A Human–Computer Interface Using Symmetry Between Eyes to Detect Gaze Direction

John J. Magee, Margrit Betke, James Gips, Matthew R. Scott, and Benjamin N. Waber

***Abstract*—In the cases of paralysis so severe that a person's ability to control movement is limited to the muscles around the eyes, eye movements or blinks are the only way for the person to communicate. Interfaces that assist in such communication are often intrusive, require special hardware, or rely on active infrared illumination. A nonintrusive communication interface system called EyeKeys was therefore developed, which runs on a consumer-grade computer with video input from an inexpensive Universal Serial Bus camera and works without special lighting. The system detects and tracks the person's face using multiscale template correlation. The symmetry between left and right eyes is exploited to detect if the person is looking at the camera or to the left or right side. The detected eye direction can then be used to control applications such as spelling programs or games. The game "BlockEscape" was developed to evaluate the performance of EyeKeys and compare it to a mouse substitution interface. Experiments with EyeKeys have shown that it is an easily used computer input and control device for able-bodied people and has the potential to become a practical tool for people with severe paralysis.**

***Index Terms*—Assistive technology, disabled computer users, face detection, face tracking, gaze estimation, video-based human–computer interfaces, webcams.**

I. Introduction

EXTREME paralysis can result from a traumatic brain injury, for example, due to a traffic accident or from cerebral palsy, brain-stem stroke [1], [2], or degenerative neurological diseases, such as multiple sclerosis (MS) or amyotrophic lateral sclerosis (ALS or "Lou Gehrig's disease"). According to the National Multiple Sclerosis Society [3], approximately 400 000 Americans acknowledge having MS, and every week, about 200 people are diagnosed. Worldwide, MS may affect over two million individuals. According to the ALS Association [4], as many as 30 000 Americans are estimated to have the disease at any given time.

For people with severe paralysis, communication abilities are extremely limited—often to *yes* and *no* responses using eye movements or blinks, since the eye muscles are the only muscles they can control. The goal of this paper is to help stop an active mind from being trapped in a body that has difficulty communicating. As progress toward that goal, we created a camera-based computer interface called EyeKeys that makes communication possible by detecting the eye-gaze direction and simulating computer keyboard input. The system is shown in use in Fig. 1.

The most basic interface for people with disabilities is a mechanical switch which allows users with some motor control to operate equipment such as wheelchairs or computer programs. A switch mounted near the user's head, which is activated by leaning the head into it, for example, is shown in Fig. 1. Other systems employ mouth-actuated joysticks, breath-puffing straws, tongue-movement analysis, infrared head-pointing devices, or head-mounted cameras as interfaces (e.g., [5]–[13]). Systems based on electrooculography use electrodes placed on the face to detect the movements of the eyes [14], [15].

Currently, available interfaces cannot be used by some of the most severely paralyzed individuals. They also have the drawback that they must be attached to the user, which is often perceived as intrusive. Electrooculographic sensors, for example, must touch the user's face and can be uncomfortable. Some users, since they cannot move to protect themselves, are very defensive about people placing electrodes on their face or touching their face. The large headgear of some head-mounted eye-gaze-detection systems is not suited for all users, particularly small children. Given these issues, it was an important design requirement for our gaze-detection system to not touch the user. The camera was therefore placed in front of the user at a short distance rather than on a head mount.

Many current camera-based interfaces that analyze eye information make use of active infrared illumination [16]–[27].

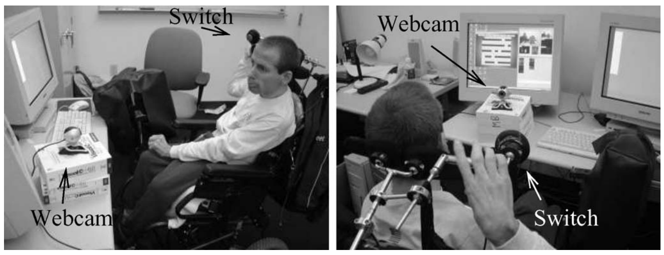

Fig. 1. Two views of a subject with severe cerebral palsy who is using the EyeKeys system with a webcam. The switch attached to his wheelchair is an assistive technology that he uses regularly.

Manuscript received February 21, 2005; revised January 9, 2008 and February 8, 2008. Current version published October 20, 2008. This work was supported in part by The Whitaker Foundation, by the National Science Foundation under Grants IIS-0713229, IIS-0308213, IIS-039009, IIS-0093367, P200A01031, and EIA-0202067, and by the Office of Naval Research under Grant N000140110444. This paper was recommended by Associate Editor R. Hess.

J. J. Magee is with the Computer Science Department, Boston University, Boston, MA 02215 USA.

M. Betke is with the Computer Science Department, Boston University, Boston, MA 02215 USA and also with the Harvard Medical School, Boston, MA 02115 USA (e-mail: betke@cs.bu.edu).

J. Gips is with the Information Systems Department, Boston College, Chestnut Hill, MA 02467 USA.

M. R. Scott is with Microsoft, Redmond, WA 98052-6399 USA.

B. N. Waber is with The Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139-4307 USA.

Digital Object Identifier 10.1109/TSMCA.2008.2003466

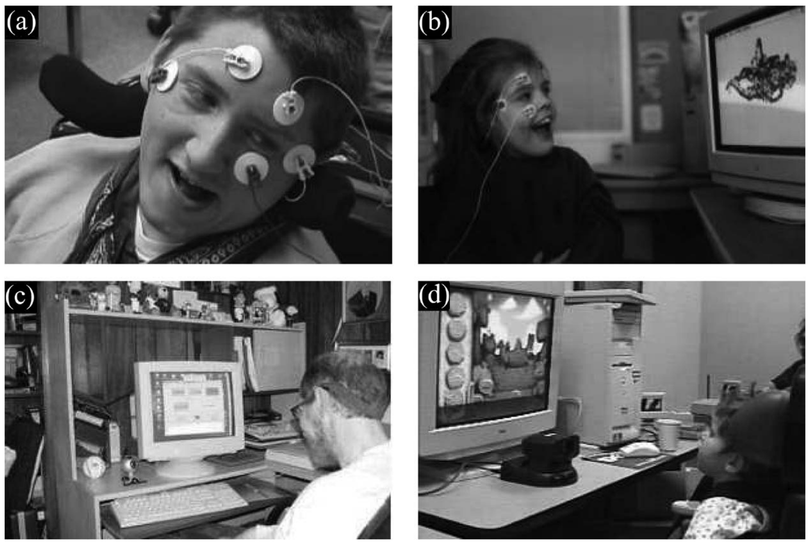

Fig. 2. Communication interfaces for people with severe disabilities: (top) EagleEyes in use with five electrodes. (Bottom) The Camera Mouse in use with (d) a pan/zoom/tilt camera and (c) a webcam. Users with [(a), (b), and (d)] severe cerebral palsy and (c) ALS, (c) write messages, (d) play games, and (b) create art.

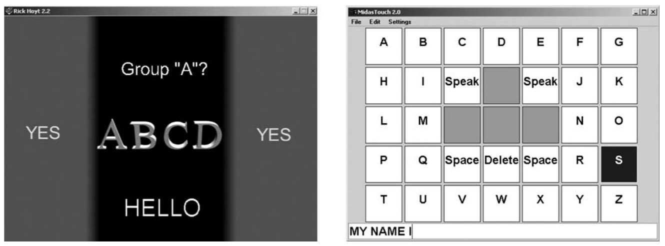

Fig. 3. Spelling interface programs for use with the EagleEyes and Camera Mouse interface systems. (Left) Spelling method based on letter groups [30]. (Right) On-screen keyboard [28].

In many approaches, the infrared light reflects off the back of the eye to create a distinct "bright pupil effect" in the image. The light is synchronized with the camera to illuminate the eyes in alternate frames. The eyes are then located by comparing a frame with this bright pupil effect with a subsequent frame without the illuminated pupils. Typically, the relative gaze direction of an eye is found by analyzing the difference between the center of the bright eye pixel area and the reflection off the surface of the eye from the light source. There is a concern that infrared camera-based interfaces require a complicated calibration procedure which is difficult for small children to follow. Another issue is that they are too expensive for many users, typically costing over $10 000 (e.g., [5]).

Avoiding specialized hardware and infrared light sources are important goals of more recent efforts in designing camera-based computer interfaces for people with severe disabilities (e.g., [28] and [29]). One successful system is the Camera Mouse [28], which is available as a free download at http://www.cameramouse.org. People with disabilities control a mouse pointer by moving their head, nose, chin, finger, or any body feature that they can move, while the system uses video to track the motion (Fig. 2). The Camera Mouse is regularly used by many people to control application programs to enter messages, create art, navigate the Internet, or play games (Fig. 3). It is successful for those who can move their heads or limbs; however, people who can only move their eyes are unable to use it. These are the people for whom we aim to provide a communication device. Our interface system therefore only uses information from the eyes.

One method to locate the eyes is to use a series of intentional blinks [29], [31], [32]. Some people with severe disabilities simply cannot blink voluntarily. The proposed system does not require a blink-based initialization. It instead tracks the face in order to locate the eyes. Face detection and tracking research has yielded a variety of methods [33], [34] that use features, textures, color, or templates, for example, that are tracked in either two [31], [35]–[37] or three dimensions [38], [39].

Skin color analysis is often used as part of a face detection system (e.g., [31], [35], and [37]). Various methods and color spaces can be used to segment pixels that are likely skin from pixels that may belong to the background. This is a difficult problem because skin tones are widely varied over the population. The system presented here uses color and motion information as a preprocessing mask. Similar approaches have been used previously to segment the background and foreground (e.g., [40]).

