2006-05-11

A Customizable Camera-based Human Computer Interaction System Allowing People With Disabilities Autonomous Hands Free Navigation of Multiple Computing Task

Boston University Technical Report 2006-006

A customizable camera-based human computer interaction system allowing people with disabilities autonomous hands-free navigation of multiple computing tasks

Wajeeh Akram, Laura Tiberii, and Margrit Betke

Department of Computer Science, Boston University
111 Cummington Street, Boston, MA 02215, USA
{wajeeha, ltiberii, betke}@cs.bu.edu

Abstract. Many people suffer from conditions that lead to deterioration of motor control making access to the computer using traditional input devices difficult. In particular, they may loose control of hand movement to the extent that the standard mouse cannot be used as a pointing device. Most current alternatives use markers or specialized hardware, for example, wearable devices, to track and translate a user's movement to pointer movement. These approaches may be perceived as intrusive. Camera-based assistive systems that use visual tracking of features on the user's body often require cumbersome manual adjustment. This paper introduces an enhanced computer vision based strategy where features, for example on a user's face, viewed through an inexpensive USB camera, are tracked and translated to pointer movement. The main contributions of this paper are (1) enhancing a video based interface with a mechanism for mapping feature movement to pointer movement that allows users to navigate to all areas of the screen even with very limited physical movement and (2) providing a customizable, hierarchical navigation framework for human computer interaction (HCI). This framework provides effective use of the vision-based interface system for accessing multiple applications in an autonomous setting. Experiments with several users show the effectiveness of the mapping strategy and its usage within the application framework as a practical tool for desktop users with disabilities.

Keywords: Computer-vision, assistive technology, alternative input devices, video-based human-computer interfaces, autonomous navigation.

1 Introduction

Several conditions may cause computer users to be unable to use the standard mouse. Paralysis from brain injury, stroke, multiple sclerosis, or Amyotrophic Lateral Sclerosis (ALS, also called Lou Gehrig's disease) may cause the user to have very little motor control except for limited head or eye movement. Loss of fine motor

control with age and muscle injuries may also make use of the standard mouse difficult.

According to the National Multiple Sclerosis Society [1], approximately 400,000 Americans and 2 million individuals worldwide suffer from Multiple Sclerosis, and about 200 people are diagnosed every week in the US. As such conditions restrict physical mobility and often speaking capability, loss of the ability to communicate is one of the most limiting problems for these individuals. Being able to use computers for common tasks such as sending email and browsing the web opens a huge avenue of possibility to improve quality of life.

A study by Forrester Research for Microsoft Corporation [2] presents statistics on the need and significance of accessible technology. It is estimated that about 17% (22.6 million) of computers users who suffer from severe impairments are very likely to benefit from accessible technology. It is also postulated that the need for accessibility devices may grow due to the increase in computer users above the age of 65 and the increase in the average age of computer users.

There has been extensive research in the domain of mouse alternatives as accessibility aids for users who have very limited movement. Broadly, these efforts can be divided into two main categories: systems that rely on specialized mechanical or electronic hardware devices and camera-based systems. Mouse-actuated joysticks, mechanical switches, breath-puffing straws, and electrodes placed on the user's face that measure movement of features are some of the strategies in the first category [3]. Many camera based systems track physical markers, for example, infrared markers placed on the user's body [4, 5] or markers on glasses. Systems that capture gaze information often rely on infrared illumination or special headgear-mounted cameras; a survey of these methods is provided by Magee et al. [6]. Most of these systems are expensive, require special devices, and may be intrusive. In addition, significant levels of technical expertise may be required to install and configure these systems. Betke et al. [7] presented a vision based solution called the camera mouse which tracks features on a user's body in a non-intrusive manner.

There has also been substantial work in developing applications for people with disabilities [8, 9, 10]. Some existing applications include on-screen keyboards [11], alternate text entry mechanisms [12, 13], games and learning aids for children [7], and tools that interact with a web browser to make the internet more accessible for camera mouse users [14, 15].

In this paper, we present a system that tracks features on the user's body, usually the face, and translates feature movement to pointer movement on the screen. Our work builds on the camera mouse presented by Betke et al. [7], which proposed a vision based feature tracking approach for pointer movement. Here, we present an improved mapping strategy that allows translation of minimal feature movement to pointer movement across the entire range of the screen. A framework for using the camera mouse to carry out common tasks, with minimal intervention from a caregiver, is also proposed. Experiments were conducted to determine how well the users were able to access and perform each of the computing tasks in the HCI framework. Test results have shown that the system successfully provides access to common tasks such as opening games, web sites, text entry, and playing music.

