A Multi-Gesture Interaction System Using a 3-D Iris Disk Model for Gaze Estimation and an Active Appearance Model for 3-D Hand Pointing

Michael J. Reale, *Student Member, IEEE*, Shaun Canavan, *Student Member, IEEE*, Lijun Yin, *Senior Member, IEEE*, Kaoning Hu, and Terry Hung

***Abstract*—In this paper, we present a vision-based human–computer interaction system, which integrates control components using multiple gestures, including eye gaze, head pose, hand pointing, and mouth motions. To track head, eye, and mouth movements, we present a two-camera system that detects the face from a fixed, wide-angle camera, estimates a rough location for the eye region using an eye detector based on topographic features, and directs another active pan-tilt-zoom camera to focus in on this eye region. We also propose a novel eye gaze estimation approach for point-of-regard (POR) tracking on a viewing screen. To allow for greater head pose freedom, we developed a new calibration approach to find the 3-D eyeball location, eyeball radius, and fovea position. Moreover, in order to get the optical axis, we create a 3-D iris disk by mapping both the iris center and iris contour points to the eyeball sphere. We then rotate the fovea accordingly and compute the final, visual axis gaze direction. This part of the system permits natural, non-intrusive, pose-invariant POR estimation from a distance without resorting to infrared or complex hardware setups. We also propose and integrate a two-camera hand pointing estimation algorithm for hand gesture tracking in 3-D from a distance. The algorithms of gaze pointing and hand finger pointing are evaluated individually, and the feasibility of the entire system is validated through two interactive information visualization applications.**

***Index Terms*—Gaze estimation, hand tracking, human–computer interaction (HCI).**

I. Introduction

THE ideal human-computer interaction (HCI) system should function robustly with as few constraints as those found in human-to-human interaction; moreover, it should map human gestures to application control in the most natural and intuitive way possible. Knowledge of eye gaze and point-of-regard offers insight into the user's intention and mental focus, and consequently this information is vital for the next generation of user interfaces [1], [2]. Applications in this vein can be found in the fields of HCI, security, advertising, psychology, education, and many others [1], [3]. While eye gaze gives us a more precise yet perhaps more transient idea of the user's focus, head pose gives a coarser but more committed approximation of the user's region of interest. As such, head pose has been leveraged directly for coarse gaze estimation [2], video game control [4] and navigation [5], and screen magnification [6]. Finally, hand pointing, one of the most common hand gestures, shows us the region that the user wishes another entity to focus on, irrespective of whether the user is actually looking at the point himself/herself. In an HCI context, this naturally maps to command control [7], and a few sample applications in this line include navigation in a 3-D world [8] and TV control [9].

While many approaches use individual gestures for specific tasks, it is uncommon to use them simultaneously. In this paper, we propose a new algorithm for eye gaze estimation from a distance using a 3-D eyeball/iris disk model, and we also present a new hand pointing estimation approach in 3-D space. We use the combination of four different gesture-based inputs (eye gaze, head pose/position, hand pointing direction, and mouth opening/closing) to develop a multi-gesture interaction system. We demonstrate the feasibility and utility of our system through two application case studies. One application is the 3-D Orb File Navigator, and the other is a geographic information visualization program (informally known as the "Midgard viewer"). The overall system has the potential to be extended into many different HCI applications in the educational, entertainment, business, and military sectors. We would also argue that our system allows gesture input on a more detailed level than that which the current wave of commercial gesture control systems, such as the Microsoft Xbox Kinect [10], is able to provide. Specifically, we track eye gaze and mouth movement, and our hand tracking system incorporates a more precise model of the hand. Moreover, we use only regular, visible-spectrum cameras. In contrast, systems like the Kinect are robust and reliable but at present only provide very coarse gesture control (e.g., whole body movement); they also require special hardware (i.e., depth cameras). It is our belief that a combination of coarse and fine grained control will be desirable in the next generation of gesture input device systems.

This paper is organized as follows: in Section II, we will review the related work, and then we will provide an overview

Manuscript received November 01, 2010; accepted January 24, 2011. Date of publication February 28, 2011; date of current version May 18, 2011. This work was supported in part by the Air Force Research Laboratory (FA8750-08-1-0096), in part by the National Science Foundation (IIS-1051103), and in part by the New York State Science and Technology Office (C050040). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Zhengyou Zhang.

M. J. Reale, S. Canavan, L. Yin, and K. Hu are with the Department of Computer Science, State University of New York at Binghamton, Binghamton, NY 13902 USA (e-mail: lijun@cs.binghamton.edu).

T. Hung is with Corning Corp., Taichung City 40763, Taiwan.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org. Part 1 of the video demo is an evaluation of both the eye gaze tracking and hand pointing tracking. The total size is 11.08 MB. Part 2 of the video demo shows two applications of our multi-gesture interaction system in action. The total size is 19.97 MB.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2011.2120600

of the proposed system in Section III. Details of the new algorithms for gaze estimation and hand pointing estimation will be described in Sections IV and V, respectively. The experiments and performance evaluations will be reported in Section VI, followed by a system usability study with two applications outlined in Section VII. Finally, the system's limitations and future work will be discussed in Section VIII.

II. Related Work

A. Eye Tracking and Gaze Estimation

There has been intensive research on eye tracking and gaze estimation for the past 30 years [3]. However, with the conventional single-camera system, either the user must keep his or her head locked in place so that it remains within the narrow field of view, the camera must be strapped to the subject's head, or some other marker like a small infrared light must be worn by the subject [11]. The resulting situation can be prodigiously inconvenient and uncomfortable for the user. Because of the unique reflective properties of the retina to infrared (IR) light, some systems have opted to detect the face by first finding the eyes [11]–[13]. While robust to visible light conditions, these methods have issues with changes in head pose, reflections off glasses, and even decreased reflectivity of the retina from contact lenses [14]. An alternative approach is to use stereo cameras [11], [13]; while robust, these systems generally require substantial calibration time. Some complex systems (like "smart rooms" [15]) require expensive and/or non-portable setups. Even existing, non-stereo, two-camera systems [16] often restrict the user to a preset location in the room.

Therefore, we propose a system that first robustly locates the face and then provides a rough approximation of the eye positions from a static, wide-angle camera view. The system then uses this information to guide an active pan-tilt-zoom camera to focus in on the eye area of the detected face. This allows the user translational freedom relative to the camera, and a simple adjustment of parameters also permits a range of movement in depth. Our system differs from the majority of other systems in this vein in that we do not use infrared, stereo cameras, specially-constructed hardware (like Noureddin *et al.* [12]), or specific room setups (i.e., "smart rooms"). We intend to leverage this system for gaze and point-of-regard (POR) estimation.

The overwhelming majority of gaze estimation approaches for constructing 2-D or 3-D gaze models rely on glints—reflections of light off the cornea [3]. Alternatively, eye gaze may be determined from the pupil or iris contours [17] using ellipse fitting approaches [18], [19]. With a sufficiently large image of the eye, the iris contour and reflection of the screen off the cornea can be used to determine gaze [20]. One can also leverage the estimated iris center directly and use its distance from certain reference points (e.g., the eye corners) for gaze estimation [21], [22]. Indeed, the entire eye region may be segmented into the iris, sclera (white of the eye), and the surrounding skin; the resulting regions can then be matched pixel-wise with 3-D rendered eyeball models using different parameters [23], [24]. However, different subjects, head pose changes, and lighting conditions could significantly diminish the quality of the segmentation [24].