Existing methods for eye detection and tracking, as well as gaze analysis, that do not rely on infrared lighting typically depend on high-resolution images of eyes, often obtained by pan/tilt/zoom cameras or cameras that are head mounted [41]–[49]. The system proposed here was designed around an inexpensive visible-light "webcam," and it works with images that have a lower resolution than the images used in previous approaches (e.g., [41] and [43]–[54]). These approaches require high-resolution images of the eyes for appearance- or feature-based methods that locate, for example, eye corners, eyelids, or contours of the iris. An exception is the work by Kawato and Tetsutani [55], who used low-resolution video to track the position of the eyes by tracking a template region between the eyes, which is updated online. Since EyeKeys works with lower resolution images of the eyes, the user may move further from the camera and still allow the system to function. The use of webcams also allows the system to be made available to people at significantly less expense than those requiring specialized hardware.

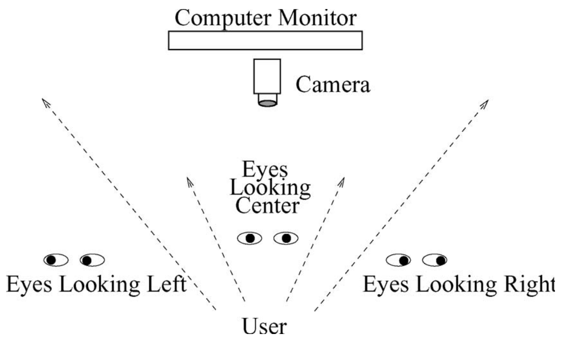

Fig. 4. Configuration layout of the EyeKeys system. The user faces both the camera and the computer monitor. The computer uses the camera to detect if the user's eyes move to the right or to the left. The dashed lines indicate possible view directions of the person's eyes.

Another interesting eye-gaze estimation method is presented by Darrel *et al.* [56]. In this method, the correlation response of three eye templates is measured and interpolated to infer gaze direction. This method may need the templates to be trained on the user for best performance and also relies on images of the eye that are of the same scale as the template.

The EyeKeys system combines existing computer vision techniques in a new way to quickly track the face using multi-scale template correlation with the aid of skin color and motion analyses. The left and right eyes are compared to determine if the user is looking at the center or to the left or right side. The new approach exploits the symmetry of the face when comparing the eyes. The output of the system can be used to control applications such as spelling programs or games.

Earlier work and preliminary tests of the system have been reported previously [57]–[59]. In the traditional human–computer interface configuration, the computer user sits in front of a keyboard, mouse, and monitor. Interfaces based on computer vision techniques are designed to mimic that setup, with a camera substituting for the input devices. In the EyeKeys system, instead of hands controlling the keyboard and mouse, the movement of the eyes effectively controls the computer through the use of the camera directed at the user's face (Fig. 4). In the case of a severely disabled user, a dual display is useful in that an assistant can monitor the computer vision interface on one display while the user's application is displayed on the other screen. A triple monitor configuration could also be used so that the user moves his or her eyes between the three displays. The system can also be configured to use a large projected display instead of a monitor.

II. Methods

The system consists of two main modules: 1) the face detector and tracker, and 2) the eye analysis module (Fig. 5). The 2-D face tracker locates the scale and position of the face and a region containing the eyes. The eye analysis module then refines the estimate of the location of the eyes and determines if the eyes are looking toward the center, to the left, or to the right of the camera. The output from the eye module can be the input to a simple computer control interface by simulating a key press. The user's eye movements can be interpreted, for example, as a selection command. While the command is issued, the user is supposed to keep his or her head relatively stable. The system assumes that the user is facing the camera and the computer monitor and has the ability to move his or her eye toward the left and right. The user's head should be upright and not rotated away from the camera. Although the system can be calibrated for intersession lighting variations, the system assumes constant intrasession illumination. Illumination is also assumed to be primarily frontal or uniform on the face.

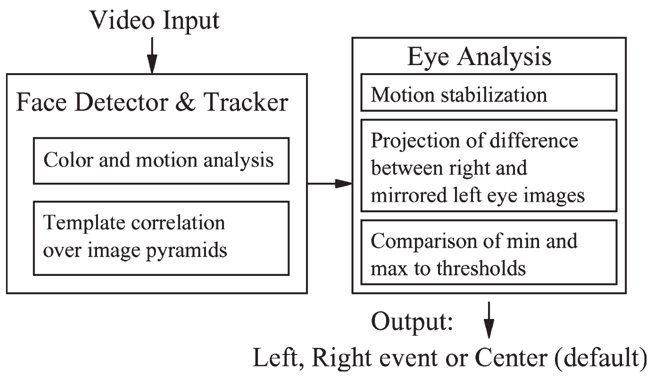
Video Input
Face Detector & Tracker
Color and motion analysis
Template correlation over image pyramids
Eye Analysis
Motion stabilization
Projection of difference between right and mirrored left eye images
Comparison of min and max to thresholds
Output:
Left, Right event or Center (default)

Fig. 5. EyeKeys' system flowchart.

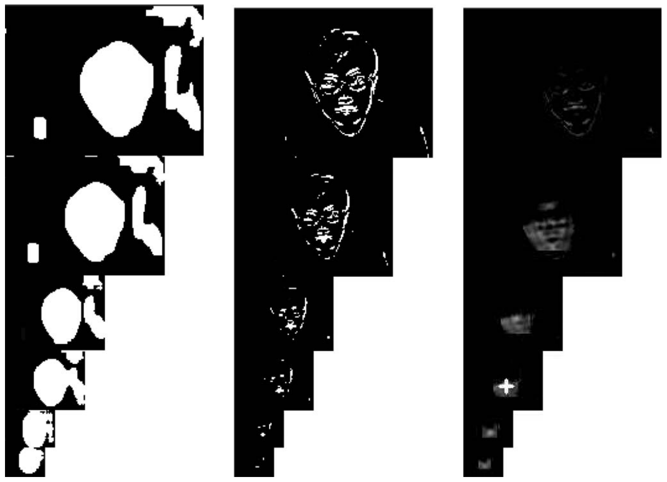

Fig. 6. Pyramids P_{Color} and P_{Motion} before the application of a low-pass filter and $P_{\text{Correlation}}$ computed by the face detection and tracking algorithm. Regions with potential skin color are shown in white in P_{Color}. Regions with strong changes in brightness, which are typically due to motion, are shown in white in P_{Motion}. High correlation values between face subimage and template face are visualized by bright gray levels in $P_{\text{Correlation}}$. The face was detected in the (white cross) third smallest resolution image.

A. Face Detection and Tracking

The system uses color, motion, and correlation-based template matching to detect and track faces. It can detect and track different size faces at various distances to the camera. This allows users to move closer or farther from the camera and still be detected automatically. To achieve this, the system uses image pyramids [60], as shown in Fig. 6. Each pyramid consists of eight levels. The highest resolution image in the pyramid is the 640 $\times$ 480 video frame, and the lowest resolution image contains 32 $\times$ 24 pixels. In each level of the pyramid, the system searches for a face of size 12 $\times$ 16. This approach allows the system to operate in real time. The level of the pyramid at which the face is detected can then be used to infer the "true" size and location of the face, which is the size and location

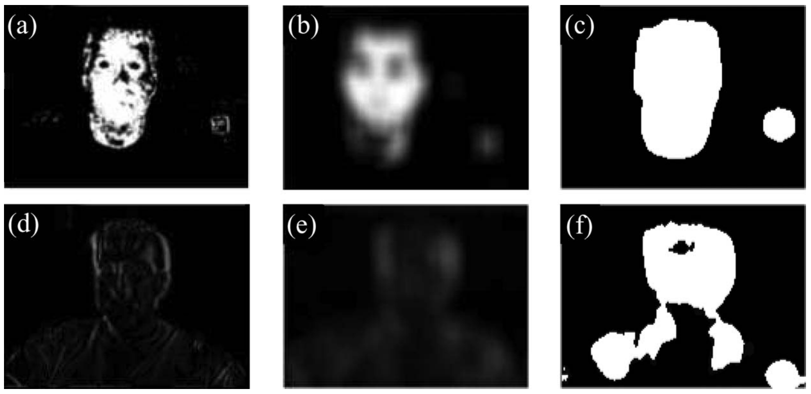

Fig. 7. (Top row) Intermediate images produced during color analysis. (a) Regions with potential skin color are shown in white. (b) Low-pass filtering of image (a). (c) Binary color mask computed by thresholding image (b). The facial features (nose and eyes) in (a) are marked as potential face pixels in (c); some background pixels in (a) are removed in (c). (Bottom row) Intermediate images produced during motion analysis. (d) Regions with motion, computed from frame differencing, are shown in white. (e) Low-pass filtering of image (d). (f) Binary motion mask computed by thresholding image (e).

of the face in the original video frame. In the case where the person is far from the camera and appears relatively small in the video frame, the system can efficiently detect the face in a high-resolution level of the pyramid. In the case where the person is close to the camera and therefore appears relatively large in the original video frame, the face is detected efficiently in a lower resolution level of the pyramid.