The system is cost effective and requires little technical expertise of the user and caregiver. Use or extension of the proposed system does not incur significant

cost, because the system was developed with open source technologies such as OpenCV [16] and Java. The only additional hardware required, besides a personal computer, is a low-cost USB camera. We refer to the interface system as the camera mouse throughout this paper. However, as an alternative to the camera mouse [7], any interface system, video-based or even the standard computer mouse that provides a pointing and selection mechanism can be used with our HCI framework.

2 System Overview

The goal of our work is to provide a customizable camera-based human computer interaction system allowing people with disabilities autonomous hands free navigation of multiple computing tasks. We focus on two main aspects of the system; designing a robust feature tracking strategy and an effective interaction approach that operates optimally with a camera mouse. The following sections give an overview of the components of the system.

2.1 Tracking features

This section describes our method to track a feature or set of features on the user's body, usually face, and convert the feature movement to pointer movement. The study by Fagiani et al. [18] gives an experimental comparison of various tracking mechanisms for use with the camera mouse and recommends either an optical flow or correlation based tracker. We found the optical flow based algorithm to be both robust and computationally efficient. Our system operates in real time on a computer with a 1.6 GHz processor, taking up on average less than 5% of processor time. This demonstrates the use of the camera mouse as a background process that does not affect the performance of other applications running on the system. Our camera mouse implementation executes as a standalone application that moves the standard windows pointer.

A USB Camera is connected to the computer and set up to capture a frontal view of the user. On starting the application, a window with the video of the user is displayed. The camera location should be adjusted so that the feature to be tracked is in clear view. Typically, the user sits within 1 m of the camera. However, if the user is very close to the camera, even a small physical movement can result in the feature falling out of the camera's field of view. Therefore, the distance from the camera should be carefully adjusted such that the feature remains within the camera's field of view throughout the session.

The caregiver selects a feature on the user's body by clicking at the desired location of the input video stream. We designed the system to automatically refine the feature location by finding an image patch with the highest brightness gradient in the 11-by-11-pixel neighborhood of the manually selected feature [16]. The feature is then tracked in subsequent frames using the Lucas-Kanade optical flow computation [17]. We used a pyramid-based implementation of the Lucas-Kanade tracker provided in Intel's OpenCV library [16].

2.2 Feature movement to pointer movement

Once the feature movement in pixels is known, an effective mapping from pixels of movement in the video frames to pointer movement on the screen is required. Pointing devices such as the standard mouse and mouse pad do not have an absolute mapping of device movement to pointer movement. The pointer is moved in a differential manner, governed by speed and acceleration parameters set by the user. Similarly, the camera mouse cannot be used with any degree of flexibility if this mapping is absolute: an absolute mapping would mean that the feature to be tracked would have to move the same distance (in pixels, as viewed by the camera) as the pointer is to move on the screen. Most users do not have such a large range of movement and even if such movement were possible, it does not complement the natural movement of a computer user as they view the computer screen. Therefore the camera mouse operates the pointer in a relative manner.

A relative scheme of pointer movement must consider how to adjust for the difference in scale of feature movement and pointer movement. The movement of the detected feature must be scaled in some reasonable manner before being added to the current pointer position. In previous systems, the scale factor is a user-customizable setting. However, adjusting the scale factor manually is a cumbersome trial and error process and requires intervention by a caregiver for manually entering scale factors. The scale factor is pertinent to the usability of the system, because if the scale factor is too low, all areas of the screen may not be reachable by the pointer. Alternatively, if it is too high the pointer may become too sensitive and thus move too quickly.

It can be observed that the scale factor is a function of the user's distance from the screen, as well as the range of possible movement of the feature in both horizontal and vertical directions. The user's range of movement may be seriously limited by motor dysfunction. The range of movement is also typically asymmetric in the vertical and horizontal directions due to the fact that vertical rotation of the head when viewing a standard computer screen is smaller than horizontal rotation.

From a usability point of view, the scaling factor should not be such that the system requires the user to move in a way that interferes negatively with the user's visual focus on the screen. In other words, during facial feature tracking with the camera mouse, feature movement and visual focus cannot be decoupled. Feature movement required for effective use of the system should not be such that it causes a strain on the visual focusing mechanism of the user.