To bypass the issue of the diminished accuracy of iris center detection from visible light imagery while simultaneously avoiding the problems heir to the potential instability of iris contour extraction, we leverage both to determine eye gaze direction with a 3-D eyeball model. We calculate the 3-D eyeball centers, radii, and fovea points through a unique calibration approach. During gaze estimation, we map both the estimated iris center and the iris contour points onto the eyeball sphere. After converting these points into vectors starting at the eyeball center, we then use a least-squares approach to find a vector that is at a known angle from the contour vectors while also being approximately in line with the iris center vector. We then find the current position of the fovea, which is dependent on the rotation of the eye, compute a new iris center from our initial gaze vector, and use the vector from the fovea to this new iris center as our final gaze direction. Our approach differs from the majority of previous techniques, in that *we map the 2-D contour pixels onto the known 3-D eyeball; in contrast, most approaches in this vein project the 3-D model into the 2-D plane*.

B. Hand Detection and Tracking

Hand gestures offer an efficient means of human–computer interaction [7], [10], [25], [26], and the simplest, most basic gesture is pointing. The pointing gesture can resolve ambiguities springing from verbal communication, thus opening up the possibility of humans interacting or communicating intuitively with computers or robots by indicating objects or pointed locations either in 3-D space or on the screen. However, it is a challenging task to estimate the 3-D hand pointing direction automatically and reliably from streams of video data due to the great variety and adaptability of hand movement and the undistinguishable features of the joints on the hand. Some previous work shows success in hand detection and tracking using multi-colored gloves [27], depth-aware cameras [28], background subtraction [29], color-based detection [28], [30], stereovision-based approaches [31]–[33], or binary-pattern-based hand feature detection [34], [35]. However, the big challenge remains to accurately detect and track the hand in spite of various hand rotations.

Motivated by recent advances in feature detection [28], [34]–[39], we propose a novel technique to estimate pointing direction based on two orthogonal-view cameras. Here, we only focus on the gesture of hand pointing. We set up two regular cameras in orthogonal positions, one on the top of the user, and the other to the left side. Unlike binary-pattern-based approaches [34], [35] which are limited to a certain degree of hand rotation, we propose a hand image warping approach to transform the original hand image to a polar-coordinate plane in order to make the hand detection invariant to orientation. We apply two cascade detectors based on binary pattern features and AdaBoost [40] for hand region detection in the two views. Then, we use the active appearance model (AAM) [41] to track the finger points to identify the direction the hand is pointing. Extending the idea of AAM face tracking to track landmark features of hands, we are able to infer the orientation of a pointing finger in 3-D space. This is done via two simultaneous captures of the hand. Using the correspondence between the points of a hand in two orthogonal views allows us to infer the (x, y, z) coordinates of those points, thus resulting in a finger vector by connecting two points along

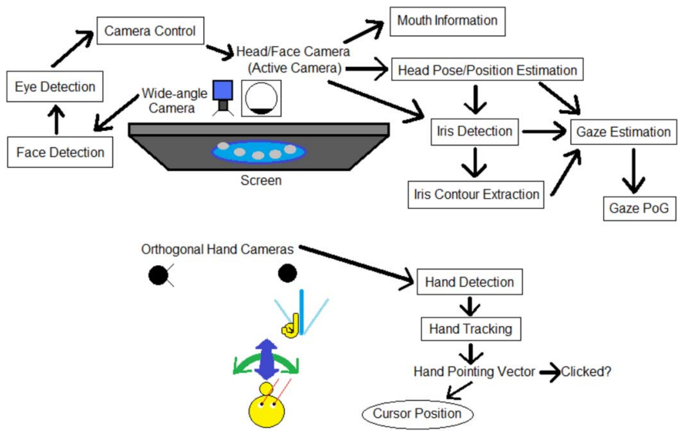


Fig. 1. System overview.

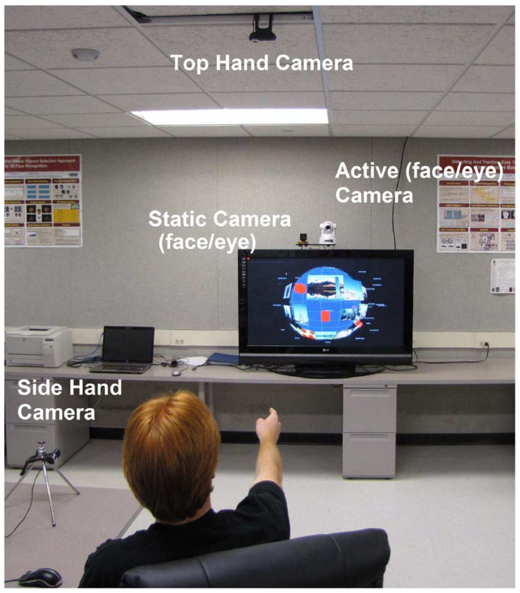


Fig. 2. Complete system in action.

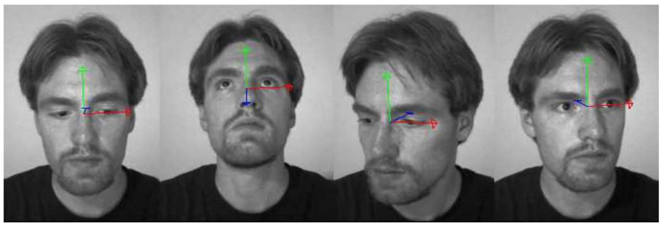


Fig. 3. Head pose and position estimation.

the finger. Furthermore, the 3-D finger position can also be mapped to the 2-D screen space.

III. System Overview

Our multi-gesture interaction system consists of five major components: face detection, eye area detection (wide-angle view), iris and contour detection (narrow-angle view), gaze calibration/estimation (narrow-angle view), and hand pointing detection. We use a static camera for the wide-angle view; images from this camera are used for face detection and eye detection. Once the face and eye area centers are detected, the detection program controls an active camera with pan/tilt/zoom functions to capture the face area. This active camera gives us a close-up view of the face which can then be used for iris detection, iris contour detection, and gaze estimation. After gaze calibration, the gaze directions and starting points (i.e., foveae) will be estimated and then mapped to screen coordinates. Another important component is hand pointing gesture detection. We set up two orthogonal cameras from the top view and side view to detect hand regions, track the finger pointing features, and estimate the pointing directions in 3-D space. Fig. 1 depicts the system composition, while Fig. 2 shows the system in action.

A. Face Detection

We apply an appearance-based technique based on the work by Viola and Jones [40] for face detection. To improve the robustness of the face detector, we have developed a novel, linear-time mechanism to make the system invariant to variable lighting conditions: all features are scaled by a ratio of the average gray-level intensity of the training samples over the average gray-level intensity of the current search region. We use the integral image to efficiently compute feature block intensity levels. The best Haar-like features are selected with Adaboost; we then used Information Gain in the Gain Ratio as a metric of usefulness. The details of the developed face detector can be found in [42].