The face tracking algorithm computes pyramids P_{Color} and P_{Motion} from the result of a color histogram lookup C and a motion image M, as described in Sections I and II hereafter. These pyramids are used to mask the pyramid P_{Input} of the input video frame to yield a pyramid $P_{\text{MaskedInput}}$, which is then used to compute a pyramid $P_{\text{Correlation}}$ of template correlation values, as described in Section III hereafter.

```
**Face Detection and Tracking Algorithm**
**input**: Color video, 640 × 480 pixels, state at $t-1$
**output**: Location and scale of face: ($x$, $y$, *scale*).
**for** *Each video frame* $I_t$ **do**
  /* **Color image analysis** */
    $C_t \leftarrow$ histogram-lookup($I_t$)
    $P_{\text{Color}} \leftarrow$ pyramid-decimate($C_t$)
    $P_{\text{Color}} \leftarrow$ average-filter($P_{\text{Color}}, 12 \times 16$)
    $P_{\text{Color}} \leftarrow$ threshold($P_{\text{Color}}, 10$)
  /* **Motion analysis** */
    $M_t \leftarrow |I_t - I_{t-1}|$
    $P_{\text{Motion}} \leftarrow$ pyramid-decimate($M_t$)
    $P_{\text{Motion}} \leftarrow$ average-filter($P_{\text{Motion}}, 12 \times 16$)
    $P_{\text{Motion}} \leftarrow$ threshold($P_{\text{Motion}}, 10$)
    $P_{\text{Motion}} \leftarrow$ add-prior-location ($P_{\text{Motion}}$, $x$, $y$, *scale*,
  state at $t-1$)
  /* **Correlation-based template matching** */
    $P_{\text{Input}} \leftarrow$ pyramid-decimate($I_t$)
    $P_{\text{MaskedInput}} \leftarrow P_{\text{Input}} \& P_{\text{Color}} \& P_{\text{Motion}}$
    $P_{\text{Correlation}} \leftarrow$ correlate ($P_{\text{MaskedInput}}$, *face template*)
  /* **Find location and scale of the face** */
    ($x$, $y$, scale) $\leftarrow$ arg-max-value($P_{\text{Correlation}}$)
**end**
```

1) Skin Color Analysis: The color input image, which contains three 8-bit values for each pixel in the red/green/blue color space, is converted into the luminance/chrominance (YUV) color space [61]. A binary map of likely skin pixels in the image [Fig. 7(a)] is created by comparing the UV values of the image to a histogram of UV values for skin pixels in a set of training images. The color histogram was trained on images of 15 different faces. For each face, the pixels within a rectangle covering most of the facial region were considered skin pixels. The histogram can be retrained during system operation by clicking on areas of skin in the live video. The histogram can be saved and loaded so that it can be used again for the same user or lighting conditions without retraining. This is useful if the default histogram is not appropriate for a specific user or environment.

The binary map of likely skin pixels [Fig. 7(a)] is smoothed with a 5 × 5 approximation to a Gaussian filter [60], [62] [Fig. 7(b)] and then decimated into the other pyramid levels by removing rows and columns. In each pyramid level, a low-pass filter that averages the grayscale values over a neighborhood of 12 × 16 is applied, which is the size of the face that the system aims to detect in that level. Thresholding the filter output by ten gray levels then yields a binary image of potential face regions [Fig. 7(c)]. The filter and threshold serve to remove small areas in the background, which are incorrectly detected as skin, and fill in holes in the binary mask [Fig. 7(a)], which were not detected as skin but could be part of the face.

2) Motion Detection: As in the skin color analysis, a preprocessing mask—here, the motion mask [Fig. 7(f)]—is created to aid face detection. The analysis is based on the assumption that, if the user moves, frame differencing should find the pixels where motion occurs in the image. The frame differencing creates the motion image [Fig. 7(d)]

$$M(x,y) = |I_t(x,y) - I_{t-1}(x,y)| \tag{1}$$

where the pixel values $M(x,y)$ are the absolute difference between the grayscale values of the current image frame I_t and the previous image frame I_{t-1} (grayscale values are provided by the Y channel of the YUV color image). Since the frame rate of the camera used is 15 frames per second, those frames represent images taken approximately 67 ms apart. A higher frame rate may require the algorithm to be modified to maintain a similar temporal separation between frames.

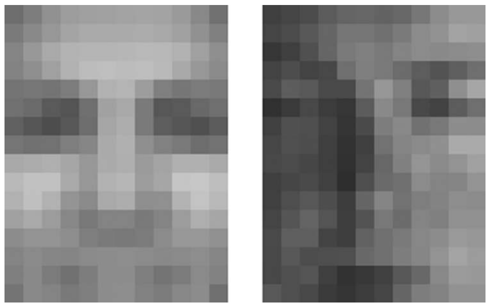
Fig. 8. (Left) Average and (right) user-specific face templates used by the correlation algorithm to find the scale and location of the face. In this user-specific face template, the user was more brightly lit from one side.

The motion image is decimated into a pyramid. As in the color analysis, the goal is to find a face of size 12 × 16. A low-pass averaging filter of support 12 × 16 is therefore applied to remove motion that cannot be due to the motion of the face [Fig. 7(e)]. Subsequent thresholding with a threshold value of ten gray levels then results in the binary motion mask [Fig. 7(f)]. Fig. 6(b) shows a motion mask pyramid with the low-pass filter disabled to better highlight pixels that contribute to the motion image.

In the case when there is little or no motion, the system tries to find the face near the same location and scale as found in a previous frame. It sets the locations within five pixels of the previously found face location to one in the binary motion image and thus prevents the motion mask from excluding the previous face location from subsequent processing. The two adjacent motion pyramid levels are also modified in this way to account for small movements toward or away from the camera, which are not detected by the motion analysis. When the system initializes and there is no previously detected face location, the center of the image is set as the default prior location for all pyramid levels.

3) Detection and Tracking Via Correlation: Template matching based on the normalized correlation coefficient [63] is used to estimate the location of the face. A 12 × 16 face template [Fig. 8(a)] was created by averaging the brightness values of eight face images aligned by a paint program. The grayscale images in the input pyramid are convolved with the binary images in the color and motion pyramids, which yields a pyramid $P_{\mathrm{MaskedInput}}$ of masked grayscale images [Fig. 6(c)]. The face template is then correlated with the images over all levels of $P_{\mathrm{MaskedInput}}$ in order to search for the position and scale of the face in the scene that matches best with the template face. The maximum correlation peak among all of the levels indicates the location of the face. The scale of the face is inferred by the level of the pyramid at which the maximum is found. Masking the grayscale input image with information from the color and motion analyses serves two purposes. First, it eliminates possible ambiguous correlation peaks in the background, and second, the search space for the correlation function is reduced.

An average face template has been used previously in face detection (e.g., [64]). EyeKeys can detect faces that were not used in creating the default template, because the default template is a relatively small and coarse representation of an

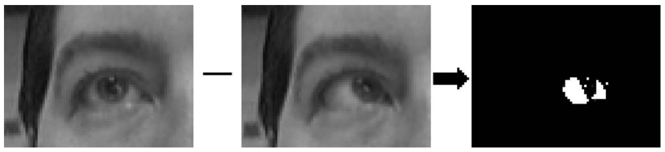
Fig. 9. Motion detected by frame differencing is thresholded and used as a mask for the left–right image differencing.

"average human face." The operator also has the option to initialize the template with the face of the current user. This can be done by clicking on the user's nose in the image on the screen and then selecting the correct face scale. Note also that, to allow detection and tracking of the interface user's face, there should be only one face in the field of view of the camera.

4) Computational Complexity of Face Tracker: The correlation module of the face tracker is the most computationally expensive function of the system. The face tracker employs multiscale techniques in order to provide real-time performance. The template correlation over the image pyramid is more efficient than performing multiple correlations with a scaled template. Color and motion information is used to improve accuracy and reduce the search space of the template matching method, which further improves efficiency.

In the face detection, all of the following processes have a computation time that is linear in the number of pixels in the image: histogram lookup, pyramid decimation, thresholding, frame differencing, and image masking. The averaging filter and normalized correlation have a fixed filter size; thus, they remain linear in the number of pixels.

B. Eye Analysis

Given the estimate of face location provided by the face tracker, the approximate location and scale of the eyes can be inferred from simple anthropomorphic properties: The eyes must be located in a region above the center of the face, the left eye must be on the right side of this image region, and the right eye is on the left. Taking advantage of these properties, the eye analysis module crops out two subimages containing the eyes from the highest resolution image. The size of the subimages depends on the scale at which the face was found. To simplify the eye analysis, the system produces eye images of a fixed size of 60 × 80 pixels by linear interpolation.

1) Motion Analysis and Stabilization and Eye-Corner Detection: Ideally, the system computes cropped eye images in which the position of the eyes does not change even as the head moves. However, a slight movement of the head may not be accurately tracked by the face tracker, which would result in an apparent movement of the position of the eyes within the cropped eye images. Our method requires the eyes to be located in the same position in each cropped eye image. Motion in the cropped eye images also occurs during blinks and eye movements. The system tries to distinguish these different movements and only use the eye movements as a communication tool. It therefore "stabilizes" the eye images so that further analysis can provide information about gaze direction.

The stabilization process involves frame differencing (1), which creates a binary eye-motion image (Fig. 9) with a

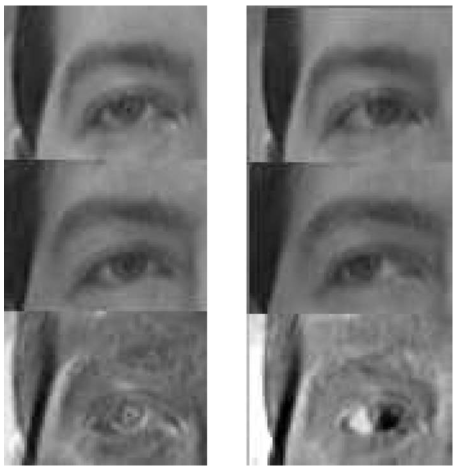

Fig. 10. Two gaze examples: (Left column) A gaze straight ahead and (right column) a gaze to the user's left. (Top and middle) Eye images automatically extracted from input video by face tracker and aligned by motion analysis. (Top) Right eye. (Middle) Mirrored left eye. (Bottom) Difference between top and middle images; zero values shown in gray. Right difference image shows a region of low brightness difference values in black and high brightness difference values in white.