Designing a mechanism to allow optimal setting of the scale factor by the user is therefore important towards the end of improving system performance and usability. A calibration phase was introduced to determine the optimal scale factor for individual users. Calibration is performed in advance of a usage session. After a feature is selected to be tracked, the users are lead through a calibration phase, in which they are directed to rotate their head towards distinct markers shown on the video stream, while retaining a comfortable view of the computer screen. The users successively move towards markers on the top, bottom, left and right boundaries of the screen (Figure 1). It is important to direct users to move within a comfortable range of motion, which permits clear and non stressful visual focus on the screen. Pointer movement is calibrated to the range of movement demonstrated by the user, using a linear mapping of demonstrated movement range to screen dimensions.

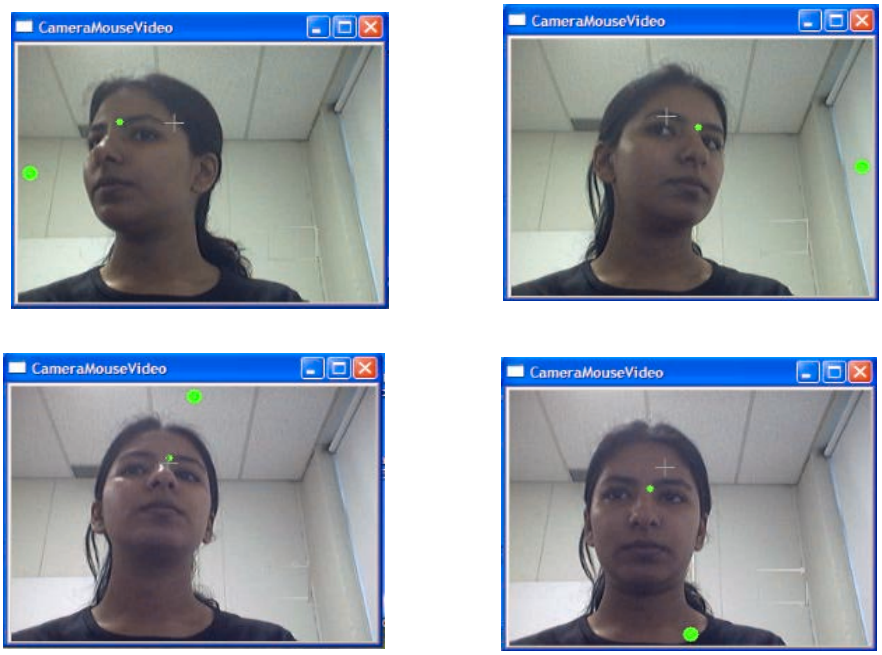

Figure 1: System Calibration: The small colored disk shown in the video indicates the tip of the eyebrow has been selected as the feature to track. The larger disk on the boundary of the video display window indicates the direction the user should move her head.

After performing the calibration phase once for a particular user and a specific feature, in situations where the distance from the camera remains approximately the same across sessions, for example, for a user in a wheelchair, the scale factors found by the calibration phase may be saved in a user configuration file that can be loaded for subsequent use.

2.3 Application framework

Applications often have to be tailored to work with the camera mouse, since the effective movement resolution of the camera mouse is not enough to navigate windows menus or operate standard windows applications. Several on-screen keyboards, educational programs, and game applications are available for use with the camera mouse. However, the user must rely on a caregiver to start the custom application before they can start using it. If the user wants to start a new application for another task, there is no means to navigate available programs autonomously without the caregiver's help. Our motivation in proposing a hierarchical framework for application navigation is to provide the camera mouse user with an autonomous experience with their computer, allowing them to perform common tasks of interest

such as text entry, internet browsing, and entertainment applications in a manner that is user friendly, requires little technical expertise, and is configurable to meet the needs of individual users.

Several considerations must be kept in mind when designing an effective interface [19].

- The user should be able to clearly identify the target being selected.
- Distinguishing information should be placed at the beginning of headings.
- Clear and simple language should be used.
- The design should be consistent.
- There should be clear navigation.

Our interface opens with a main menu that is a list of common tasks (Figure 2). The main menu items configured in the test system are: Play this Song launches the default media player and plays the chosen song, Text Entry launches an on-screen keyboard, Common Sayings speaks saved text using a speech synthesis program, View a webpage launches the default browser and displays the chosen website, and Games launches games, such as Eagle Aliens [6], which have been developed to require only pointer movement.