B. Eye Detection (Static Camera)

Once the face is detected, we also perform eye detection using the so-called topographic features of eyes presented in our previous work [19]. The basic idea is to create a terrain map from the gray-scale image, effectively treating it like a continuous 3-D surface, and extract "pit" locations as pupil candidates. The actual eye pair is chosen using a Gaussian mixture model (GMM) and certain heuristics (e.g., the eye candidates cannot be vertical). The approximate center of the two detected eye candidates is used as the focus of the pan-tilt-zoom camera. Note that we do not use the mutual information tracking approach presented in [19] and instead detect the eyes once every time the face is detected in the wide-angle view.

The above stage provides an approximate localization of eyes, allowing the active camera to zoom into the region of interest. To detect the accurate positions of irises and further estimate the viewing direction, we propose a new approach for 3-D eye gaze estimation based on a 3-D iris model, which will be described in the next section.

IV. Gaze Estimation From the Active Camera View

In order to determine the gaze direction, we need to obtain a more precise estimate of the iris center as well as find points on the iris contour. This information will be mapped to a 3-D eyeball model, which consists of its position, radius, iris size, and fovea position offset. The current eyeball position is determined by an offset from the 3-D head pose and position; the latter is given by an existing library [43] (as shown in Fig. 3). We then compute the optimal eyeball rotation that fits our model expectations, and we use the rotated 3-D fovea as our gaze starting point

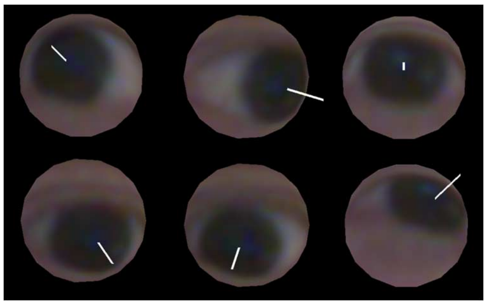


Fig. 4. 3-D rendered eyeballs (white line indicates optical gaze direction).

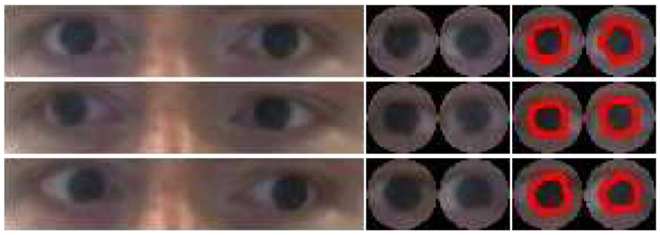


Fig. 5. 3-D iris detection and contour extraction. (Left column) Original 2-D image. (Middle column) 3-D eyeballs rendered with the iris looking into the camera. (Right column) Iris contours, shown as red dots between the iris and the rest of the eyeball, found on rendered eyeball image.

and the vector from the fovea to the 3-D iris as our gaze direction. We will now describe the algorithms used for iris center and contour detection, eyeball model calibration, and gaze estimation, respectively.

A. Iris Center and Contour Detection

Our eye detection algorithm maps the current camera image as a 2-D texture onto the current position of the 3-D eyeballs, rotates the eyeballs in pitch and yaw, renders each rotated eyeball, and picks the rotated eyeball that looks most like the user is looking into the camera. This is evaluated by 1) computing the absolute pixel difference of the center region of the rendered eyeball from a dark, circular template and 2) circle fitting on the gradient magnitude image. Once the best eyeball rotation and scale are determined, we rotate the eyeball back and project into image space, giving us our 2-D iris center. Fig. 4 shows some sample 3-D eyeballs rendered at different angles. Note that this approach differs from [24] in that 1) we map from image to model rather than from model back to image, 2) we do not search for the eyeball center at this juncture, relying instead on the 3-D eyeball offsets and thus making the approach potentially more efficient, and 3) we do not perform segmentation, which is an unreliable process at best.

For iris contour extraction, we effectively shoot rays outwards from the center of each optimally-rendered eyeball image, in an approach analogous to [44]. For each point along each ray, we compute its score:

$$score_{i,j} = m_{i,j} \times [(H_i \bullet D_{i,j}) > 0] \tag{1}$$

where i is the ray index, j is the pixel index, $m_{i,j}$ is the gradient magnitude, H_i is the normalized 2-D vector shooting outwards from the iris center estimate, and $D_{i,j}$ is the normalized gradient direction vector from dark to light pixels. We initially choose the highest scoring point within a certain radius range, and then we eliminate all points falling outside of a more restrictive range to remove noise points from the eyelids and specular highlights on the iris. The right-most column of Fig. 5 shows some examples of iris contours.

To eliminate eyelid points, we note that the eye corners are approximately at the horizontal poles of the eyeball; moreover, the opening of the eyelids should be in line with the head pose vector. We therefore rotate the contour points such that the camera z axis is in line with the head pose vector and split the points into two groups, one for each eyelid. We convert these points into vectors from the eyeball center, drop the x components, and normalize the vectors. We then iteratively compute the average vector among remaining points and remove all points inside of the current "eyelid's" range. We repeat this process until the number of points remaining does not change. We are thus left with two vectors from the center of the eyeball, one for each eyelid, in line with the head pose. Remaining contour vectors outside of the eyelids are eliminated, as are points too close to the eyelids. The entire eyelid-finding procedure is performed twice for each eyelid on each eye.

Since each eyeball is rendered to a certain image size, the approach is scale-invariant. In all cases, we work on the red channel of the image, since the iris (unlike the surrounding skin areas) generally has very low red intensity values [45]. Prior to eyeball calibration, we use an average eyeball radius of 1.25 cm [3]. After calibration, however, the true eyeball parameters are used.

B. Gaze Calibration

Our calibration procedure has two stages: the first acquires an estimate for the 3-D eyeball center and radius, and the second refines our initial estimate while also extracting the iris radius and the position of the fovea on the eyeball surface. Note that each eyeball is handled independently during calibration.

1) Stage 1 Calibration: Initial Estimate: Given a 3-D eyeball radius r, we want to find an estimate for the 3-D eyeball center E (as shown in Fig. 6). Assuming the intrinsic camera parameters are known, any 2-D pixel position can be converted into a 3-D vector in world space. We operate under the assumption that, if the user is looking straight into the camera, the eyeball center must be somewhere along a 3-D vector starting at the camera center and going through the 2-D iris center; that is, the gaze direction goes from the 3-D eyeball center E to the 3-D iris center E', a vector known as the *optical axis*. Granted, this is not strictly true, since the correct gaze direction (i.e., the *visual axis*) is a vector from the fovea through the pupil [3], and this visual axis is offset from the optical axis by approximately 4–8° [46]. However, we will correct for this in Stage 2.