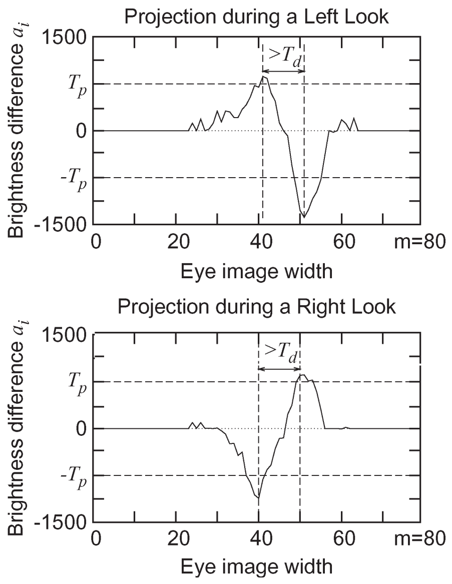

Fig. 11. Results of projecting the signed difference between right and mirrored left eyes onto the x-axis. (Top) Result of left-looking eyes. (Bottom) Result of right-looking eyes.

threshold value of 30 units of grayscale differences. The first-order moments or "centers of mass" [62] of these images are then used to adjust the estimates of the eye locations in the face image.

In some cases, the first-order moments of the binary eye-motion images do not yield proper alignment of the eye images. A secondary alignment technique based on detecting eye corners was thus introduced. This method uses an 8 × 8 image template of an eye corner for each eye. The peak of the normalized correlation coefficient is found in each corresponding eye image. The located eye corners are then used to adjust the cropped eye images so that the left and right eye images will properly align for comparison (see below). A moving history of the eye-corner locations is used to determine if the motion-based estimate described earlier is used or if the current best correlation match is used to adjust the eye locations.

The two eye regions are recentered at the newly estimated eye locations. The eyes then appear in the *same* relative locations in the eye images.

2) Left–Right Eye Comparisons: The following original method is the most important contribution of this paper. The system compares the stabilized left and right eye images to determine where the user is looking. To accomplish this, it mirrors the left eye image I_ℓ (Fig. 10, middle) and subtracts it from the right eye image I_r (Fig. 10, top). If the user is looking straight at the camera, the brightness difference is very small (Fig. 10, bottom left), and the system concludes that the user is looking straight. If the eyes are looking left, the left eye appears to look right in the mirrored left eye image, and the brightness difference of the eye images is large (Fig. 10, bottom right) and the system can conclude that the user is looking to the side. To determine the direction, the signed differences are projected onto the x-axis to yield a vector $\mathbf{a} = a_1, \ldots, a_m$, where

$$a_i = \sum_{j=1}^{n} \left(I_r(i,j) - I_\ell(m-i,j)\right) \tag{2}$$

where m is the eye image width, and n is the height. If the user is looking left, this signed difference operation creates large values in the projection because the dark-gray iris and pupil pixels in I_ℓ are subtracted from the light-gray eye sclera pixels in I_r, followed by small values in the projection because light-gray eye sclera pixels in I_ℓ are subtracted from dark-gray iris and pupil pixels in I_r. Vice versa, if the user is looking right, there will be a valley in the projection, followed by a peak (Fig. 11). If the user is looking straight, the grayscale values of the two eye images are similar, thus creating low values in the projection.

If the illumination of the face is not primarily uniform or from the front, the difference of illumination of the eyes may push the projections up or down. To compensate for this, if side illumination is present, the projections are adjusted up or down by 50% of the difference of the average brightness between the left and right eye images.

Two thresholds T_p and T_d are used to evaluate whether an eye motion occurred to the right, left, or not at all. First, the maximum and minimum components

$$a_{\min} = \min_i(a_i) \qquad a_{\max} = \max_i(a_i) \tag{3}$$

of the projection vector $\mathbf{a}$, where $i = 1, \ldots, m$, are compared to the projection threshold T_p:

$$a_{\min} < -T_p \qquad a_{\max} > T_p. \tag{4}$$

A threshold value $T_p = 750$ assures that there is a sufficient brightness difference between the mirrored left eye and the right eye to indicate a motion.

The second threshold T_d is used to guarantee a minimal spatial difference between the minimum and maximum projection values when motion is detected. This helps prevent motion that is not an intentional eye movement from being detected. The implementation uses a value of $T_d = 8$ pixel units. Given the indices

$$i_{\min} = \arg\min_i(a_i) \qquad i_{\max} = \arg\max_i(a_i) \tag{5}$$

of the minimum and maximum projection values, the direction of motion is determined as follows:

$$i_{\max} - i_{\min} > T_d \quad \Rightarrow \text{ `right motion'} \tag{6}$$

$$i_{\max} - i_{\min} < -T_d \quad \Rightarrow \text{ `left motion.'} \tag{7}$$

The thresholds can be adjusted to change the specificity and sensitivity of the system in detecting eye motion and gaze direction. The computational complexity of the eye analysis, in particular, resampling, first-order moments, and difference projections, is linear in the size of the input image. The search for the minimum and maximum projection values is linear in the width m of the cropped eye image.

```
Gaze Analysis Algorithm
input: Video 640 × 480, face location and scale ($x$, $y$, scale)
output: Gaze classification: left, center (default), right
for Each video frame $I_t$ do
  /* Compute eye region of interests (ROIs)*/
    $\mathrm{ROI}_{\mathrm{left}} \leftarrow$ compute-ROI($x$, $y$, scale, left)
    $\mathrm{ROI}_{\mathrm{right}} \leftarrow$ compute-ROI($x$, $y$, scale, right)
  /* Crop and rescale eye images */
    $I_{\ell,t} \leftarrow$ crop-resize-ROI($I_t, \mathrm{ROI}_{\mathrm{left}}$, scale)
    $I_{r,t} \leftarrow$ crop-resize-ROI($I_t, \mathrm{ROI}_{\mathrm{right}}$, scale)
  */ Motion stabilization */
    $M_{\ell,t} \leftarrow |I_{\ell,t} - I_{\ell,t-1}|$
    $M_{\ell,t} \leftarrow$ threshold($M_{\ell,t}$, 30)
    $M_{r,t} \leftarrow |I_{r,t} - I_{r,t-1}|$
    $M_{r,t} \leftarrow$ threshold($M_{r,t}$, 30)
    $\mathrm{Moment}_\ell \leftarrow$ first-order-moment($M_{\ell,t}$)
    $\mathrm{Moment}_r \leftarrow$ first-order-moment($M_{r,t}$)
    $I_\ell \leftarrow$ recenter-eye($I_{\ell,t}, \mathrm{Moment}_\ell, \mathrm{Corner\text{-}Template}_\ell$)
    $I_r \leftarrow$ recenter-eye ($I_{r,t}, \mathrm{Moment}_r, \mathrm{Corner\text{-}Template}_r$)
  /* Left-right eye comparison */
    /* $M_\ell$ and $M_r$ are recentered motion masks */
    $I_\ell \leftarrow I_\ell \& M_\ell$,    $I_r \leftarrow I_r \& M_r$
    $\mathbf{a} \leftarrow$ compute-difference-projection($I_\ell, I_r$)
    $\mathbf{a} \leftarrow$ illumination-compensation($\mathbf{a}, I_\ell, I_r$)
    $a_{\min} \leftarrow \min(\mathbf{a})$,    $a_{\max} \leftarrow \max(\mathbf{a})$
    $i_{\min} \leftarrow \arg\min(\mathbf{a})$,    $i_{\max} \leftarrow \arg\max(\mathbf{a})$
    if $a_{\min} < -T_p$ and $a_{\max} > T_p$ then
      if  $i_{\max} - i_{\min} > T_d$   output: right
      if  $i_{\max} - i_{\min} < -T_d$   output: left
  Default Output: center
end
```

C. Controlling the Computer With EyeKeys

The detected direction of gaze motion is used to control the computer by sending a message to the operating system to simulate an event, such as a keyboard press. A limit was set on how frequently events can be triggered in order to prevent too many events in quick succession from being triggered. Experimental analysis found that setting this limit to once every 0.5 s works well. In the default version of EyeKeys, when "right motion" is detected (6), a right arrow key press is simulated. When "left motion" is detected (7), a left arrow key press is simulated. The simulated keys can be changed, depending on the application the user wishes to control. In addition to key presses, EyeKeys can simulate left and right mouse movements.

TABLE I
INTERFACE MAPPINGS FOR HCI APPLICATIONS

Application	Left Look	Right Look
Scan-based text entry	Select letter	
Group-based text entry	Select group or letter	
Selection text entry	highlight next letter	select letter
Web browser	Go to next link	Follow current link
BlockEscape game	Move block left	Move block right
Driving game	Drive left	Drive right

Examples of human–computer interaction (HCI) programs that can be used with EyeKeys are scan-based text entry programs, augmented web browsers, and some games. Table I gives possible mappings for some applications that work under the constraint that only two types of inputs are possible.

Scan-based text entry programs use an on-screen keyboard or other groups of letters placed on the screen (Fig. 3) and automatically scan through the choice of letters or groups of letters [65], [66]. When the intended group or letter is highlighted, the user can select it by moving the eyes either left or right. Looking center means "select nothing." Alternatively, looking at one direction can move the highlighted selection to the next letter, while looking the other direction can select the currently highlighted letter. Using these methods, the computer user can construct words or entire sentences.