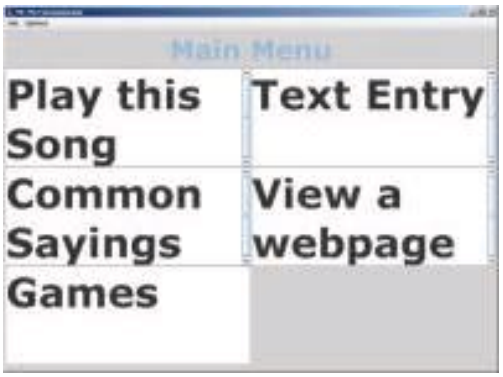

Figure 2: Main Menu of Interface.

The list of common tasks desired in the application varies depending on the interests of each user. The system is designed so that menu items can be added, removed, or modified. This allows the list to be customized for each individual user.

The user will choose the common task they desire in one of two modes, select mode or scan mode. In select mode, the user moves the pointer to an item. When the pointer reaches an item it is highlighted in blue, clearly identifying the target to be selected. In scan mode, the application scans through the list of items highlighting each item for a specified time interval. The time interval can be changed to the length of time that is reasonable for the current user.

To facilitate autonomous use, a dwell feature is available to simulate a selection command. The dwell feature acts as a timer. When an item is highlighted the timer is started. If that item stays highlighted for a specified time interval, a selection command is executed. The gray areas of the interface, shown in Figure 2, represent

rest areas where the pointer can dwell without causing a selection command to occur. Gray was used to stress the inactive nature of such areas. The dwell feature can be enabled or disabled, as alternate methods may be available to simulate pointer clicks, such as blink detection [22], raised eyebrow detection [23], or use of a mechanical switch.

The font size of the menu items was also a consideration for users who are unable to sit close to the system due to wheelchairs. The system is designed so that the font size can be increased or decreased as desired. Items on the main menu are either links that directly launch programs or links that open a submenu. Every submenu has the same font type and size. The same color is used to highlight the menu items. This consistency helps maintain usability. A 'Return to Main Menu' option is always the last item in the submenu list. This feature supports clear navigation among the various menus. When a submenu item is selected the program associated with that menu item is launched. The 'Return to Main Menu' option is displayed on the screen after the program is launched so that the user can return to the system and navigate to other programs if desired. A strategy for navigation among opened programs is proposed by our framework, but has not been implemented yet.

An example of navigating through the system and selecting a song to play is shown in Figure 3.

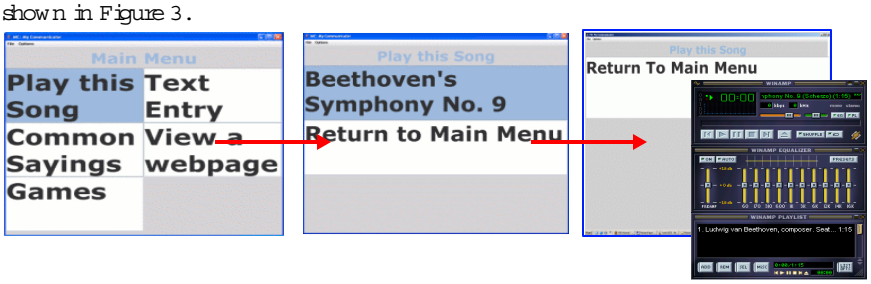

Figure 3: Navigation from the main menu through the 'Play this Song' submenu to launch a music player that automatically begins playing the selected song.

3 Experiments and Results

The system was tested to determine the performance of the tracking mechanism and to understand its limitations, as well as to determine the usability of the application framework proposed. Results from the first test provided input for the design of interface elements for the application framework.

A test group consisting of 8 subjects did the first set of experiments (Group 1). The subjects were between 14 and 60 years of age with varying levels of computer skills. The subjects did not have any functional limitations. The same set of users was asked to perform a control test, where the same sequence of steps was performed with a standard mouse (Control Group). The second test group (Group 2) consisted of two patients from The Boston Home [20]. Both subjects suffered from functional limitations that made it difficult or impossible to use the standard mouse. One of the

subjects was diagnosed with muscular dystrophy more than 15 years ago. His condition causes muscle weakness and wasting in major joints and muscles, his shoulders and hips have been affected most. The other subject was diagnosed with multiple sclerosis more than 15 years ago. His condition causes muscle weakness, limiting his ability to move his arms, hands, and neck. The limitation in neck movement has resulted in a very small range of head movement.

