Blink and Wink Detection for Mouse Pointer Control

Eric Missimer and Margrit Betke
Image and Video Computing Group
Computer Science Department
Boston University
{missimer, betke}@bu.edu

ABSTRACT

A Human-Computer Interaction (HCI) system that is designed for individuals with severe disabilities to simulate control of a traditional computer mouse is introduced. The camera-based system monitors a user's eyes and allows the user to simulate clicking the mouse using voluntary blinks and winks. For users who can control head movements and can wink with one eye while keeping their other eye visibly open, the system allows complete use of a typical mouse, including moving the pointer, left and right clicking, double clicking, and click-and-dragging. For users who cannot wink but can blink voluntarily the system allows the user to perform left clicks, the most common and useful mouse action. The system does not require any training data to distinguish open eyes versus closed eyes. Eye classification is accomplished online during real-time interactions. The system had an accuracy of $8027/8306 = 96.6\%$ in classifying sub-images with open or closed eyes and successfully allows the users to simulate a traditional computer mouse.

Categories and Subject Descriptors

H.1.2 [**User/Machine Systems**]: Human factors

General Terms

Human Factors

Keywords

Assistive technology, video based human computer interface, eye image analysis, mouse replacement system

1. INTRODUCTION

In recent years, there has been an effort to design assistive technology that provides individuals with severe disabilities a tool for communication and access to the computer. Such technology may augment traditional human-computer interfaces like the keyboard and mouse. These traditional human-computer interfaces demand good manual dexterity and refined motor control, which may be absent or unpredictable for people with severe disabilities.

The motivation of our research is to provide an alternative communication tool for non-verbal individuals whose motor abilities are extremely limited by conditions ranging from traumatic brain injuries to degenerative diseases such as multiple sclerosis (MS), muscular dystrophy (MD), or amyotropic lateral sclerosis (ALS). These individuals may only be able to control their head and eyes. Our goal was to develop a computer vision system that replaces the traditional computer mouse with a system that can be completely controlled with the head and eyes.

We propose an algorithm that allows a user to interact with the computer by using their eyes to simulate clicking a traditional mouse. The algorithm is able to automatically locate the user's eyes and learn the appearance of the user's open and closed eyes. Online learning provides a level of robustness that allows the algorithm to work consistently for various individuals and has also shown success for individuals wearing glasses.

Work on camera-based blink detection has focused on specific tasks such as human-computer interaction [4, 7] or fatigue detection [5]. Blink detection modules have been part of more general systems on eye motion analysis [1, 10, 11, 12]. Some research efforts in camera-based blink detection use infrared lighting [8, 13]. The advantage of an infrared system is that the pupils of the user are highlighted when exposed to infrared lighting. While infrared systems make the problem of detecting the eyes easier, the typical user does not have access to infrared lighting and there are safety concerns about long-term exposure to infrared lighting. Our system uses standard lighting with a typical USB camera that is easily available to users. Other systems [1] use active appearance models to locate and track the eyes of the user and make assumptions about the shape, color, and lighting of the user's eyes. The online templates of the users' eyes that our system automatically captures eliminate the need for us to make such assumptions.

Previous interaction systems for people with disabilities [4, 7] interpret a user blink as a trigger for binary switch applications. By tracking and interpreting both eyes of the user (for users who are capable of winking both eyes), our system allows interaction with a computer on a level that is closer to using a traditional mouse. Our system enables users to move the mouse pointer on the screen and issue mouse-clicking commands hands-free. For individuals who are not able to control the muscles around their eyes to a

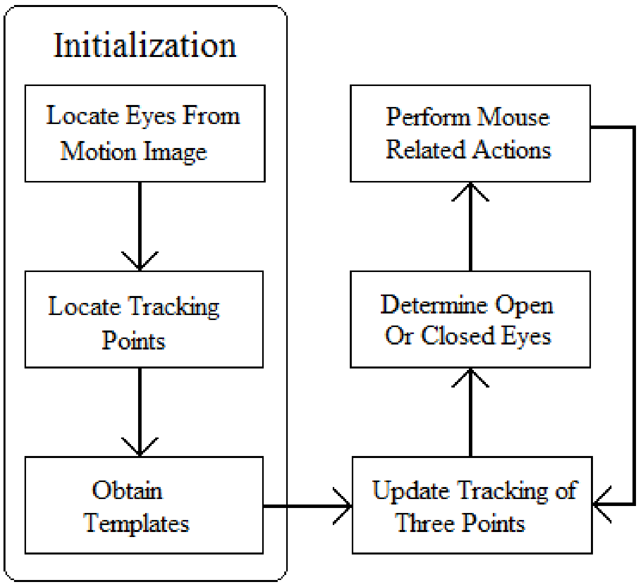
Initialization
Locate Eyes From Motion Image
Locate Tracking Points
Obtain Templates
Update Tracking of Three Points
Determine Open Or Closed Eyes
Perform Mouse Related Actions

Figure 1: Flowchart of Algorithm.