EyeKeys can be used as an interface to navigate the World Wide Web if the detected direction of gaze motion is used to control the tab and enter keys. Looking left maps to the tab key to move to the next link, and looking right maps to the enter key to follow a link. Web sites yahoo.com or myway.com have starting pages useful for navigating the Web with EyeKeys due to their hierarchical menus. Finding a page, for example, with the weather in Boston requires following only four links: Weather → United States → Massachusetts → Boston. For a satisfying browsing experience, currently available web browsers would have to be augmented to allow the use of the "back" button. The browser by Larson and Gips [67], for example, asks each time a new page appears whether the user wants to proceed or go back.

D. BlockEscape Game

The game BlockEscape was developed as a tool to test the performance of EyeKeys as an interface. The game can also be configured with other camera-based interfaces, such as the Camera Mouse [28], to test their performance. The goal was to create a game that provides test subjects with an engaging experience and, at the same time, gathers statistics about the interface performance. This was a challenging task because people with severe disabilities can typically only play for short

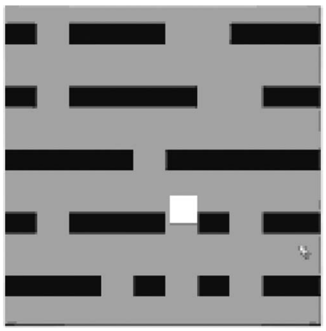
Fig. 12. Screenshot of the game BlockEscape. As the white block falls toward the bottom of the screen, the player navigates it through the holes in the black walls, which move upward, by initiating "move block left" or "move block right" commands.

periods of time, and during this time, performance data must be collected.

In the BlockEscape game (Fig. 12), the screen contains a number of horizontal walls that are separated vertically by a fixed distance. Each wall has one or more holes of various widths. The number of and distance between walls and the number and width of holes are parameters that can be set before playing. At the beginning of the game, a block appears on top of the screen. The goal of the game is for the user to lead the block through the holes in the walls until it reaches the bottom of the screen. The game was implemented so that the user only needs to initiate the block's horizontal movement by issuing "move block left" or "move block right" commands. The block continues to move in the initiated direction until the user issues a command for movement in the opposite direction. The block's vertical motion is automatic—when a hole is reached, the block "falls through" to the next wall or at bottom of the screen. The speed of the block can be set before playing.

The game can be configured so that, at the beginning of playing, the screen is empty, and during playing, the walls appear at the bottom of the screen and move upward at a constant rate. When a wall reaches the user's block, it pushes the block upward. The user wins if he or she leads the block through the holes in the walls to the bottom of the screen and loses if a wall pushes the block up to the top of the screen. The higher the wall speed is set, the more difficult the game becomes. The wall sequence of a particular play can be reconstructed, allowing the user to play the same game multiple times. This also allows different users to play the same game, which was useful for the analysis of EyeKeys. During playing, usage statistics, in particular, the departure of the user-controlled block from an optimal path, were computed based on the positions of the block, walls, and holes and compiled into Extensible Markup Language documents. In the case for games with single-hole walls, the distance $d_{i,t} = |x_{i,t} - h_i|$ from the block's position $x_{i,t}$ on wall i at time t to hole h_i was computed and compared with previous distance $d_{i,t-1}$. A binary deviation score $b_{i,t}$ was defined to measure whether the block moved closer or farther away from the hole:

$$b_{i,t} = \begin{cases} 0, & \text{if } d_{i,t} < d_{i,t-1} \text{ or } t = 1 \\ 1, & \text{otherwise} \end{cases} \tag{8}$$

for $t = 1, \ldots, T_i$, where T_i is the number of the game-update cycles during which the block is on wall i. Note that this deviation measure can be easily extended to include cases with multiple holes in a wall. The departure D from the optimal path was accumulated for the n walls of the game:

$$D = \sum_{i=1}^{n} \sum_{t=1}^{T_i} b_{i,t}. \tag{9}$$

The optimal path could be easily determined by a player. Thus, when the keyboard was used to control the block movement, D was zero. When a video-based interface was used, the nonzero values of D were assumed to be due to false detections of the interface rather than misjudgments of the player. Here, the term "false detection" refers to two problems—false classification, for example, the user wanted to issue a move-left command by looking left and the interface falsely detected a right look. It also refers to false positive detection, i.e., the user looked around without the intent to issue a command, but the system interpreted the look as a command.

III. Experiments and Results

A. *EyeKeys Detection and Classification Experiment*

Experiments were conducted to determine if the system can detect when a user intentionally looks to the left or to the right.

1) Methodology: EyeKeys was tested with the camera mounted on the end of an articulated arm, which allowed the camera to be optimally positioned in front of the computer monitor (Fig. 4). In this experiment, the Universal Serial Bus (USB) camera used was a Logitech Quickcam Pro 4000 with a retail price of $79.99 in 2004. (We also observed the system in use with other cameras of the same and different models.) The tests were run on an Athlon 2100 computer and conducted at the Image and Video Computing laboratory at Boston University.

The default face template used by the tracker was first updated to include the test subject (to isolate the performance of the eye analysis method). Test subjects were told to look at the computer monitor. When asked to look left, the tester was instructed to move his or her eyes to look at a target point to the left of the monitor. A similar target was to the right side of the monitor. After the look was completed, the user was asked to look back at the monitor.

A random sequence of 20 looks was created: ten to the left and ten to the right. The same sequence was used for all the test subjects. The system's classification of each look was recorded. If the system did not recognize a look, the user was asked to repeat it. The number of tries required to make a recognition was recorded. The first experiment involved tests without the eye-corner stabilization method, while the second experiment included the eye-corner stabilization method.

2) Results: In the first experiment, the EyeKeys system was tested by eight able-bodied people. The faces of all test subjects were correctly tracked in both location and scale while moving between 50 and 150 cm from the camera. The system correctly classified 140 (87.5%) of the 160 detected looks to the left or right. For the system to detect 160 looks, the users had to make 248 attempts. On average, 1.77 (248/140) actual looks

TABLE II
NUMBER OF ACTUAL AND DETECTED LEFT AND RIGHT LOOKS IN TESTING THE EYEKEYS SYSTEM

	Actual Left	Actual Right	Detection Rate
Detected Left	72	12	90.0%
Detected Right	8	68	85.0%
Missed	40	48	
Overall Rate	60%	53%	

TABLE III
NUMBER OF ACTUAL AND DETECTED LEFT AND RIGHT LOOKS IN TESTING THE EYEKEYS SYSTEM, WITH EYE-CORNER STABILIZATION METHOD

	Actual Left	Actual Right	Detection Rate
Detected Left	38	3	95.0%
Detected Right	2	37	92.5%
Missed	6	8	
Overall Rate	82.6%	77.1%	

are made for each correctly identified look event, which would correspond to an overall detection rate of 56.4%. The high number of look attempts and low overall detection rate were due to high thresholds that were chosen to avoid false detection of looks, since it was deemed better to miss a look than to misclassify a look. The results are summarized in Table II. For some of the test subjects, EyeKeys was more successful in detecting and classifying looks. For example, it correctly identified all 20 looks of one subject who only made 24 actual looks (a true positive detection rate of 100% and a false negative detection rate of 83%).

Cases where an incorrect recognition (false negative or positive detections, or false classifications) occurred were mostly due to a problem with aligning the right and mirrored-left eyes using the method based on the first-order moment of eye difference image described earlier. Possible solutions to this problem are discussed in Section IV. Other incorrect recognitions were due to the system missing a look in one direction but detecting eye movement back to the center position as a move in the opposite direction (i.e., a false negative detection, followed immediately by a false positive detection).

In the second experiment, EyeKeys was tested with four subjects using the same methodology but with the eye-corner stabilization intended to overcome the problem aligning the right and mirrored-left eyes, as described earlier. These subjects were a subset of the eight able-bodied subjects from the previous experiment. Each subject was observed until 20 detections had been achieved. The overall detection rate increased from 56.4% in the first experiment to $75/94 = 79.7\%$ in this experiment (Table III).

B. Side Illumination Experiment

An experiment was conducted on the performance improvement of the EyeKeys classification system under a side illumination condition.

1) Methodology: This experiment followed similar methodology as the classification experiment described in the previous section. A table lamp was set up to illuminate the subject's face from the side, while ambient room lighting remained on. A shade was used on the lamp, and it was angled away from the face to diffuse the lighting slightly (otherwise, the lighting would saturate the face image on one side). A sequence of looks was observed first without the illumination adjustment method and then repeated again with the illumination adjustment method. Three subjects, which were a subset of the subjects in the other experiments, were observed for this experiment. The subjects were familiar with the system at this point; therefore, learning during this experiment was unlikely to be a factor in the results.

2) Results: The results of this experiment show an improvement of performance of the system under side illumination conditions with the illumination adjustment method enabled. The overall detection rate increased from $45/101 = 44.6\%$ without side illumination adjustment to $53/73 = 72.6\%$ with the illumination adaptation (Table IV).

TABLE IV
NUMBER OF ACTUAL AND DETECTED LEFT AND RIGHT LOOKS IN TESTING THE EYEKEYS SYSTEM, WITH AND WITHOUT SIDE ILLUMINATION ADJUSTMENT METHOD

Without Illumination Adjustment			
	Actual Left	Actual Right	Correct Rate
Detected Left	23	8	76.6%
Detected Right	7	22	73.3%
Missed	22	19	
Overall Rate	44.2%	44.9%	
With Illumination Adjustment			
Detected Left	26	3	86.6%
Detected Right	4	27	90.0%
Missed	8	5	
Overall Rate	68.4%	77.1%	

C. BlockEscape Experiment

An experiment was conducted to evaluate the use of EyeKeys as an interface to a possible real-world application—here, the BlockEscape game.