3.1 Evaluating tracker performance

The tests were designed to record indicators of tracker performance. Specifically, we focused on factors pertaining to the tracker's ability to track features and translate feature movement to pointer movement on the screen. Specific factors include:

- Effective Dwell Area: the smallest region within which the user can dwell for 3 seconds. This will allow us to study the tradeoff between tracker sensitivity and dwelling ability.
- Movement patterns that cause the tracker to lose features while tracking.
- Movement patterns that affect the smoothness of the tracker's constructed pointer movement.

A movement evaluation tool was developed to analyze the above factors (Figure 4). During the test, users were asked to move the pointer from box to box. The order of movement between boxes was chosen so that we could evaluate the user's ability to move the pointer vertically, horizontally, and diagonally. The placement of the boxes on the screen was chosen to allow us to determine if there were areas of the screen that the users found difficult to reach, or were unable to reach. Different sized boxes were used to evaluate the smallest area that the user can easily dwell in for a few seconds. The size and location of the boxes was chosen so as to discern if it was easier to dwell in smaller boxes in some areas of the screen. The use of color in the boxes allows the user to recognize the area they are asked to move without having to read through the labels.

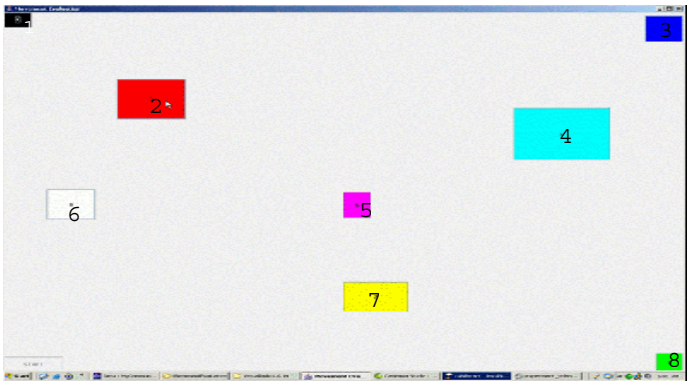

Figure 4: Movement Evaluation Tool.

The users were asked to move the pointer in the following sequence, dwelling for three seconds in each box: dark blue box labeled 3, yellow box labeled 7, green box labeled 8, red box labeled 2, light blue box labeled 4, black box labeled 1, purple box labeled 5, white box labeled 6.

Figure 5 shows a user with multiple sclerosis performing a subset of steps in the movement evaluation test. It is apparent from the test that despite being restricted to only slight movements of the head, the user was able to reach all areas of the screen, including corners, and could dwell even in small regions.

Figure 5: User with multiple sclerosis while performing movement evaluation test (left), simultaneous screen shots depicting pointer location (center), and the instruction given (right). (Note: Pointer is shown enhanced in the figure.)

Figure 6 shows the entire trajectory of pointer movement as a user performs the movement evaluation test.