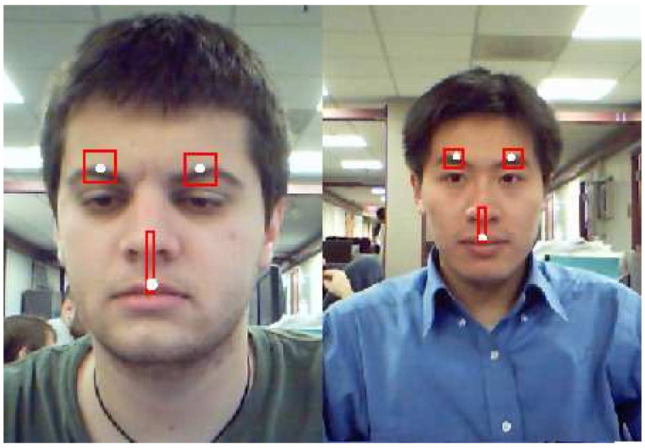

Figure 2: Examples of users during automated initialization. Outlined rectangles depict the search areas for the tracking points (white disks).

degree that they can wink, the system still enables them to simulate the left-click command of a traditional mouse. This is an improvement over current assistive mouse-replacement systems such as Camera Mouse [2, 3], which limits the user to left-click commands by hovering over a certain location for a predetermined amount of time. This is counterintuitive as the lack of action on the part of the user causes a click to occur. It can lead the system to issue a click command that was not intended by the user if the user is not moving the mouse within the threshold of hovering time. Our system provides a more intuitive method for controlling the mouse, as it requires a specific action by the user to simulate a mouse click.

2. METHODS

The algorithm used to detect blinks has a three-stage initialization phase. Stage one involves detecting the eyes by looking for the involuntary or voluntary blinks of the user. In stage two, appropriate tracking points are obtained. Stage three involves obtaining online template images of the eyes. The online templates are then used to detect closed and open eyes and two finite state machines are used to control clicking. A flowchart depicting the main stages of the algorithm is shown in Figure 1.

2.1 Initialization: Detecting Eyes and Learning Appearance of Open and Closed Eyes

The first step of the algorithm is to locate the eyes of the user. This is accomplished by looking for the motion that occurs when the user blinks. First, three consecutive images are obtained. No computations occur while obtaining these images so they are as close together in time as determined by the frame rate of the camera. Two difference images are obtained using the three consecutive frames and then thresholded to produce two binary images. Erosion with a cross-shaped structure element [9] is applied to both binary images to reduce noise and then a union operation [9] merges the images into a final motion image. The algorithm then finds the largest connected component [9] of the motion image and the second largest connected component that is a minimum distance away from the first. The minimum distance requirement is necessary because sometimes the motion of a single eye blinking can create two separate connected components and, by requiring that the two connected components are located sufficiently apart, the algorithm can avoid misinterpreting these two components as two blinking eyes.

The two connected components are now the candidate eye locations and are tested against common properties that hold for all individuals. These include neither component is exceptionally large, the width is greater than the height of each component, the two components have approximately the same vertical position and the sub-images at each location are mirror images of each other. Determining whether the sub-images s and m are mirrored images of each other is accomplished using the normalized correlation coefficient

$$\frac{1}{n}\sum_i\left[\frac{(s_i-\bar{s})(m_i-\bar{m})}{\sigma_s\sigma_m}\right],$$

where n is the number of pixels, s_i and m_i are the brightness values at pixel i in images s and m, $\bar{s}$ and $\bar{m}$ are the average brightness values of images s and m, and σ_s and σ_m are the standard deviations of the brightness values of images s and m. If the candidate eye locations pass all of these tests the algorithm assumes these are the location of the user's eyes.

Next, the algorithm obtains tracking points that will be used throughout the use of the program (Figure 2). Three tracking points are obtained; one is located near the upper lip and is used to control the mouse pointer. The other two tracking points are on each eyebrow of the user and are used to track the eyes indirectly. All the tracking points are obtained by finding the point with the highest summations of Sobel gradient magnitudes [9] in an 11×11 pixel region within a specific search region (Figure 2). For the mouse tracking point, the search region is the area between and below the eyes with a height equal to 0.6 times the distance between the eyes. The fraction 0.6 was chosen as a conservative value since obtaining a tracking point below the face and on the neck or clothing is very undesirable. This results in the tracking point near the upper lip, which is a desirable point for tracking because of the intensity changes. For the eye tracking points, the search region is the area directly above the eye. This results in the tracking point being on the eyebrows. Finally, the offset vectors $\overrightarrow{p}_{\text{left}}$ and $\overrightarrow{p}_{\text{right}}$ between each eye and the tracking point on the corresponding eyebrow are calculated. These vectors will be used later to approximate the locations of the eyes. Tracking is performed with the Lucas-Kanade optical flow algorithm, which has been tested for mouse-replacement systems [6].