The user first looks into the camera and then at a calibration point P whose 3-D location is known. This gives us 2-D iris locations $m1$ and $m2$, respectively. These points are then converted into normalized 3-D perspective vectors A and B. We want to find two points, E (the eyeball center) along vector A and E' (the iris center) along vector B, such that the distance between them is r (the radius of the eyeball) and the vector between them points at P. Fig. 6 illustrates this idea. Let t_a and t_b

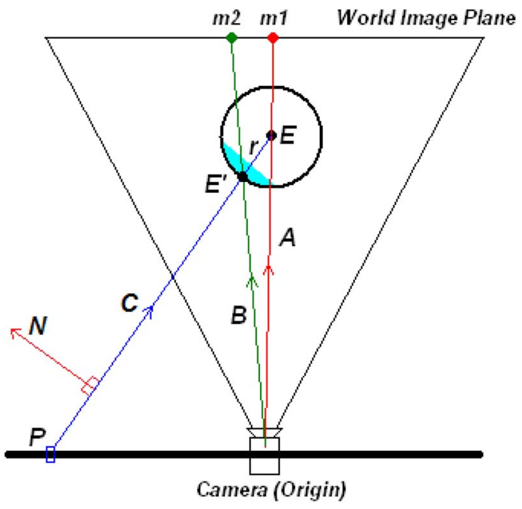


Fig. 6. Stage 1 calibration.

represent the lengths such that $At_a = E$ and $Bt_b = E'$; we can then express our first constraint as follows:

$$||At_a - Bt_b|| = r. \tag{2}$$

Point E' (or Bt_b) is the point of intersection between $C = At_a - P$ and B. This is equivalent to the point of intersection between B and a plane determined by point P and normal $N = (A \times B \times C)$. So, our second constraint is as follows:

$$t_b = \frac{N \bullet P}{N \bullet B}. \tag{3}$$

Given above (2) and (3) with two unknowns t_a and t_b, we can derive the following quartic formula and solve for t_a:

$$St_a^4 + Tt_a^3 + Ut_a^2 + Vt_a + W = 0 \tag{4}$$

where

$$S = Y^2 \tag{5}$$
$$T = -2YZ - 2(A \bullet B)XY \tag{6}$$
$$U = Z^2 + 2(A \bullet B)XZ - r^2Y^2 + X^2 \tag{7}$$
$$V = 2r^2YZ \tag{8}$$
$$W = -r^2Z^2 \tag{9}$$

and

$$X = (A \times B \times A) \bullet P \tag{10}$$
$$Y = (A \times B \times A) \bullet B \tag{11}$$
$$Z = (A \times B \times P) \bullet B. \tag{12}$$

The derivation of (4) and its solution can be found in the Appendix.

Once we have t_a, we can scale A by this value and get a 3-D eyeball center estimate for a given radius. We cycle through the natural range of human eyeball radius sizes (1.2 cm to 1.3 cm) [3] in 1/10th millimeter increments and get an eyeball estimate for each radius size. During the calibration procedure, we have the user look into the camera first, and then look at two calibration points, one near the upper left corner of the screen and one near the upper right corner. Since we assume we know the screen position, size, and orientation relative to the camera, we have the 3-D coordinates of any pixel position on the screen. The calibration point color slowly transitions from red to blue and back to red again to attract the user's attention [47]. The first calibration point is used to get our B vector. For each eyeball estimate and each sample of the two calibration points, we use our gaze estimation approach to determine the estimated point of regard and get its distance from the true gaze point. These error distances are added together, and the eyeball estimate with the smallest error is chosen. Should gaze estimation fail on a sample, a penalty term p is added:

$$p = \frac{100 \times (2.0 - d^2)}{r} \tag{13}$$

where d is the dot product of the iris center's 3-D normalized perspective vector and the normalized vector from the origin (camera) to the eyeball center, and r is the eyeball radius.

Note that this approach assumes that eyeball center (E), 3-D iris center (E'), and 3-D gaze point (P) are all coplanar, which may not be true if the user's head moves during the calibration procedure; thus, we must adjust our vectors for head rotation and translation before performing our calibration calculations.

2) Stage 2 Calibration: Refinement and Fovea Location: Due to the optical/visual axis offset as well as user error during calibration, our initial estimates can be off slightly from their true values. Therefore, we reset the radius to 1.25 cm and find the best position and radius using the Nelder-Mead downhill simplex method [48]. To ensure that the radius values remain reasonable, we add a penalty term if the radius is outside of the range 1.2–1.3 cm.

We also estimate the fovea position and iris radius. During gaze estimation, each 2-D contour point is converted to a 3-D world vector and intersected with the current eyeball sphere; we refer to a normalized vector going from the eyeball center to one of those intersection points as an "iris contour vector" C_i. The expected iris radius, therefore, can be stored as an expected dot product $(aveDot)$ between the optical axis G and each contour vector C_i. In this way, we define an "iris disk", illustrated in Fig. 7(a). Thus, for each position and radius estimate, we use the "look-into-camera" frames to estimate the current optical axis G and compute this expected dot product using an extension of our gaze estimation approach, as shown in (15). We also compute the fovea position as the intersection of the vector from the origin to the 3-D iris center with the back of the eyeball. We use this fovea position to compute a fovea offset such that, given the rotation angles of the optical axis, we can get the current fovea position.

We also add to the overall error term what we call the "optical error," a measure of how "circular" the contour points are on the surface of the eyeball. Given d_i as the dot product of each contour vector C_i with the optical axis G, this optical error is the average absolute difference of each d_i from $aveDot$. If the contour point does not intersect with the eyeball surface, we add 1.0 for that point's optical error.

When calibration is complete, we now have the 3-D eyeball center, the 3-D eyeball radius, the average dot product $(aveDot)$, and the fovea offset. The center can be rotated and translated with head position and pose information, and our

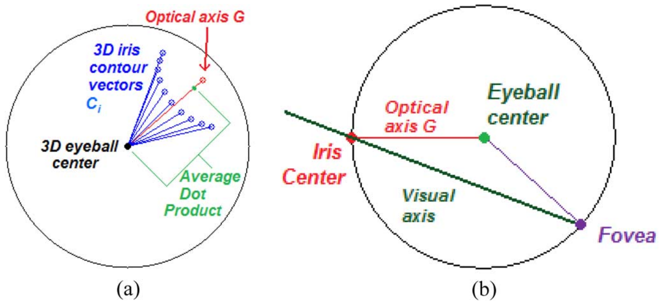


Fig. 7. (a) "Iris disk" concept: blue vectors are the "iris contour vectors," while the red vector is the computed (optical axis) gaze direction. (b) Optical versus visual axis.

fovea offset is stored as a vector offset from the optical axis. In both calibration stages, error values are scaled up to 1/10ths of a millimeter. Also, we use 15 samples for the look-into-camera stage, and 40 samples for each regular calibration point.

C. Gaze Estimation

For each frame, we adjust the eyeball center position based on the current head pose and position. Using the 2-D iris center, we want to find the iris contour points, map them to the eyeball sphere, determine the optical axis, rotate the fovea point accordingly, compute the new (3-D) iris center from the optical axis, and finally get the visual axis as the vector from the fovea to the new iris center. To estimate the optical axis G, we solve the following linear equations:

$$\begin{pmatrix} C_0^X & C_0^Y & C_0^Z \\ \vdots & \vdots & \vdots \\ C_N^X & C_N^Y & C_N^Z \\ V^X & V^Y & V^Z \\ \vdots & \vdots & \vdots \\ V^X & V^Y & V^Z \end{pmatrix} \begin{pmatrix} G^X \\ G^Y \\ G^Z \end{pmatrix} = \begin{pmatrix} aveDot \\ \vdots \\ aveDot \\ 1 \\ \vdots \\ 1 \end{pmatrix} \tag{14}$$

where the C_i vectors are the contour vectors described earlier, G is the optical axis, $aveDot$ is our expected dot product, and V is the normalized vector from the eyeball center to the iris center point mapped onto the eyeball surface. The basic idea is to find the optical axis G such that 1) it is parallel to V and 2) the dot product of G and each C_i is $aveDot$. Note that $aveDot$, V, and the constant 1 are repeated in their respective matrices N times, once for each contour vector. This unique approach of mapping 2-D contour points directly onto the 3-D eyeball sphere allows for efficient estimation of the gaze direction.