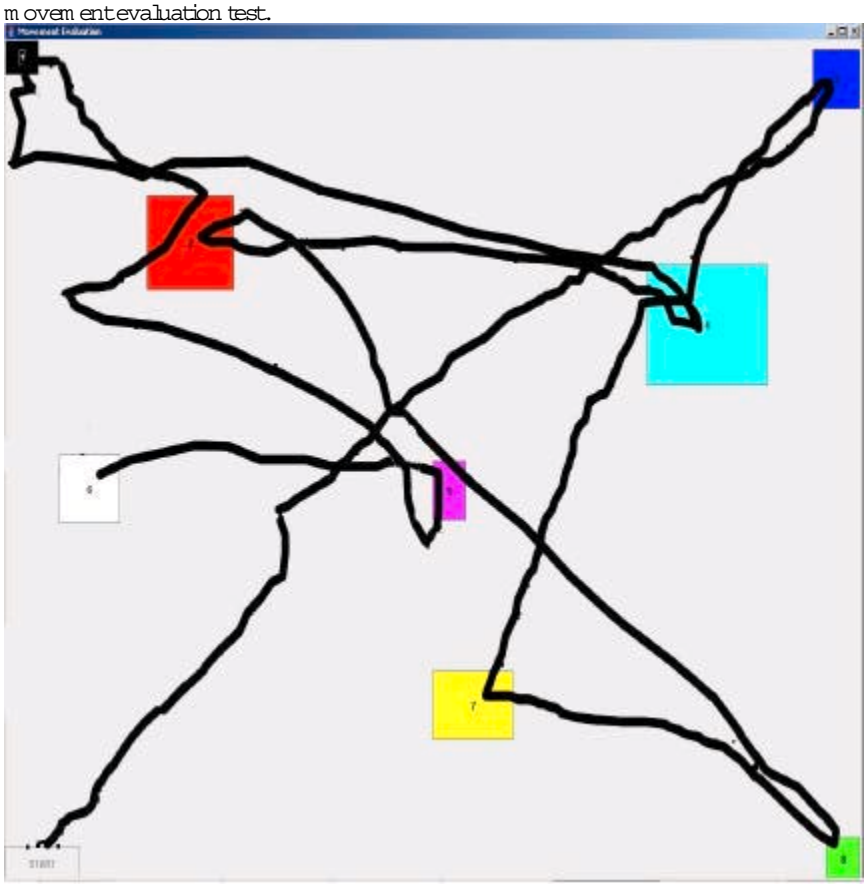

Figure 6: Pointer trajectory of the movement evaluation test.

The task in the tracker evaluation test was to move from one colored box to another (Figure 4) and then focus on the box for several seconds. The test consisted of eight tasks. The tracker evaluation tests showed that all ten users, with and without disabilities, were able to move the pointer to every location. This indicates that we were successful in designing a system that tracks features and translates feature movement to pointer movement on the screen. Table 1 categorizes three levels of movement error, no overshooting, overshooting once, and overshooting more than once. Overshooting occurs when the mouse pointer moves beyond the target on the screen. This did not prevent the user from selecting the desired target. The control experiment was done using the standard mouse.

Table 1: Results of Movement Evaluation Test

	Control	Group 1	Group 2
Average Completion Time	1.0 s	1.8 s	3.2 s*
Average % of Tasks Completed on the First Trial	8/8 = 100%	8/8 = 100%	7.5/8 = 94%
Average % of Not Overshooting	8/8 = 100%	4/8 = 50%	2/8 = 25%
Average % of Overshooting Once	0/8 = 0%	2/8 = 25%	2.5/8 = 31%
Average % of Overshooting more than Once	0/8 = 0%	2/8 = 25%	3.5/8 = 44%

* We discounted the timing result of one of the eight assigned tasks for one user in Group 2 in computing the average completion time. The reason was that, during the test, the subject was asking questions and the recorded time of 30 seconds did not reflect the actual time to move the pointer, which was on average less than 3 seconds for the remaining seven tasks performed by this user.

3.2 Evaluating application design

The tests in this section were designed to capture the usability of the application framework with respect to the design and layout of the interface elements. The test consisted of launching five applications in sequence, Text Entry (Keyboard application), Common Sayings (speech synthesizer), View a webpage (open browser), Games (open a game), and Play this Song (open a media player).

We were interested in determining how well users were able to navigate through the menus (average completion time), how many times the users had to try before they successfully launched the correct application (number of tasks completed on the first trial), and how often the programs were launched unintentionally (percent of unintentional launches). Table 2 presents the results.

Table 2: Application Evaluation Results

	Control	Group 1	Group 2
Average Completion Time *	5.0 s	6.3 s	9.4 s
Number of Tasks Completed on the First Trial **	1	5/5 = 100% for 5 users 4/5 = 80% for 2 users	5/5 = 100% for user 1 3/5 = 60% for user 2
Percent of Unintentional Launches	0	0/5 = 0% for 5 users 2/7 = 29% for 1 user 3/8 = 38% for 1 user	0/5 = 0% for user 1 4/9 = 44% for user 2

*Actual task completion times for Group 1 and 2 were not significantly different. The computed results for Group 2 were affected by the fact that users in Group 2 showed much interest in the system; they stopped to discuss, ask questions, and give ideas. Such instances skewed the average of the recorded times.
**The users needed more than one trial to complete a task due to unintentional launches. The unintentional launches were instances where the user diverted from the test to discuss something and hence caused unintentional launching of the applications. This forced them to return to the main menu and repeat the task. This also highlights the need for a binary switching mechanism to turn off the tracker when not in active use.

Another consideration for the application evaluation was the degree of independent use, i.e., the degree to which the user can effectively use the application without intervention, once it has been setup. This factor is difficult to measure quantitatively. From personal observation we saw that the subjects were able to launch all of the programs independently and interact with the applications. For example, using the cascading menu selection strategy, they were able to launch and play a game, get back to the main menu by hovering above it and then launch and use a text entry application.