The third and final stage of the initialization is obtaining the online templates of the user's open and closed eyes (Figure 3).

We observed that, right after a user blinks, their eyes are open for a few seconds. This gives us a time window to identify a template of each eye while the eye is open. We collect m consecutive frames of the eyes, assuming that the sub-images to be mostly sub-images of open eyes (in practice, $m = 20$ works well). For each left eye sub-image, the correlation coefficient $c_{i,j}$ between sub-image i and j is calculated. The image that is chosen as the open-eye template of the left eye is the sub-image i^*_{left} such that

$$i^*_{\text{left}} = \arg\max_i \sum_j c_{i,j}.$$

Image i^*_{left} is the most representative of the m left-eye images. The open-eye template i^*_{right} of the right eye is chosen similarly by correlating right-eye subimages. If the user blinked once or twice while sub-images were being collected these sub-images would not be chosen as the most representative sub-images, because they correlate poorly with the open-eye images.

After the open-eye templates are learned, the system obtains closed-eye templates by looking for a sub-image that is significantly different from the open-eye sub-image. More specifically, the system again collects m sub-images of the user's eyes. All are compared to the open-eye templates using the normalized correlation coefficient. If the sub-image most unlike the open-eye template has a normalized correlation coefficient value below a threshold it is chosen as the closed-eye template. If no sub-image below the threshold exists, the system would start to collect m new sub-images and repeat the process until such a sub-image was found. In practice, a correlation threshold of 0.8 worked well. After the system found closed-eye templates for both eyes, the initialization phase is complete.

2.2 Control of Mouse Pointer Movement

The tracking point located near the upper lip is used to control the location of the mouse (lowest disk in Figure 2). Movement in the image is mapped to movement of the mouse. Smoothing is applied to the sequence of mouse pointer locations, so that the mouse pointer is easier to control. The distance the pointer moves on the screen that one-pixel movement of the tracking point maps to can be adjusted to the user's preference.

2.3 Blink Detection

The algorithm, on a frame by frame basis, determines if the user's eyes are open or closed. This is accomplished by performing a normalized correlation search with both templates. The center of each search is the approximate location of the respective eye, which is obtained by using the respective eyebrow tracking point and offset vector $\overrightarrow{p}_{\text{left}}$ or $\overrightarrow{p}_{\text{right}}$. To allow the template search area to be sufficiently large to always contain the actual eye location and to allow real-time use, we do not employ a pixel-by-pixel search. Instead an initial search is performed by skipping over every k pixels vertically and horizontally (e.g., $k = 4$). Then a pixel-by-pixel search is performed around the three best-matched locations. This allows the search area to be large enough and does not sacrifice any significant accuracy. The final decision of whether the eye is open or closed is determined by which template has a higher correlation.

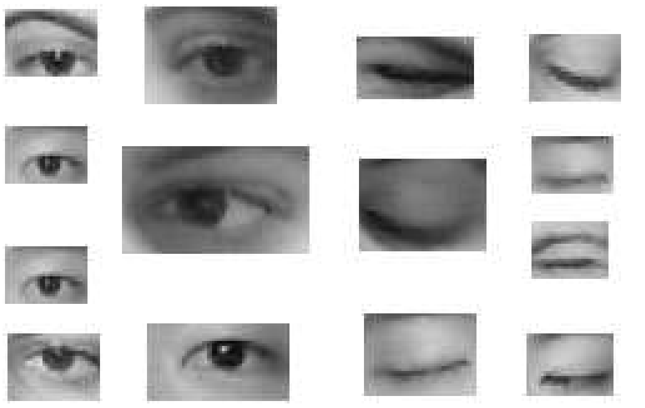
Figure 3: Examples of eye templates automatically collected from various users.