To get the visual axis, we rotate the fovea offset based on the rotation angles of the optical axis G. We intersect the optical axis with the eyeball sphere to get a new estimate for the 3-D iris center, and we take the normalized vector from the fovea to this new iris center as our final gaze direction [see Fig. 7(b)].

Once we have our current 3-D fovea position and gaze vector, we must map this information to screen coordinates. We assume that the screen's 3-D position, size, and orientation are already known, and so the mapping is a simple ray-plane intersection. We use the average position of the two foveae as our final starting point and the average of the two visual axis vectors as our final gaze direction; this is intersected with the screen to get our 2-D gaze point of regard.

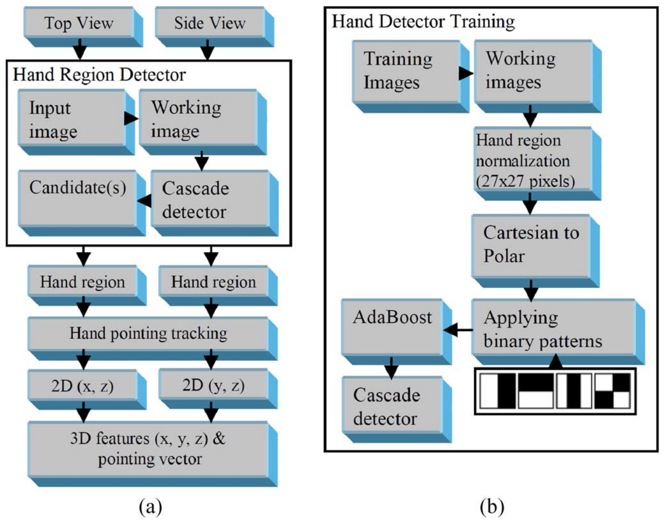


Fig. 8. (a) Diagram of pointing estimation system, including the hand region detector. (b) Hand detector training process.

As we mentioned earlier, we need to estimate the average dot product and optical axis from the look-into-camera samples during calibration. We do so with an extension of (14), resulting in (15):

$$\begin{pmatrix} C_0^X & C_0^Y & C_0^Z & -1 \\ \vdots & \vdots & \vdots & \vdots \\ C_N^X & C_N^Y & C_N^Z & -1 \\ V^X & V^Y & V^Z & 0 \\ \vdots & \vdots & \vdots & \vdots \\ V^X & V^Y & V^Z & 0 \end{pmatrix} \begin{pmatrix} G^X \\ G^Y \\ G^Z \\ aveDot \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix}. \tag{15}$$

Thus, both the optical axis G and $aveDot$ are determined by solving this equation.

V. Hand Tracking and Pointing Estimation

The 3-D hand pointing gesture is estimated from two camera views, as shown in Fig. 8(a). There are three major stages: 1) hand region detection, 2) hand feature tracking, and 3) estimation of the 3-D pointing direction.

A. Hand Region Detection

Hand region detection is the first step towards estimating the pointing direction. Motivated by the success of the face detection approach developed by Viola-Jones [40] using Haar-like features and an AdaBoost cascade detector, we extend the features to hand region detection for the pointing gesture. In order to get reliable hand features, we use color channel arithmetic to reduce the influence of the background. By observing that the skin color has a much stronger red component than its green and blue components, we compute the image by

$$I(x, y) = R(x, y) - \max\left[G(x, y), B(x, y)\right]. \tag{16}$$

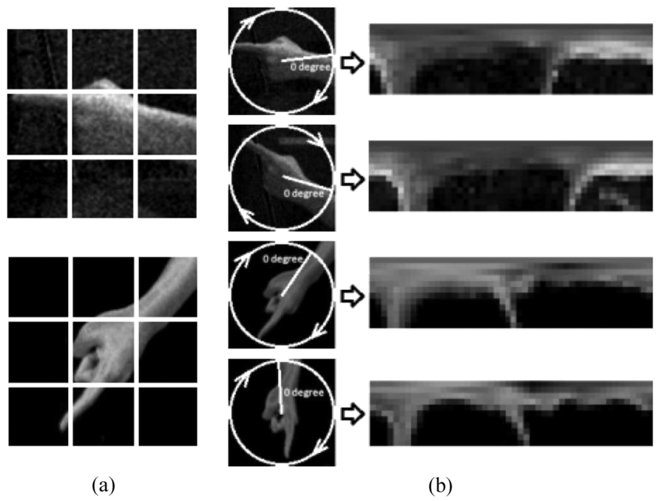


Fig. 9. (a) Hand wrist estimation by 3 × 3 blocks division. (b) Image conversion from Cartesian to polar coordinates, where the upper two rows are from the side view, while the lower two rows are from the top view of the hand.

This simple process can roughly highlight the skin area. As a result, a working image I is generated, based on which subsequent operations using the binary pattern approach can be carried out effectively.

1) Hand Image Warping Using Polar Coordinates: Some existing work has applied pre-designed binary patterns for hand detection successfully [35], [37]. However, the detection is still sensitive to the variation of hand orientations. The report in [34] shows that only 15° of hand rotation can be detected by applying the Viola-Jones-like approach [40]. To improve the orientation invariance of hand region detection, we propose to warp the hand image from Cartesian coordinates to polar coordinates. To do so, we use the center of the window as a pole ("o"), and the polar angle (θ) at 0° is determined by the position of the wrist [as illustrated in Fig. 9(b)]. The radius (r) of the polar axis is determined by the window size.

Since a hand is always connected with its corresponding wrist, in order to estimate the wrist position, we divide the 27 × 27 hand image into 3 × 3 blocks as shown in Fig. 9(a).

The fist is contained in the central block, and the wrist is located at one of the surrounding eight blocks. Due to the strong correlation of skin colors between hand and wrist, the average color of the block containing the wrist is the most similar to the average color of the central block among the eight surrounding blocks. Thus, we are able to identify the position of the wrist by comparing the average color of the eight blocks and the central block.

After the position of the wrist is determined, we use this position as the 0° polar coordinate, and convert the image from Cartesian (x, y) to polar coordinates (θ, r). Fig. 9(b) shows examples of the image warping from both views. As we can see, the converted images have similar appearances regardless of hand orientations rotated in the image plane.

2) Binary Patterns Based Hand Detection: After the image conversion, we apply the binary patterns as shown in Fig. 8(b) (four black-white patterns) to the warped image in (θ, r) coordinates. Fig. 10 illustrates an example of the binary patterns overlapping on the warped image.

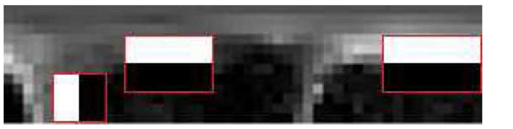


Fig. 10. Example of applied binary patterns.

Similar to the procedure used in [40], our hand detector training procedure performs the following three operations: 1) integral image generation, 2) Haar-like features generation using the above binary patterns, and 3) building cascade detector using AdaBoost. Fig. 8(b) shows a diagram of the training procedure, while Fig. 8(a) depicts the hand detection procedure as part of the 3-D pointing system.

After the detector has been built, it scans the input image in a brute-force way. All subwindows with different size and position in the image will be input to the detector. Once a subwindow has passed the detector, it will be marked as a candidate.