The users were also provided with the opportunity to use the system on their own, without a guided sequence of steps. This helped determine their opinion on the overall use of the system. During this period unexpected problems with the system could be identified. A survey was used to gather the opinions of the sample test group.

Issues were determined by analysis of survey questions and by personal observation. The tests performed by Group 1 revealed several issues. It was observed that after a program was launched it was not possible to return to the application without using the standard mouse. To resolve this issue, the system was configured such that when the pointer moves over the title area of the partially occluded application, the application is brought into the foreground. This assumes that the programs opened will not take the full screen area.

Another issue noticed during preliminary testing was that the testers could not easily identify where to rest the pointer without causing a selection command to occur. As a result, programs were opened unintentionally; the Midas touch problem [21]. To resolve this issue, all areas where the pointer can rest were changed to have a gray background color distinguishing them from the areas with a white background that cause a selection command to be executed. The users of Group 2 also found that the pointer had some jitter, due to the increased sensitivity. We propose a simple averaging mechanism to solve this problem.

Users showed interest in the prospect of being able to write and save text and send email autonomously using the camera mouse. Current users rely on a separate application to enter the text and then the caregiver has to copy and paste the text into an email application to dispatch the email. Users also expressed interest in a system that allowed effective web browsing with the camera mouse.

4 Discussion

In summary, we developed a customizable camera-based human computer interaction system and showed that people with and without disabilities can complete multiple computing tasks with slight head movements. The improvements made to the camera mouse have resulted in a robust feature tracker and a calibration of feature movement to pointer movement that is specific for each individual user. Taking advantage of the features of the camera mouse, our interaction system was able to provide hands-free access to many common computing tasks. The test results show that users were able to successfully open all of the programs available in our system with only a small percentage of error. This provides evidence that we designed a user-friendly interface with an effective navigation strategy. Survey results obtained from the test subjects showed that their holistic experience of the system was positive and they especially enjoyed playing the games.

Several of the test subjects in the first group used the system more than once. Their ability to control the pointer movement and dwell in a selection area improved as quickly as the second use. This indicates that the difference in average completion time between the control experiment and the camera mouse experiment would be reduced if all subjects were given more time to become accustomed to moving the pointer with the camera mouse.

A possibility for extension is to provide automatic feature detection. This would eliminate the dependence of tracking performance on the manual selection of an appropriate feature. The type of features best suited for tracking with the camera mouse was studied by Cloud et al. [24], who suggested that the tip of the nose was a robust feature. Gorodnichy [25] also discussed the robustness of nose tracking. Our experiments with the camera mouse showed similar results. Features on the sides of the face were lost by the tracker frequently as they were occluded upon rotation of the head. The outer tips of the eyes and features on the outer boundaries of the face were similarly not suitable for tracking. Features that exhibited good contrast on the central band of the face, e.g. the inner tip of the eyebrows, the tip of the nose and the outer boundary of the top or bottom lip, were the best features to track with the camera positioned so that it has a frontal view of the person's face. Tracking a feature on the lips may however be problematic if the user speaks during use. Features on the eye were often lost during blinking. Also, experiments showed that if the user wore glasses, especially of a dark color, features on the glasses, such as the bridge of the glasses, were robust to track.

Directions for future work include:

- Providing an automatic feature detection method.
- Smoothing pointer jitter that resulted from the increased sensitivity.
- Navigation among the opened programs.
- Providing better internet browsing, text entry, and email programs.
- Designing interaction strategies that allow the camera mouse to be used with standard, non-specialized applications. For example, adding features such as generalized dwell that is decoupled from camera mouse enabled applications and operates with the desired dwell radius on the entire screen. To overcome

the limitation of small interface elements found in many standard applications, screen magnification could be used to magnify menus as the pointer hovers above them. A binary switch could then be provided to toggle to the magnified area and select menu items. A cursor lock could also be used to aid selection of small interface elements.

- Extension to usage scenarios within the ambient intelligence paradigm [26]. The computer vision strategy presented here as a pointer alternative can be applied to menu selection tasks in common appliances such as telephones, microwave ovens, web-enabled digital television (DTV) and CD players.

Acknowledgements

The authors thank David Young-Hong from The Boston Home for his help with the experiments and for sharing his insights regarding technologies needed for people with disabilities. This work was supported by NSF grants IIS-0093367, IIS-0329009, and 0202067.