1) Methodology: Four able-bodied subjects were read the rules of BlockEscape, followed by two demonstrations of the game using a mouse. The Camera Mouse was chosen in this experiment so that the effectiveness of EyeKeys could be compared to a previously developed HCI system for people with disabilities. The keyboard was selected as a control against the video-based HCI systems. All subjects were unfamiliar with BlockEscape, EyeKeys, and the Camera Mouse.

In the "practice phase," the subjects were allowed to become familiar with the game and the interfaces. They played up to three trial games, or for up to 3 min, on the keyboard, Camera Mouse, and EyeKeys. They were then asked to play at least one game for 30 s with each device. For the "trial phase," the test subjects played three games on each input device.

2) Results: Users had different levels of success with EyeKeys. One user mastered EyeKeys quickly, winning all three games, but had trouble with the Camera Mouse: losing one game and performing poorly on another. The other users improved their performance on succeeding games with EyeKeys. Table V summarizes the results. The number of errors made by the camera-based interfaces was small—on average, less than three misclassifications per game—and the users were thus able to win the game at a high rate.

TABLE V
NUMBER OF DEPARTURES D FROM THE OPTIMAL PATH AND WINS FOR FOUR USERS EMPLOYING THREE DEVICES TO PLAY BLOCKESCAPE

D	EyeKeys	Camera Mouse	Keyboard
Mean	2.90	2.27	0
Median	2.54	0	0
Std. Dev.	4.01	2.68	0
Wins	10/12 (83%)	10/12 (83%)	12/12 (100%)

D. System Limitation Experiments

Experiments were conducted to determine the limitations of the system when the computer user's head was tilted forward or sideways, turned sideways, or located at different distances from camera and when the angle between the user's center and side looks was changed.

1) Methodology: During the head orientation experiments, the subjects sat in front of the computer using the system while slowly turning or tilting their heads. The video of these experiments, recorded by a second camera, was then analyzed to determine at which head orientations the EyeKeys system performed reliably. Distance limits were similarly obtained by moving the subjects' heads to predetermined distances from the camera and observing EyeKeys' performance. The minimum eye angle was determined by placing targets for a user to move their eyes to the side and adjusting them closer or further from center.

2) Results: The head orientation experiments determined that the face tracker generally fails during a left or right head turn of approximately 39° and a tilt of about 20°. The head tracker does not work well when the user's head is tilted sideways. The face was tracked reliably for a camera-to-head distance of 70 to 160 cm. A look had to be at least 30° to the left or right sides for a reliable eye-motion detection.

E. Real-Time Performance of System

To be a useful human–computer interface, a video-based face and eye analysis system must run in real time and must not use all of the processing power of the computer, since the same computer has to run both the camera-based interface as well as the user program, such as a text entry program or game. The EyeKeys system worked within these constraints. It achieved real-time performance at 15 frames per second, which is the limit of the USB camera at 640 × 480 resolution, while application programs, such as the BlockEscape game, were running concurrently. The windows task manager reports 58%–62% CPU usage while only EyeKeys is running.

IV. DISCUSSION

A. Testing Experience and Comparisons

The test subjects had little difficulty learning the EyeKeys interface. After only a minute of practice, able-bodied users could use it to play the BlockEscape game.

The performance of EyeKeys was similar to the performance of the Camera Mouse. However, when the tracking method failed in the Camera Mouse, the interface performance decreased dramatically, and manual intervention to reset the tracker was needed. With EyeKeys, a false detection could be easily rectified in applications, such as BlockEscape, that are forgiving of incorrect inputs—the user only needed to look in the appropriate direction to correct a mistake.

Some users preferred EyeKeys over the Camera Mouse because EyeKeys did not require them to move their heads in order to interact with the computer. Others did not like to move their eyes to interact with the computer and preferred the Camera Mouse. Users of both interfaces had to face the "Midas Touch Problem" [68]—in order to avoid issuing not-intended commands, they had to restrict their natural movements. Camera Mouse users could not turn their heads, EyeKeys users could not look around to observe their environment. EyeKeys' gaze analysis algorithm, however, was able to correctly interpret some eye movements that users made to observe the computer screen, as long as these movements were not too rapid to inadvertently trigger computer commands.

It initially seems awkward for a person to have to look away from the screen to make a selection. When working with the system, however, we experienced that the required rapid movement of the eye feels much like a natural saccadic eye movement. It is over so quickly that it does not seem to distract the user's focus of attention on the screen. This is demonstrated with a video: http://www.cs.bu.edu/fac/betke/videos/EyeControlled-BlockEscape-Game.avi. If looking away from the screen causes a problem for some users, a possible solution would be to use either a multimonitor configuration or a large projected display.

B. Initial Experience: Users With Severe Disabilities

The preliminary testing of EyeKeys involving six subjects with severe paralysis has been promising. The first subject was a nonverbal person with severe cerebral palsy who could control his eyes and head movements. He also had some involuntary head and limb movements. His primary method of communication was small head nods to spell out words and sentences with an assistant. Subject 1 was also able to use a computer by pressing a switch with his head or using the Camera Mouse system. These observations were conducted at the HCI laboratory at Boston University.

It was first tested if the EyeKeys system could detect and track the subject's head and eyes. One problem was immediately apparent: The subject had difficulty holding his head straight up. Since the face detector used a default template of an upright face, it was not immediately able to locate the head correctly. Manually reinitializing the face template solved this initial problem. Since the eye analysis algorithm assumed symmetry across a vertical face as well, repositioning the subject's body in the chair was needed so that the subject could more easily hold his head vertically.

It was then tested if the EyeKeys system could detect when subject 1 intentionally looked to the left or to the right. The system was able to correctly identify most of the times when the subject looked to one side without moving his head. Many of the attempts he made were not correctly identified because he also moved his head to the side to try to look at the target.

Subject 1 was asked to use the EyeKeys system to move a window left and right across the screen. In most cases, it was observed that he was able to move the window in the direction that he was asked. Sometimes, involuntary head motion would cause the system to detect an unintentional eye event.

It was more difficult for subject 1 to play the BlockEscape game when using EyeKeys than when using the Camera Mouse. The subject had used the Camera Mouse to control a computer in the past and was therefore accustomed to controlling the computer with his head. At the beginning of a game, he could not help moving his head in an attempt to play while, for the most part, keeping his eyes fixed on the computer screen. Head movements had been his only method of communicating, so it might have been difficult for him to try to use only eye movements while keeping his head still.

Five additional subjects were tested with the system to qualitatively evaluate the performance of the system classification. These subjects were observed at The Boston Home in Dorchester, MA. Subject 2, who is a 50-year-old woman with MS, was able to move her eyes to both the left and right. Initially, she was not able to hold her head in a vertical position, which caused both the head tracking and eye analysis to fail. After adjusting her position and the camera position, the system was able to detect many of her eye movements correctly. Sometimes, the system would miss a classification because the eyes did not move quickly to the side. More than once, the system was confused when she moved her eyes first to one direction and then to the other direction.

Subject 3 was a 61-year-old woman with MS. The system was able to detect her eye motion to the left most of the time. However, she was unable to move her eyes to the right far enough to cause a reliable detection. Because of this, her use of the system was limited to single inputs with left looks.

Subjects 4–6 all had the ability to move their heads and communicate verbally. Subjects 4 and 5 were able to use the EyeKeys system to play the BlockEscape game for a short period of time. Subject 6, however, had difficulty keeping one eyelid open to the extent needed by EyeKeys to do gaze analysis. Subject 4 observed that it was somewhat difficult to play the game because EyeKeys required him to move his eyes off the computer screen (where the game is being displayed). Subject 5 observed that he would often try to move his head in the direction he wanted to control the block instead of his eyes.

C. Discussion of Design Choices

The two thresholds of the gaze analysis algorithm are adjustable. Increasing threshold T_p on the projection values yields a system that is more likely to miss an intentional look but less likely to misclassify a look. Increasing threshold T_d on the distance between projection peak and valley has the effect of requiring that the looks be faster and more deliberate. While this can decrease false positive detections, and thus addresses the Midas Touch problem, it also makes the system difficult and uncomfortable to use.

The classical image pyramid consists of images that are half the resolution of the previous pyramid level. The use of a classical pyramid would require the face to be found at discrete size levels that are far apart, i.e., the face appearance in the image would need to halve in size in order to be found in a higher pyramid level. The pyramid used in this system consists of intermediate levels, so that the face can be found at more size levels. The template correlation method could locate a face as it moved close or far from the cameras at these intervals, whereas it would have had difficulty with the levels used in a classical image pyramid.

The default face template allowed most people to use the system without manual initialization. There were some cases, however, when the user's face image differed significantly from the average face and the default template needed to be updated. The ability to update the face template was particularly useful when the user's face and the faces that produced the default face template were illuminated differently. While a tracker based on the normalized correlation coefficient can work with uniform intensity changes [63], problems occur if the user becomes more brightly lit from one side. Updating the template solves this. It is also convenient that the template can be saved in one session and reloaded in another, so that it does not have to be retrained for the same user or lighting conditions.

It is important to note that the eye analysis method is independent of the face detection and tracking method. The face detection and tracking method could be completely replaced by another suitable method that would be able to output the location of the center of the face and an approximate scale. The eye stabilization method can then adjust the location of the cropped eye images based on the provided location and scale of the face.

The system was designed around hardware that would be easily accessible to computer users, such as USB webcams, versus much more expensive head-mounted optics systems (e.g., [5]). While these optic systems focus on gaze tracking, our system focuses on detecting rapid left and right eye movements to serve as a command input for HCI. Higher resolution images may be needed to allow detections of horizontal gaze angles smaller than the extreme left and right. Our method does not easily extend to the analysis of vertical eye movements: The left and right eyes are not symmetrical if mirrored vertically, and eyelids may cover part of the eye when a user looks down.