2.4 Interpretation of Mouse Commands with Finite State Machines

The algorithm uses two finite state machines (FSM), one for each eye, to control the clicking of the mouse (Figure 7). The FSMs are used to determine if a blink is voluntary or involuntary, filter out false detections, and determine which type of mouse command the user is attempting. False detections are filtered out by comparing a single detection to the detections around it and taking a majority vote. The algorithm allows the user two modes of control for simulating mouse commands. If a user is able to wink, the algorithm simulates a double-click when a user blinks and a left or right click when the user winks with their left or right eye. The algorithm simulates a left-drag action which occurs when the user closes their left eye and keeps it closed while they move the mouse pointer. When the user opens their left eye, the algorithm sends a left-release signal to the operating system. If a user is only able to blink and cannot wink the same FSMs can be used with a slight modification. Instead of double clicking when the user blinks, the algorithm sends a left-click signal to the operating system. All other transitions and states can still exist but do not perform any mouse commands.

2.5 System Feedback to Users

The system presents the user with feedback so the user knows if the system is working properly. This is especially important right after initialization. The algorithm moves a semi-transparent feedback window (Figure 4) along with the mouse pointer so the user can be informed about whether the system is detecting a closed or open eye. The window

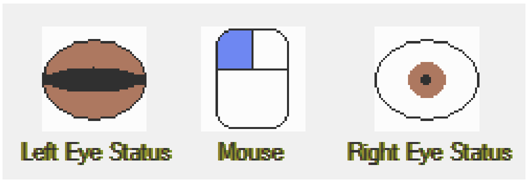

Figure 4: Feedback window used to indicate to user or caregiver the detected status of the eyes, here a closed left eye and an open right eye, which is interpreted as a command to click the left mouse button. The feedback window is positioned above the mouse pointer and follows the mouse pointer throughout the tracking. The window is semi-transparent to allow the user to see the interface below.

also contains a mouse image that changes as the user initiates mouse commands so the user knows when a command has been executed. In addition to the window, the algorithm also performs a beep to signal that it has detected a voluntary blink. This is helpful since the user is not able to see the window when their eyes are closed.

3. EXPERIMENTS AND RESULTS

Two types of tests were performed. The first was to determine the accuracy of the template matching for open-eye versus closed-eye classification. The second was to test the usability of the click interface system. Tests were performed with a 640×480 resolution camera with the test subjects sitting approximately 60 cm away from the camera (Figure 6). The camera was a built-in laptop camera and the computer had a 2.16 GHz Intel Core 2 processor with 2 gigabytes of RAM.

Tests were conducted with a total of 20 subjects (13 males and 7 females). Test subjects had various ethnicities and included individuals wearing glasses (Figure 6).

3.1 Eye Classification Accuracy

To test the accuracy of the template matching system in an environment similar to how the entire system would be used, we asked test subjects to move the mouse pointer to a region on the screen that was highlighted. The test subject opened and closed their eyes with the mouse pointer in the region and then proceeded to the next region. The positions of the regions included all edges and corners of the screen. The images received by the camera were recorded by the system. Each eye was then manually marked as open or closed. Images in which eyes appeared in between being open or closed were not considered. A total of 8306 left and right eye sub-images were tested. An overall accuracy of $8027/8306 = 96.6\%$ was obtained with the template matching classification method (Table 1).

3.2 Mouse Pointer Clicking

The mouse-click functionality of the system was tested by simulating the typical tasks that a user performs with a computer mouse. For left, right and double clicking this involved moving the mouse pointer to a specific location of the screen and performing the desired task. To simulate a left click and drag command, this involved moving the pointer to a desired location, performing a left-down-click command, moving the pointer to another location and performing a left-release command.

Image Type	Number of Images	Successful Classifications	Success Rate
Open Eye Images	4987	4796	96.8%
Closed Eye Images	3319	3231	97.5%
All Images	8306	8027	96.6%

Table 1: Results of classification tests: Left and right eye images, automatically segmented from videos of 8 test subjects, were classified to contain an open or closed eye.

During testing the user was presented with a white screen in which circles appeared one at a time (Figure 5). The user's task was to move the mouse pointer inside each circle and send a mouse command. The color of the circle indicated the mouse command the user was asked to perform.

We added an intermediate task between every mouse command that the user was asked to perform. This task only required the user to move the pointer to a new location. This intermediate task added rigor to our testing protocol. It increased the period of time during which the user was not supposed to perform a mouse command and thus allowed us to measure occurrences of false-positive detections of mouse commands.

The tests that required users to wink had fewer subjects because not all test subjects were able to wink both their left and right eyes while keeping the other eye visibly open. The success rate for sending left-click commands via blinking was $91.8\% = 201/219$. This is significant because left-clicking is the command most commonly used to interact with a computer. The success rates for each mouse command is presented in Table 2.

A successful action required the user to perform the correct type of mouse command in the target circle within a 10-second time limit. We imposed the time limit so if the system was not detecting a blink or wink, our tests would record this. Failed actions included performing the wrong type of mouse command, performing the mouse command while the pointer was outside the target circle, and failing to issue the mouse command within the time limit.