B. Hand Feature Tracking

Given the detected hand regions, we are able to track hand features in the limited search regions. We apply an active appearance model (AAM) [41] to track 21 pre-defined feature hand points on both the top and side camera views. AAM is a method of matching statistical models to images developed by Cootes *et al.* [41]. A set of landmark images is used to create the training set. The model parameters that control the shape and gray-level variation are subsequently learned from this training set.

The landmarks selected for the training set represent the shape of the object to be modeled. These landmarks are represented as a vector, and principal component analysis is applied to them. This can be approximated with the following formulas: $x = \overline{x} + P_s b_s$ and $g = \overline{g} + P_g b_g$ for texture. In the shape formula, $\overline{x}$ is the mean shape, P_s represents the modes of variation, and b_s defines the shape parameters. In the texture formula, $\overline{g}$ is the mean gray level, P_g represents the modes of variation, and b_g defines the gray-level parameters.

We use AAMs to create a statistical model of the hand from two orthogonal views via a simultaneous capture. We create a separate appearance model for each view, and we track the hand in the two views separately. To create the hand shape and gray-level models, we chose 21 landmarks for the training set images. These landmarks for the top and side views can be seen in Fig. 17. Note that the hand detection of the previous stage allows us to narrow down the search region for fitting our model to the hand, thus reducing the time for finding a correct fit.

C. Estimation of 3-D Pointing Direction

Since the two views of the hand are tracked separately with different models, we are able to create the best fit for the corresponding hand in each frame. There is correspondence between multiple landmarks in the separate views. Those landmarks, most notably on the finger, allow us to infer the 3-D coordinates from 2-D coordinates, and thus infer the 3-D orientation of the finger. For one point that has correspondence between the two models, we can use the top view as the (x, z) coordinate and the side view as the (z, y) coordinate. We can then combine both of the views to infer the (x, y, z) coordinates

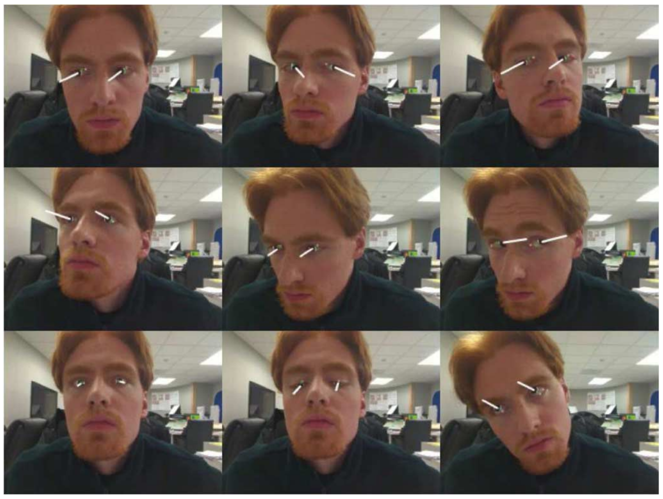


Fig. 11. Sample gaze results (web camera). Green circle is eyeball center, white circle is fovea, red circle is iris center, white lines are iris contours, and the large white lines are the gaze directions (visual axis).

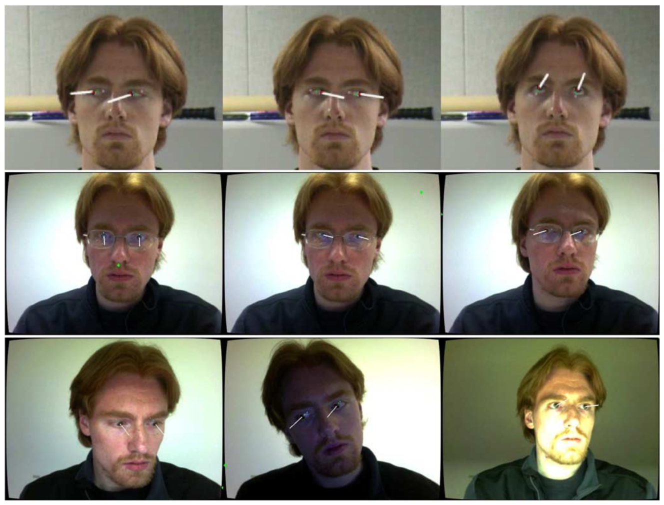


Fig. 12. (Top row) Sample gaze results from active camera. (Middle row) Sample gaze results with subject wearing glasses. (Bottom row) Sample gaze results under different lighting conditions. Note that calibration was performed only once in lighting conditions, the same as that of the left-most image.

for that tracked landmark. Since the z coordinate may not be the same in both of the views, we take the average of both values to give us a new z coordinate.

Once we have the 3-D coordinates of the tracked points, we take two points on the finger that are "connected by a line" to create a vector that points in the direction of the finger. The two points selected are near the top and bottom of the pointing finger. These were selected as they appear to give us the most reliable vector in determining the orientation of the finger. The other landmarks are used to create a better fit for the AAM search, as well as for future modeling of hand details. This vector is shown in Fig. 17 as a line pointing from the finger.

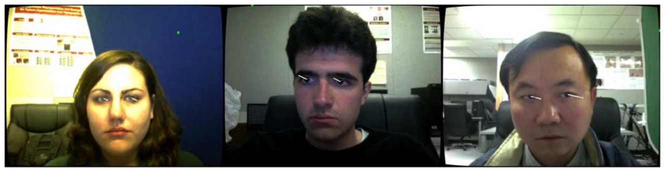
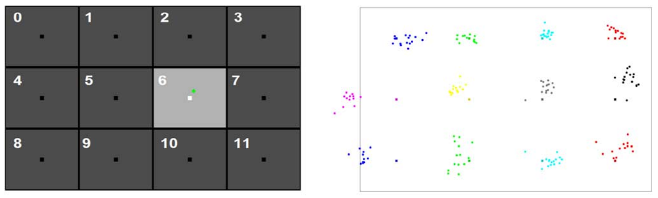


Fig. 13. Sample gaze results from different subjects.



(a) (b)

Fig. 14. (a) Gaze test grid. Markers glow white when active, and box becomes light gray when eye cursor enters region. Eye gaze cursor is the green diamond. The numbers are drawn here for clarity but were not drawn during the test. (b) Example of redrawn gaze points from one of our tests (the gray box is the screen region).

VI. Experiments and Evaluation

A. Eye Gaze Estimation

We tested our gaze calibration and estimation algorithm with various cameras and user-to-camera distances in our lab environment. Fig. 11 shows some samples of the gaze estimation results with the user's head rotated in front of a web camera, while the top row Fig. 12 presents the results from our active zoom camera while the subject was approximately 4 m away from the camera. The middle row of Fig. 12 also shows some sample gaze vectors when the subject was wearing glasses. The bottom row presents gaze results under different lighting conditions; however, please note that calibration was performed only once in lighting conditions that were the same as that of the left-most image of the bottom row. Finally, Fig. 13 shows some sample results from different subjects.

We also performed a gaze and point-of-regard estimation experiment wherein the user was asked to look at 12 gaze markers on the screen [effectively, the center of each brick in a 3 × 4 grid, as shown in Fig. 14(a)]. To ensure that our calibration points were at known locations relative to the camera, we used the built-in webcam in a laptop and measured the screen size as well as its position and orientation relative to the camera. Camera images were 640 × 480 pixels. Each marker was focused on for 2–4 s.