Other possibilities for extending the EyeKeys system to serve people with severe disabilities include the analysis of the duration that the user looks left or right and the addition of a blink analysis module [29], which would give the interface additional events that can be mapped to commands. The number of available commands could also be increased by trying to detect more than three gaze directions as described earlier. Higher resolution eye images would allow a feature detector to find the corners of the eyes, the irises, and the pupils. The analysis of the difference projection could be done by fitting a function to the projection curve. The parameters of the function could then be used as a measure for the degree that the eyes are looking away from center.

D. Additional Applications of a Gaze-Detection System

Perceptual User Interfaces: Perceptual user interfaces have been studied to find new ways for people to control electronic devices in their homes (e.g., [69] and [70]). EyeKeys has the potential to become an integral part of such an interface that, for example, can turn on the television by detecting eye gaze and turn it off when the user falls asleep.

Linguistic Communications Research: EyeKeys may be useful in linguistic and communication research, for example, to study facial feature movements during sign language conversations. EyeKeys could aid in the annotation of video sequences of such conversation, which are currently analyzed manually with the SignStream system [71], as part of the American Sign Language Linguistic Research Project at Boston University.

Store Attention Monitoring: A store might want to gather information on what products and displays are interesting or eye catching to potential customers. A computer vision system that detects a customer's gaze direction would be able to gather statistics that could be used in marketing research.

Vehicle Driver Monitoring: There have been significant efforts in industry and academia to explore methods for automatic monitoring of drivers for accident avoidance systems (e.g., [72]–[74]). This involves the analysis of eye blinks and eyelid positions to determine if the driver is falling asleep. The face detection and gaze analysis techniques developed for EyeKeys may be useful here.

E. Summary and Conclusion

Assistive technology enables people with severe paralysis to communicate their thoughts and emotions. It also allows them to exhibit their intellectual potential—sometimes disproving a previous diagnosis of a mental disability. To provide such communication technology, we have created the camera-based human–computer interface EyeKeys, which is a new tool to use gaze direction to control the computer. The EyeKeys' face tracker combines existing techniques in a new way that allows the face to be tracked quickly as a means to locate the eyes. The method of mirroring and projecting the difference between the eyes is a novel approach to detecting to which side the eyes look. Experiments with EyeKeys have shown that it is an easily used computer input and control device for able-bodied people and has the potential to become a practical tool for people with severe paralysis.

Acknowledgment

The authors would like to thank the subjects for their enthusiasm and patience in testing EyeKeys, G. Kollios and S. Sclaroff for helpful discussions, and P. Carey and D. Young-Hong for assistance.

References

[1] J.-D. Bauby, *The Diving Bell and the Butterfly*. New York: Vintage Books, 1997.

[2] *The Diving Bell and the Butterfly*, 2007, France: Pathé Renn Productions. Film, Directed by Julian Schnabel.

[3] National Multiple Sclerosis Society. [Online]. Available:http://www.nationalmssociety.org/

[4] ALS Association. [Online]. Available: http://www.alsa.org/

[5] Bedford, MA: Appl. Sci. Lab. [Online]. Available: http://www.a-s-l.com

[6] Y. L. Chen, F. T. Tang, W. H. Chang, M. K. Wong, Y. Y. Shih, and T. S. Kuo, "The new design of an infrared-controlled human–computer interface for the disabled," *IEEE Trans. Rehabil. Eng.*, vol. 7, no. 4, pp. 474–481, Dec. 1999.

[7] Don Johnston, Inc., *Infrared head-mounted mouse alternative.* Penny & Giles HeadWay. [Online]. Avialable: http://www.donjohnston.com

[8] D. G. Evans, R. Drew, and P. Blenkhorn, "Controlling mouse pointer position using an infrared head-operated joystick," *IEEE Trans. Rehabil. Eng.*, vol. 8, no. 1, pp. 107–117, Mar. 2000.

[9] LC Technologies, *Eyegaze system.* [Online]. Available: http://www.lctinc.com

[10] Madentec. [Online]. Available: http://www.madentec.com

[11] Tash solutions. [Online]. Available: http://www.tashinc.com

[12] R. Vaidyanathan, B. Chung, L. Gupta, H. Kook, S. Kota, and J. D. West, "Tongue-movement communication and control concept for hands-free human–machine interfaces," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 4, pp. 533–546, Jul. 2007.

[13] L. Young and D. Sheena, "Survey of eye movement recording methods," *Behav. Res. Meth. Instrum.*, vol. 7, no. 5, pp. 397–429, 1975.

[14] R. Barea, L. Boquete, M. Mazo, and E. López, "System for assisted mobility using eye movements based on electrooculography," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 10, no. 4, pp. 209–218, Dec. 2002.

[15] P. DiMattia, F. X. Curran, and J. Gips, *An Eye Control Teaching Device for Students Without Language Expressive Capacity: EagleEyes.* Lewiston, NY: Edwin Mellen Press, 2001. [Online]. Available: http://www.bc.edu/eagleeyes

[16] Y. Ebisawa, "Improved video-based eye-gaze detection method," *IEEE Trans. Instrum. Meas.*, vol. 47, no. 4, pp. 948–955, Aug. 1998.

[17] A. Gee and R. Cipolla, "Determining the gaze of faces in images," *Image Vis. Comput.*, vol. 12, no. 18, pp. 639–647, Dec. 1994.

[18] T. E. Hutchinson, K. P. White, Jr., W. N. Martin, K. C. Reichert, and L. A. Frey, "Human–computer interaction using eye-gaze input," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 6, pp. 1527–1533, Nov./Dec. 1989.

[19] Q. Ji and Z. Zhu, "Eye and gaze tracking for interactive graphic display," *Mach. Vis. Appl.*, vol. 15, no. 3, pp. 139–148, Jul. 2004.

[20] A. Kapoor and R. W. Picard, "Real-time, fully automatic upper facial feature tracking," in *Proc. 5th IEEE Int. Conf. Autom. Face Gesture Recog.*, Washington, DC, May 2002, pp. 10–15.

[21] C. H. Morimoto, D. Koons, A. Amir, and M. Flickner, "Pupil detection and tracking using multiple light sources," *Image Vis. Comput.*, vol. 18, no. 4, pp. 331–335, Mar. 2000.

[22] G. A. Myers, K. R. Sherman, and L. Stark, "Eye monitor: Microcomputer-based instrument uses an internal mode to track the eye," *Computer*, vol. 24, no. 3, pp. 14–21, Mar. 1991.

[23] B. Noureddin, P. D. Lawrence, and C. F. Man, "A non-contact device for tracking gaze in a human computer interface," *Comput. Vis. Image Underst.*, vol. 98, no. 1, pp. 52–82, Apr. 2005.

[24] K. R. Park, "A real-time gaze position estimation method based on a 3-D eye model," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 199–212, Feb. 2007.

[25] S.-W. Shih and J. Liu, "A novel approach to 3-D gaze tracking using stereo cameras," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 234–245, Feb. 2004.

[26] D. H. Yoo, J. H. Kim, B. R. Lee, and M. J. Chung, "Non-contact eye gaze tracking system by mapping of corneal reflections," in *Proc. 5th IEEE Int. Conf. Autom. Face Gesture Recog.*, Washington, DC, May 2002, pp. 94–99.

[27] D. H. Yoo and M. J. Chung, "Non-intrusive eye gaze estimation without knowledge of eye pose," in *Proc. 6th IEEE Int. Conf. Autom. Face Gesture Recog.*, Seoul, Korea, May 2004, pp. 785–790.

[28] 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 Trans. Neural Syst. Rehabil. Eng.*, vol. 10, no. 1, pp. 1–10, Mar. 2002.

[29] K. Grauman, M. Betke, J. Lombardi, J. Gips, and G. R. Bradski, "Communication via eye blinks and eyebrow raises: Video-based human–computer interfaces," *Univers. Access Inf. Soc.*, vol. 2, no. 4, pp. 359–373, Nov. 2003.

[30] J. Gips and J. Gips, "A computer program based on Rick Hoyt's spelling method for people with profound special needs," in *Proc. ICCHP*, Karlsruhe, Germany, Jul. 2000, pp. 245–250.

[31] K. Schwerdt and J. L. Crowley, "Robust face tracking using color," in *Proc. 4th IEEE Int. Conf. Autom. Face Gesture Recog.*, Grenoble, France, Mar. 2000, pp. 90–95.

[32] O. Takami, K. Morimoto, T. Ochiai, and T. Ishimatsu, "Computer interface to use head and eyeball movement for handicapped people," in *Proc. IEEE Int. Conf. Syst., Man Cybern. Intell. Syst. 21st Century*, 1995, vol. 2, pp. 1119–1123.

[33] E. Hjelmas and B. K. Low, "Face detection: A survey," *Comput. Vis. Image Underst.*, vol. 83, no. 3, pp. 236–274, Sep. 2001.