4. DISCUSSION AND CONCLUSION

The system presented here extends the functionality of the camera-based binary-switch systems by Grauman et al. [7] and Chau and Betke [4] and thus provides a more versatile system for users. By interpreting the motion of three facial regions, including both eyes, the system enables a user to control the mouse pointer on a level similar to a traditional mouse. Also, during system development we observed that by using both an open-eye and closed-eye template, instead of one template, the system was more accurate in classifying the eye state during head turns. As a user's head turned and their eyes were open, the correlation coefficient for the open-eye template dropped. However, since a slightly-turned open eye looks more like the open-eye template than the closed-eye template, the system still registered it as an open-eye. The same is true for when the user's eye or eyes were closed and their head turned. A system that uses a single-eye template has difficulty in interpreting the eye state when the

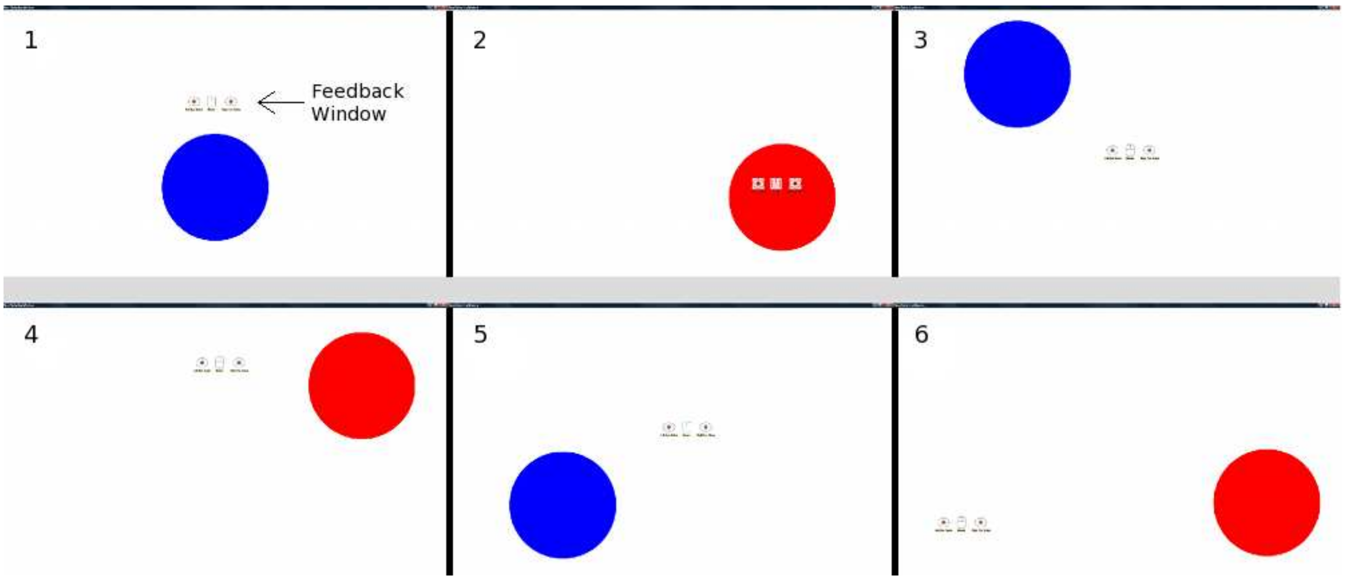

Figure 5: Six screenshots of the testing environment taken while a user was performing mouse commands. At the beginning of the test (frame 1), a blue circle appeared in a white screen and the user was asked to move the mouse pointer into the blue circle. The location of the mouse is underneath the feedback window (Figure 4). When the mouse pointer reached the blue circle, a red circle appeared at another location on the screen, as shown in frame 2. The user moved the mouse pointer to the red circle and sent a click command. The screen changed to contain a blue circle, again in a different location, as shown in frame 3. This process was repeated until a total of 20 red and 20 blue circles had appeared. The user issued a mouse command whenever a red circled was reached and just passed through the blue circles without sending a mouse command. We used the blue circles in our test, because they provided an opportunity for us to evaluate if the system made any false-positive interpretations of the eye state that resulted in unintended mouse commands.