We evaluated the gaze estimation accuracy by having four subjects taken from students and faculty in our research lab perform a real-time gaze tracking test. We recorded the angular error, which is measured as the angle between the vector from the gaze starting point to the target point and the gaze direction vector, and the "hit percentage," which refers to how frequently the cursor was within the target block. Please note that a point going past the edge of the screen is still considered a "hit" on the gaze target block. Our results are presented in Table I; the overall angular error was 5.953°, and the average hit percentage was 90.54%.

We also tested each individual gaze target block across all subjects; the per-target average results are presented in Table II. The "Gaze Target" indices correspond to the numbers on the blocks in Fig. 14(a). The average angular error information in Table II is also presented in Fig. 15. Overall, the results show that the angular errors are reasonably small, while the hit percentage is fairly high for almost all the blocks.

TABLE I
GAZE ESTIMATION TEST RESULTS

Type	Mean
Angular Error	5.953°
Hit Percentage	90.54%

TABLE II
GAZE ESTIMATION TEST RESULTS: AVERAGE ANGULAR ERROR AND HIT PERCENTAGE FOR EACH TARGET BLOCK

Gaze Target	Angular Error	Hit Percentage
0	5.873°	100%
1	5.788°	89.86%
2	4.170°	98.57%
3	7.181°	100%
4	8.754°	98.48%
5	3.715°	83.33%
6	3.942°	60%
7	6.957°	96.55%
8	8.018°	98.21%
9	4.618°	73.24%
10	5.080°	92.06%
11	7.912°	100%

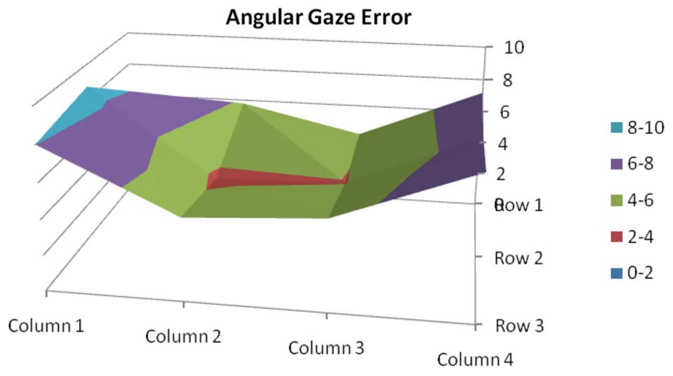


Fig. 15. Average angular gaze error for each target. Height units are in degrees.

Fig. 14(b) shows an example of redrawn gaze points from one of our tests.

The angular error results show promise, since natural light methods often average around 5° of gaze error [3]. The hit percentage results are also encouraging, and it demonstrates the potential usefulness of our gaze estimation approach in an HCI context.

B. Hand Pointing Estimation

In order to train two detectors for two views separately, we selected 107 positive image samples and 160 negative samples for the top view of the hands, and 128 positive samples and 145 negative samples for the side view. Fig. 16 shows examples of training images.

In the training stage, we applied the binary patterns to the converted images in the polar coordinate system, and we generated over 5000 features for each sample. Then, two cascade detectors were built based on the feature selection by AdaBoost. In the testing stage, after the input image is converted to the working image, each detector scans the working image in each view separately. Note that, during the hand region search, the integral image is computed locally on the warped image in the polar coordinate system. The experiments were conducted in our lab environment. The hand motion is in the range of $[-60°, +60°]$ for both pan and tilt. Comparing the detected hand regions with the manual selected ones, we achieved 90% and 91% correct detection rate for the top view (691 images) and side view (691 images), respectively. Those frames were collected from seven subjects in our department. Fig. 17 shows three sample frames from two views with the tracked points and the detected finger vectors. In addition, the estimated hand pointing orientations are also compared to the physically measured hand orientation during the time of capture. Among 691 frames, 628 frames show less than 6° difference between the two data sets. The correct pointing rate is 91%.

To analyze the robustness of the AAM tracking, we selected sequences of frames for both the top and side views of subjects in our lab. We manually selected 21 landmarks on all frames to use as ground truth. We then tracked those sequences using the AAM and compared those new tracked points to the ground truth. The ground truth landmarks were selected in a clockwise direction starting on the bottom of the forearm and continuing in order to the top of the wrist in the side view. The top view was selected in a similar fashion starting from the right side of the forearm and continuing in order to the left side. Following this ordering scheme, landmarks 13 and 15 are the main landmarks used to determine the 3-D pointing vector. It can be seen in Figs. 18 and 19 that these points are approximately in the range of 2 to 5 pixels of error for all subjects tested. We have found the rest of the landmarks' pixel error to be acceptable in fitting the AAM to the subject's hand.

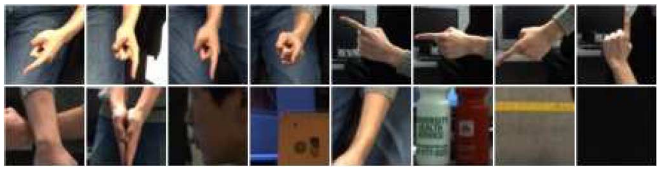


Fig. 16. Examples of hand training images. (Top row) Positive samples of top and side views. (Bottom row) Negative samples for both views.

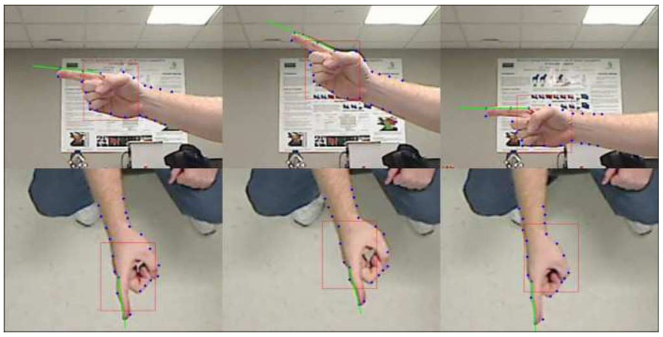


Fig. 17. Sample frames from a testing video showing detected hand regions (red blocks), tracked points (blue dots), and estimated pointing direction (green lines) from side view (upper row) and top view (lower row).

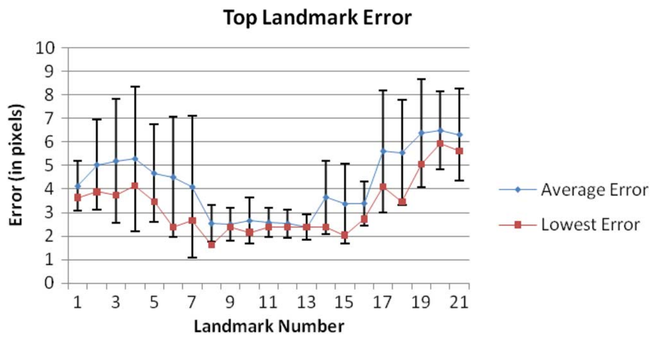


Fig. 18. Average errors of tracked frames from ground truth of top view (each vertical bar across a point is the standard deviation of the error at the point).

