The pointer remains more stable in the face of noise and inadvertent user movement, while allowing smooth and highly accurate pointer control. It allows the user to smoothly stop the pointer without having it jump around. These characteristics make for a more pleasing user experience and initial results indicate it may have better usability as well.

(A demonstration version of HTP is available at www.alphaworks.ibm.com/tech/headpointer)

REFERENCES

[1] Betke, M., Gips, J. and Fleming, P. The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access For People with Severe Disablilities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*. April 2002.

[2] Fast B. N. Waber, J. J. Magee, and M. Betke. Fast Head Tilt Detection for Human-Computer Interaction. *Proceedings of the ICCV Workshop on Human Computer Interaction.*, Beijng, China, October 2005. Springer Verlag.

[3] Gorodnichy, D., Malik, S., and Roth, G. Nouse 'Use your nose as a joystick or a mouse' - a new technology for handsfree games and interfaces. in *Proc. Intern. Conf. on Vision Interface* (VI'2002), Calgary, May 2002.

[4] Hanson, V. L., Brezin, J., Crayne, S., Keates, S., Kjeldsen, R., Richards, J. T., Swart, C.,Trewin, S. accessibilityWorks: Web Access for an Open Source Browser, *IBM Systems Journal* Vol. 44, No. 3

[5] Ballard, D.H., Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition*, Vol. 13, Issue 2, 1981, pg. 111-122

[6] Kjeldsen, F. , *Visual Recognition of Hand Gesture as a Practical Interface Modality*, PhD thesis,Department of Computer Science, Columbia University, 1997

[7] Kohler, M., Using the Kalman Filter to Track Human Interactive Motion – Modeling an Initialization of the Kalman Filter for Translational Motion. Technical Report 629, Informatik VII, University of Dortmund, January 1997.

[8] MacKenzie, I.S., and Ware, C., Lag as a Determinant of Human Performance in Interactive Systems, in *Proceedings of INTERCHI '93*, Amsterdam, April 1993.

[9] Matsumoto, Y., Ogasawara, T., and Zelinsky, A., Behavior Recognition Based on Head Pose and Gaze Direction Measurement, in *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*

[10] Meyer, D. E., Smith, K. J. E., Kornblum, S., Abrams, R. A., Wright, C. E., Speed-Accuracy Tradeoffs in Aimed Movements: Toward a Theory of Rapid Voluntary Action. in *Attention and Performance XIII*, M. Jeanerod, Ed. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1990 pp. 173-226.

[11] *Proceedings of the 7th IEEE International Conference on Automatic Face and Gesture Recognition* (FGR 2006), April 2006, Southampton, United Kingdom.

[12] Toyama, K.. 'Look, Ma – no hands!' Hands-free cursor control with real-time 3D face tracking. in *Proceedings of the Workshop on Perceptual User Interfaces* (PUI'98), San Fransisco, November 1998.