Click Type	Number of Tests	Total Number of Clicks	Successful Clicks	Incorrect Command	Click Outside of Circle	Failed to Click in Time	Success Rate
Double Click	14	277	239	17	19	2	86.3%
Left and Right Click	4	77	63	6	4	4	81.8%
Dragging	3	84	80	2	2	0	95.2%
Blinking Left Click	14	260	239	N/A	19	2	91.9%

Table 2: Results of mouse command tests with 15 subjects. Success rates are computed from 21 tests. Five individuals, who were able to wink, performed multiple tests involving different commands (top three rows). Fourteen subjects used blinking to initiate left-click commands (bottom row).

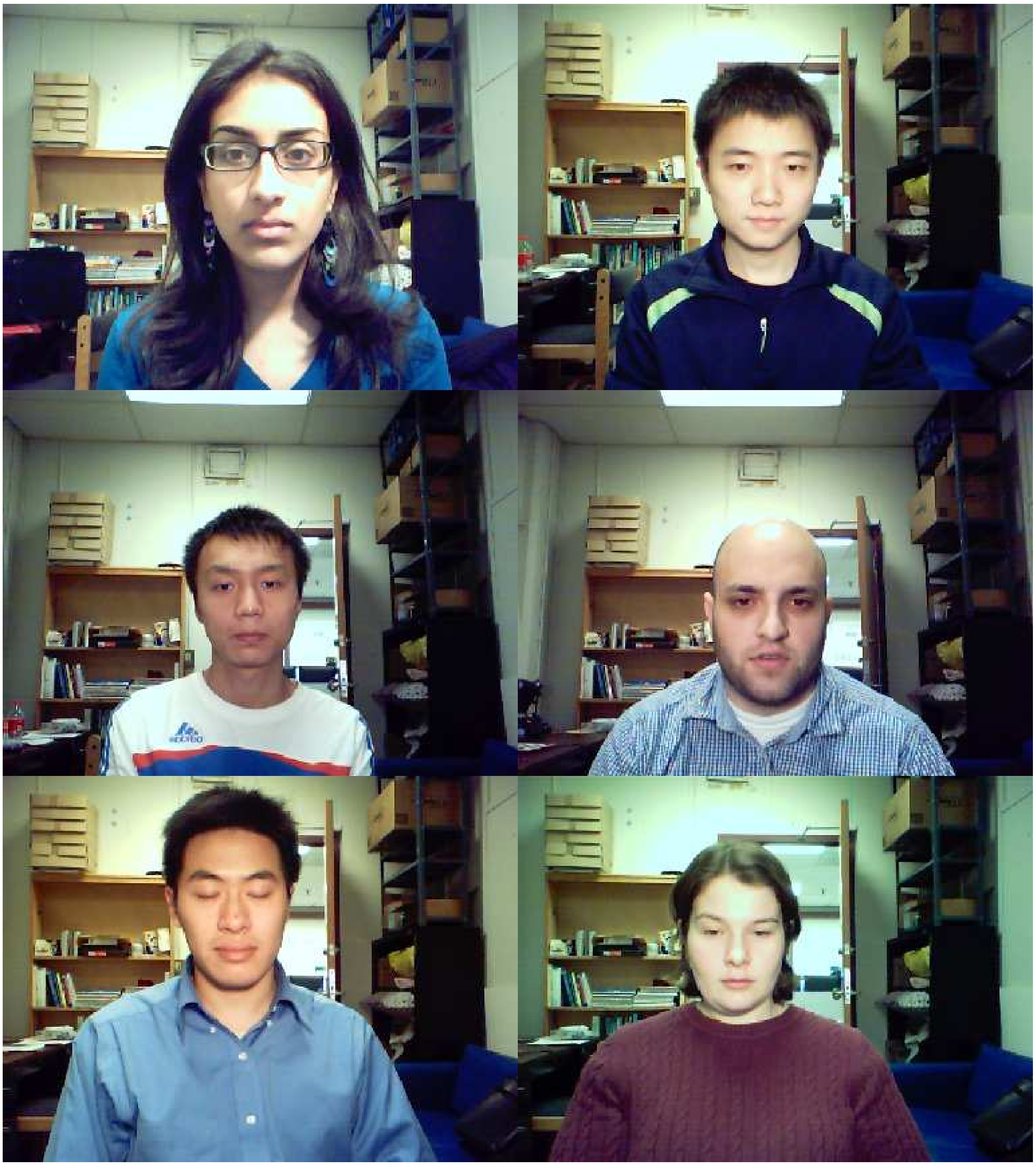

Figure 6: Examples of users interacting with our system to move the mouse pointer with their head and send click commands with their eyes.