References

1. National Multiple Sclerosis Society, http://www.nationalmssociety.org, accessed April 2006.
2. Microsoft Accessibility, http://www.microsoft.com/enable/research/agingpop.aspx, accessed April 2006.
3. J. Gips, P. Olivieri, and J.J. Tecce, "Direct Control of the Computer through Electrodes Placed Around the Eyes", Human-Computer Interaction: Applications and Case Studies, M. J. Smith and G. Salvendy (eds.), Elsevier, pages 630-635. 1993.
4. Synapse Adaptive, http://www.synapseadaptive.com/prc/prchead.htm, accessed April 2006.
5. NaturalPoint SmartNAV, http://www.naturalpoint.com/smartnav/, accessed July 2006.
6. J.J. Magee, M. R. Scott, B. N. Waber and M. Betke, "EyeKeys: A Real-time Vision Interface Based on Gaze Detection from a Low-grade Video Camera," In Proceedings of the IEEE Workshop on Real-Time Vision for Human-Computer Interaction (RTV4HCI), Washington, D.C., July 2004.
7. M. Betke, J. Gips, and P. Fleming, "The camera mouse: Visual tracking of body features to provide computer access for people with severe disabilities", IEEE Transactions on Neural Systems and Rehabilitation Engineering, 10:1, pages 1-10, March 2002.
8. D.O. Gorodnichy and G. Roth, "Nouse 'Use your nose as a mouse' perceptual vision technology for hands-free games and interfaces", Proceedings of the International Conference on Vision Interface (VI2002), Calgary, Canada, May 2002.
9. Assistive Technologies, http://www.assistivetechnologies.com, accessed April 2006.
10. Apple Computer Disability Resources, http://www.apple.com/accessibility, accessed April 2006.
11. WiViK on-screen keyboard (virtual keyboard) software, http://www.wivik.com, accessed April 2006.
12. The Dasher Project, http://www.inference.phy.cam.ac.uk/dasher, accessed April 2006.
13. J. Gips and J. Gips, "A Computer Program Based on Rick Hoyt's Spelling Method for People with Profound Special Needs," Proceedings International Conference on Computers Helping People with Special Needs (ICCHP 2000), Karlsruhe, pages 245-250.
14. B. N. Waber, J.J. Magee, and M. Betke, "Web Mediators for Accessible Browsing," Boston University Computer Science Department Technical Report BUCS 2006-007, May 2006.

15. H. Larson and J. Gips, "A Web Browser for People with Quadriplegia." In Universal Access in HCI: Inclusive Design in the Information Society, Proceedings of the International Conference on Human-Computer Interaction, Crete, 2003, C. Stephanidis (ed.), Lawrence Erlbaum Associates, pages 226-230, 2003.
16. OpenCV library. http://sourceforge.net/projects/opencvlibrary, accessed April 2006.
17. B D. Lucas and T. Kanade. "An iterative image registration technique with an application to stereo vision." In Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI), pages 674-679, Vancouver, Canada, April 1981.
18. C. Fagiani, M. Betke, and J. Gips, "Evaluation of tracking methods for human-computer interaction." In Proceedings of the IEEE Workshop on Applications in Computer Vision (WACV 2002), pages 121-126, Orlando, Florida, December 2002.
19. "Human-centered design processes for interactive systems," International Organization for Standardization ISO 13407, 1999.
20. The Boston Home, http://www.thebostonhome.org, accessed April 2006.
21. R J.K. Jacob, "What you look at is what you get," Computer, 26:7, pages 65–66, July 1993.
22. M. Chau and M. Betke, "Real Time Eye Tracking and Blink Detection with USB Cameras," Boston University Computer Science Technical Report 2005-012, May 2005.
23. J. Lombardi and M. Betke, "A camera-based eyebrow tracker for hands-free computer control via a binary switch", In Proceedings of the 7th ERCIM Workshop, User Interfaces For All (UI4All 2002), pages 199-200, Paris, France, October 2002.
24. R. L. Cloud, M. Betke, and J. Gips, "Experiments with a Camera-Based Human-Computer Interface System." In Proceedings of the 7th ERCIM Workshop "User Interfaces for All," UI4ALL 2002, pages 103-110, Paris, France, October 2002.
25. D O. Gorodnichy, "On importance of nose for face tracking", In Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FG 2002), pages 188-196, Washington, D.C., May 2002.
26. A. Ferscha, "Contextware: Bridging Physical and Virtual Worlds." In Proceedings of the Ada-Europe Conference on Reliable Software Technologies, 2002.