[34] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 34–58, Jan. 2002.
[35] R.-L. Hsu, M. Abdel-Mottaleb, and A. K. Jain, "Face detection in color images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 696–706, May 2002.
[36] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 23–38, Jan. 1998.
[37] H. Wu, Q. Chen, and M. Yachida, "Face detection from color images using a fuzzy pattern matching method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 6, pp. 557–563, Jun. 1999.
[38] M. LaCascia, S. Sclaroff, and V. Athitsos, "Fast, reliable head tracking under varying illumination: An approach based on robust registration of texture-mapped 3D models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 4, pp. 322–336, Apr. 2000.
[39] R. Stiefelhagen, J. Yang, and A. Waibel, "A model-based gaze tracking system," *Int. J. Artif. Intell. Tools*, vol. 6, no. 2, pp. 193–209, 1997.
[40] R. Féraud, O. J. Bernier, J.-E. Viallet, and M. Collobert, "A fast and accurate face detector based on neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 1, pp. 42–53, Jan. 2001.
[41] M. Betke and J. Kawai, "Gaze detection via self-organizing gray-scale units," in *Proc. Int. Workshop Recog., Anal., Tracking Faces Gestures Real-Time Syst.*, Kerkyra, Greece, Sep. 1999, pp. 70–76.
[42] M. Betke, W. J. Mullally, and J. Magee, "Active detection of eye scleras in real time," in *Proc. IEEE Workshop Human Model., Anal. Synth.,* Hilton Head Island, SC, Jun. 2000. [Online]. Available: http://www.cs.bu.edu/faculty/betke/papers/betke-mullally-magee.pdf
[43] D. Beymer and M. Flickner, "Eye gaze tracking using an active stereo head," in *Proc. CVPR*, Madison, WI, Jun. 2003, vol. 2, pp. 451–458.
[44] J. P. Ivins and J. Porril, "A deformable model of the human iris for measuring small three-dimensional eye movements," *Mach. Vis. Appl.*, vol. 11, no. 1, pp. 42–51, Jun. 1998.
[45] K.-N. Kim and R. S. Ramakrishna, "Vision-based eye-gaze tracking for human computer interface," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Tokyo, Japan, Oct. 1999, vol. 2, pp. 324–329.
[46] S. Sirohey, A. Rosenfeld, and Z. Duric, "A method of detecting and tracking irises and eyelids in video," *Pattern Recognit.*, vol. 35, no. 5, pp. 1389–1401, Jun. 2002.
[47] Y. Tian, T. Kanade, and J. Cohn, "Dual-state parametric eye tracking," in *Proc. 4th IEEE Int. Conf. Autom. Face Gesture Recog.*, Grenoble, France, Mar. 2000, pp. 110–115.
[48] J.-G. Wang and E. Sung, "Study on eye gaze estimation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 3, pp. 332–350, Jun. 2002.
[49] J. Zhu and J. Yang, "Subpixel eye gaze tracking," in *Proc. 5th IEEE Int. Conf. Autom. Face Gesture Recog.*, Washington, DC, May 2002, pp. 124–129.
[50] D. W. Hansen and A. E. C. Pece, "Eye tracking in the wild," *Comput. Vis. Image Underst.*, vol. 98, no. 1, pp. 155–181, Apr. 2005.
[51] T. Kawaguchi and M. Rizon, "Iris detection using intensity and edge information," *Pattern Recognit.*, vol. 36, no. 2, pp. 549–562, Feb. 2003.
[52] H. Wu, Q. Chen, and T. Wada, "Visual line estimation from a single image of two eyes," in *Proc. 17th ICPR*, Cambridge, U.K., Aug. 2004, vol. 3, pp. 290–293.
[53] X. Xie, R. Sudhakar, and H. Zhuang, "Real-time eye feature tracking from a video image sequence using Kalman filter," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 12, pp. 1568–1577, Dec. 1995.
[54] Z.-H. Zhou and X. Geng, "Projection functions for eye detection," *Pattern Recognit.*, vol. 37, no. 5, pp. 1049–1056, May 2004.
[55] S. Kawato and N. Tetsutani, "Detection and tracking of eyes for gaze-camera control," *Image Vis. Comput.*, vol. 22, no. 12, pp. 1031–1038, Oct. 2004.
[56] T. J. Darrell, I. A. Essa, and A. P. Pentland, "Task-specific gesture analysis in real-time using interpolated views," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 12, pp. 1236–1242, Dec. 1996.
[57] J. J. Magee, "A real-time human–computer interface based on gaze detection from a low-grade video camera," M.S. thesis, Dept. Comput. Sci., Boston Univ., Boston, MA, May 2004.
[58] J. J. Magee, M. Betke, M. R. Scott, and B. N. Waber, "A real-time vision interface based on gaze detection—EyeKeys," in *Real-Time Vision for Human–Computer Interaction*, B. Kisacanin, V. Pavlovic, and T. Huang, Eds. Berlin, Germany: Springer Verlag, Jun. 2005.
[59] 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 *Proc. IEEE Workshop RTV4HCI*, Washington, DC, Jul. 2004, p. 8.
[60] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA Eng.*, vol. 29, no. 6, pp. 33–41, 1984.
[61] J.-C. Terrillon and S. Akamatsu, "Comparative performance of different chrominance spaces for color segmentation and detection of human faces in complex scene images," in *Proc. 4th IEEE Int. Conf. Autom. Face Gesture Recog.*, Grenoble, France, Mar. 2000, pp. 54–61.
[62] B. K. P. Horn, *Robot Vision*. Cambridge, MA: MIT Press, 1986.
[63] M. Betke and N. C. Makris, "Recognition, resolution and complexity of objects subject to affine transformation," *Int. J. Comput. Vis.*, vol. 44, no. 1, pp. 5–40, Aug. 2001.
[64] C. Liu, "A Bayesian discriminating features method for face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 6, pp. 725–740, Jun. 2003.
[65] L. A. Frey, K. P. White, Jr., and T. E. Hutchinson, "Eye-gaze word processing," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 4, pp. 944–950, Jul./Aug. 1990.
[66] R. C. Simpson and H. H. Koester, "Adaptive one-switch row-column scanning," *IEEE Trans. Rehabil. Eng.*, vol. 7, no. 4, pp. 464–473, Dec. 1999.
[67] H. Larson and J. Gips, "A web browser for people with quadriplegia," in *Universal Access in HCI: Inclusive Design in the Information Society*, C. Stephanidi, Ed. Mahwah, NJ: Lawrence Erlbaum Assoc., 2003, pp. 226–230.
[68] R. J. K. Jacob, "The use of eye movements in human–computer interaction techniques: What you look at is what you get," *ACM Trans. Inf. Syst.*, vol. 9, no. 3, pp. 152–169, Apr. 1991.
[69] R. Sharma, V. I. Pavlovic, and T. S. Huang, "Toward multimodal human–computer interfaces," *Proc. IEEE*, vol. 86, no. 5, pp. 853–869, May 1998.
[70] M. Turk and G. Robertson, "Perceptual user interfaces," *Commun. ACM*, vol. 43, no. 3, pp. 32–34, Mar. 2000.
[71] C. Neidle, S. Sclaroff, and V. Athitsos, "SignStream: A tool for linguistic and computer vision research on visual–gestural language data," *Behavior Res. Meth., Instrum., Comput.*, vol. 33, no. 3, pp. 311–320, 2001.
[72] M. Betke and W. J. Mullally, "Preliminary investigation of real-time monitoring of a driver in city traffic," in *Proc. Int. Symp. Intell. Veh.*, Dearborn, MI, Oct. 2000, pp. 563–568.
[73] Q. Ji and X. Yang, "Real-time eye, gaze, and face pose tracking for monitoring driver vigilance," *Real-Time Imaging*, vol. 8, no. 5, pp. 357–377, Oct. 2002.
[74] D. Tock and I. Craw, "Tracking and measuring drivers' eyes," *Image Vis. Comput.*, vol. 14, no. 8, pp. 541–547, 1996.

John J. Magee received the B.A. degree from Boston College, Chestnut Hill, MA, in 2001 and the M.A. degree from Boston University, Boston, MA, in 2004. He is currently working toward the Ph.D. degree in the Computer Science Department, Boston University.

He has previously worked for DaimlerChrysler Research, Ulm, Germany, and for The Boston Red Sox. His research interests include computer vision and its applications for human–computer interfaces.

Margrit Betke received the Ph.D. degree in computer science and electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1995.

She is an Associate Professor of computer science with the Computer Science Department, Boston University, Boston, MA, where she coleads the Image and Video Computing Research Group, and was a Research Scientist with the Massachusetts General Hospital, Boston, MA, and the Harvard Medical School, Boston, MA. She conducts research in computer vision and is particularly interested in real-time applications to rehabilitation engineering and human–computer interaction. She has published over 50 original research papers.

Dr. Betke received the National Science Foundation Faculty Early Career Development Award in 2001 for developing "Video-based Interfaces for People with Severe Disabilities." She was one of two academic honorees of the "Top 10 Women to Watch in New England Award" by Mass High Tech in 2005.

James Gips received the S.B. degree from the Massachusetts Institute of Technology (MIT), Cambridge, and the M.S. and Ph.D. degrees in computer science from Stanford University, Stanford, CA.

He is the John R. and Pamela Egan Professor of Computer Science with the Information Systems Department, Boston College, Chestnut Hill, MA. In Spring 2004, he was a Visiting Professor at the MIT Media Laboratory, Cambridge. His research centers on developing new assistive technologies for people with severe disabilities.

Dr. Gips is a 2007 winner of the da Vinci Award, "honoring exceptional design and engineering achievements in accessibility and universal design, that empowers people of all abilities."

Matthew R. Scott received the B.A. degree in computer science from Boston University, Boston, MA.

He is currently a Software Engineer with Microsoft, Redmond, WA. His research interests include artificial intelligence and computer vision with an emphasis on human–computer interaction.

Benjamin N. Waber received the B.A and M.A. degrees in computer science from Boston University, Boston, MA. He is currently working toward with the Ph.D. degree in the Human Dynamics Group, The Media Laboratory, Massachusetts Institute of Technology, Cambridge.

He was with the Naval Research Laboratory, Washington, DC, and the Hitachi Central Research Laboratory, Japan. His current research interests include social and sensor networks, prediction mechanisms, and information flow.

Mr. Waber won the Department's Research Award for Undergraduate Students, Computer Science Department, Boston University.