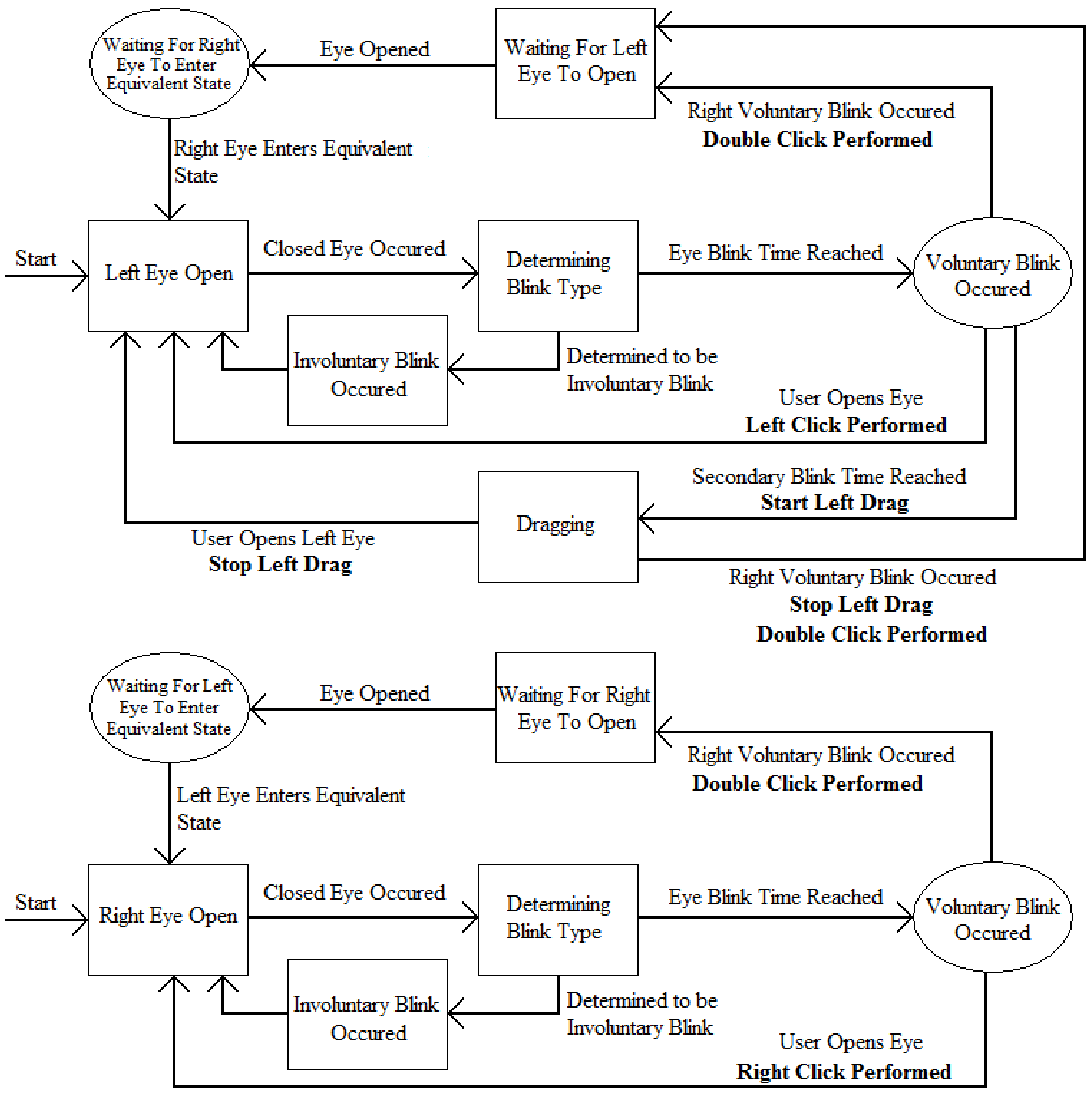
Waiting For Right Eye To Enter Equivalent State
Eye Opened
Waiting For Left Eye To Open
Right Voluntary Blink Occured
Double Click Performed
Right Eye Enters Equivalent State
Start
Left Eye Open
Closed Eye Occured
Determining Blink Type
Eye Blink Time Reached
Voluntary Blink Occured
Involuntary Blink Occured
Determined to be Involuntary Blink
User Opens Eye
Left Click Performed
Secondary Blink Time Reached
Start Left Drag
Dragging
User Opens Left Eye
Stop Left Drag
Right Voluntary Blink Occured
Stop Left Drag
Double Click Performed
Waiting For Left Eye To Enter Equivalent State
Eye Opened
Waiting For Right Eye To Open
Right Voluntary Blink Occured
Double Click Performed
Left Eye Enters Equivalent State
Start
Right Eye Open
Closed Eye Occured
Determining Blink Type
Eye Blink Time Reached
Voluntary Blink Occured
Involuntary Blink Occured
Determined to be Involuntary Blink
User Opens Eye
Right Click Performed

Figure 7: Finite state machines (FSM) used to control mouse commands. The top FSM determines the state of the left eye and the bottom FSM the state of the right eye. Input to the FSMs comes from the eye classification section of the algorithm. Mouse commands are indicated in bold. Ellipses indicate states that require communication with the other FSM.

head is turned significantly. The second template did not result in a significant decrease in speed of our system because we implemented a fast two-state template search as described in Section 2.3.

The system does not require any equipment beyond a standard USB camera. It makes no assumptions about the shape or skin color of the user's face. Some tests were also conducted with users wearing glasses and the system has shown success with individuals with glasses. Lighting was sometimes required to be moved so there was not a glare on the user's glasses, but after this precaution was taken the system was able to detect the location of the users eyes and accurately classify open and closed eyes.

The tests for evaluating the success of the left-and-right-click and dragging commands had fewer subjects because not everybody was able to wink with one eye while keeping the other eye open.

Another issue with wink detection was that winking and blinking eyes may appear different which impacted the initialization phase. Before the system could be fully used, the online templates might have to be reinitialized to detect winking eyes. Also when a user winks their other eye involuntarily begins to close which the system might interpret as a blink. This fact would sometimes require the eye templates to be reinitialized during testing. The user feedback window allowed the user to quickly know when new templates needed to be obtained. When being used by individuals with severe motion impairments, the feedback window would indicate to a caregiver that new eye templates should be acquired. If the user was only performing left-click commands via blinks, the eye templates would rarely need to be reinitialized.

Future work will include efforts to improve the performance of wink detection. An improved system could automatically detect when new templates needed to be obtained so the assistance of a caregiver would not be necessary. We also plan to transition this system from a basic research environment into a web accessible technology. This has shown to be effective for the Camera Mouse system which had started as a research system and is now a popular assistive technology. A large user base providing feedback would be beneficial to the improvement of the system.

Acknowledgments

The authors thank the test subjects for their time and gratefully acknowledge NSF funding (HCC grant IIS-0713229). The first author also acknowledges the NSF Scholar Award to Attend Doctoral Consortium of PETRA 2010.

5. REFERENCES

[1] I. Bacivarov, M. Ionita, and P. Corcoran. Statistical models of appearance for eye tracking and eye-blink detection and measurement. *IEEE Transactions on Consumer Electronics,*, 54(3):1312–1320, August 2008.

[2] 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*, 10:1–10, 2002.

[3] Camera mouse, January 2010. `http://www.cameramouse.org`.

[4] M. Chau and M. Betke. Real time eye tracking and blink detection with USB cameras. *Department of Computer Science Technical Report BUCS-2005-012, Boston University*, April 2005.

[5] M. Divjak and H. Bischof. Eye blink based fatigue detection for prevention of computer vision syndrome. In *Proceedings of the IAPR Conference on Machine Vision Applications (MVA 2009)*, pages 350–353, May 2009.

[6] C. Fagiani, M. Betke, and J. Gips. Evaluation of tracking methods for human-computer interaction. In *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision (WACV '02)*, pages 121–126, December 2002.

[7] K. Grauman, M. Betke, J. Lombardi, J. Gips, and G. R. Bradski. Communication via eye blinks and eyebrow raises: Video-based human-computer interfaces. *Universal Access in the Information Society*, 2(4):359–373, 2003.

[8] A. Haro, M. Flickner, and I. Essa. Detecting and tracking eyes by using their physiological properties dynamics and appearance. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 1*, pages 163–168, 2000.

[9] R. Jain, R. Kasturi, and B. Schunk. *Machine Vision.* McGraw Hill, 1995.

[10] T. Moriyama, T. Kanade, J. F. Cohn, J. Xiao, Z. Ambadar, J. Gao, and H. Imamura. Automatic recognition of eye blinking in spontaneously occurring behavior. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR 2002*, pages 78–81, 2002.

[11] D. Torricelli, M. Goffredo, S. Conforto, and M. Schmid. An adaptive blink detector to initialize and update a view-based remote eye gaze tracking system in a natural scenario. *Pattern Recognition Letters*, 30(12):1144–1150, 2009.

[12] J. Wu and M. M. Trivedi. Simultaneous eye tracking and blink detection with interactive particle filters. *EURASIP Journal on Advances Signal Processing*, 2008:1–17, 2008.

[13] Z. Zhu, K. Fujimura, and Q. Ji. Real-time eye detection and tracking under various light conditions. In *Proceedings of the Eye Tracking Research & Applications Symposium (ETRA)*, pages 139–144. ACM Press, 2002.