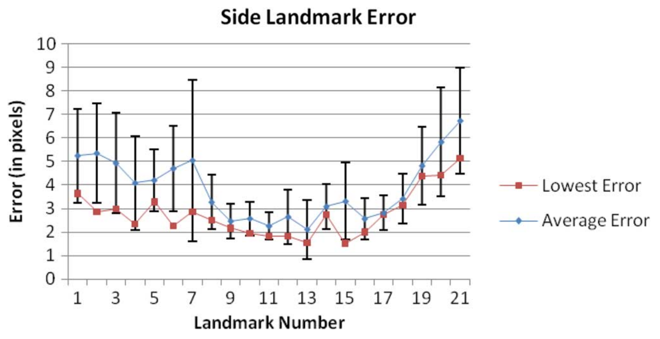


Fig. 19. Average errors of tracked frames from ground truth of side view.

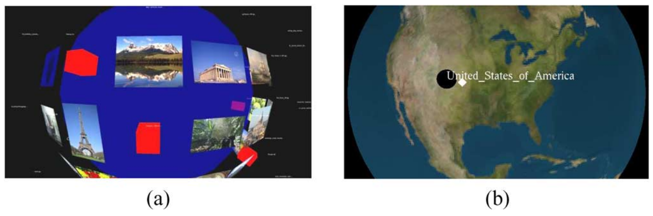


Fig. 20. (a) Orb 3-D File System Navigator. (b) "Midgard" Geographic Information Visualization Application.

VII. Validation: System Applications

To validate the feasibility and utility of the proposed new system, we performed case studies with two applications. The first is the Orb File System Navigator, which visualizes the file system as a 3-D sphere with files and subdirectories floating above the surface. The second is the "Midgard viewer," a geographic information visualization application that allows rotation and interaction with a globe of the earth.

A. 3-D File System Navigation

Rather than using the conventional system of folders and files, the Orb navigator visualizes the current directory as a large 3-D sphere (orb). Subdirectories are represented as smaller orbs on the surface of this larger orb, and opening a subdirectory is analogous to zooming into this smaller orb. Files are represented as cubes or, in the case of image files, as thumbnails floating over the surface of the larger orb [as shown in Fig. 20(a)]. If the user clicks on a subdirectory, the contents of that subdirectory are loaded, and this becomes the new current directory. If the user clicks on a file type, the system will open that file using whatever program is associated with that file type.

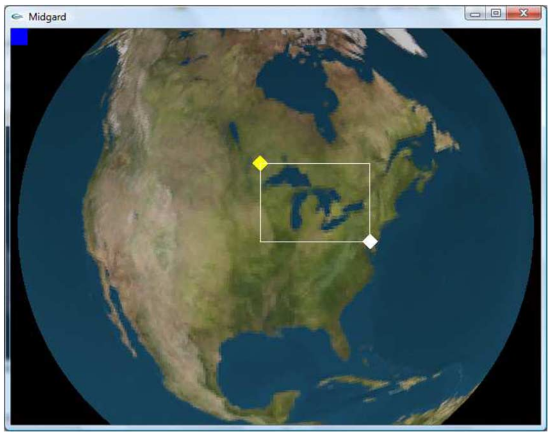


Fig. 21. Selecting a zoom region with the hand cursor (white) and eye gaze cursor (yellow) in the "Midgard" application.

Orb rotation is controlled by the pitch and yaw of the head. We first get the normalized vector from the 3-D head position to the center of the screen. Then, we compute the angle offset of this vector from the normalized head pose vector. If the offset pitch angle exceeds a certain threshold, the sphere is rotated about the x axis. A similar procedure is performed for yaw. Also, if the head moves backwards from its original position enough, the camera will begin to zoom out from the orb.

We map finger pointing to the mouse cursor movement. Since cursor movement even with a mouse is all relative (rather than absolute) motion, we normalize the pointing vector and map the x and y components directly onto the screen. When the AAM first initializes, we record the starting z position of the hand from the top view; if the user moves his/her hand forward by a certain distance from that starting z position, we interpret this as a mouse click.

The eye gaze point-of-regard is used as a second cursor. When we are in "zoom mode," the user can click with the primary cursor (shown as a white diamond) and drag a box with the second cursor (shown as a yellow diamond) to create a zoom region. When the "hand mouse" is released, the Orb is centered and zoomed to the selected region. The reason for using the eye cursor like this is that it creates the possibility of fast region zoom, since the eye can move much faster than the hand.

Using the vision library from [43], we also detect when the mouth is open or closed and allow a "mouth click." This switches between "selection mode" (where hand clicks open files and directories) and "zoom mode" (where the hand and eye cursor form a zoom region).

B. "Midgard" Geographic Information Visualization

The controls for Midgard are mapped in a similar way as with the Orb file viewer: head pose controls the rotation of the earth, head position allows the user to zoom out, mouth opening switches modes, the hand controls the primary cursor, and the eyes control the secondary cursor. However, when the user clicks in a region on the globe that is part of a country, the name of that country will appear, and the national anthem for that country plays. Fig. 20(b) shows the selection of a country with the hand cursor, while Fig. 21 shows selecting a zoom region with the hand and eye cursors.

TABLE III
GESTURE CONTROL MAPPING FOR TEST APPLICATIONS

Input Gesture	**Control**
Head pose change	Rotation of orb/globe
Head moves backwards	Zoom out
Hand pointing	Primary mouse cursor
Hand moves forward and back	Mouse click *Selection mode:* open file or folder / select country and play national anthem *Zoom mode:* hand moving forward starts defining region, hand moving back rotates and zooms to that region
Eye gaze	Secondary mouse cursor (used for defining zoom region)
Mouth opens and then closes	Toggle between Selection mode and Zoom mode

Table III defines the list of input gestures and their corresponding controls used for the two applications. In both cases, we have found that these gestures map naturally, intuitively, and effectively to their respective commands, as can be seen in our video demo. We believe that other applications could benefit from similar control mappings.

VIII. DISCUSSION AND CONCLUSION

In this paper, we have presented a real-time, vision-based system using multiple gestures for human–computer interaction. Moreover, we have proposed an eyeball center/fovea offset extraction procedure and gaze estimation approach, both of which are fully capable of dealing with head pose changes. The hand pointing gesture is estimated and mapped to the screen coordinate system and has replaced the mouse functions for intuitive interaction with the screen. We have also incorporated other gestures, such as head pose changes and mouth opening/closing, into the system, and we have performed case studies on two applications.

The current work still has a number of limitations:

1) Hand gesture is currently limited to only the pointing gesture.
2) The two-view camera setup limits the working space for hand detection. A system using a single camera's view is to be developed in the next step of the work.
3) Eyeball-based gaze estimation is limited by the accuracy of the head pose detection. The reliability and spatial discrimination of the gaze estimation algorithm need be further improved.

Other future work will include dealing with larger head pose angles and increasing the robustness of the iris and contour detection phases. Although the system can work even in the presence of glasses (as shown in the middle row of Fig. 12), heavy lens glare could skew the results [e.g., Fig. 22(b)]. We would also like to expand upon the secondary refinement pass to allow for a more accurate estimate of the fovea. Moreover, we plan to further develop our eyelid elimination algorithm, perhaps incorporating an approach analogous to that of [49]; this would help us address cases wherein the eyes are almost closed and the gaze results are negatively influenced, as shown in Fig. 22(a). Finally, future versions of the dual camera system will allow for dynamic depth changes while tracking the gaze directions of multiple subjects.

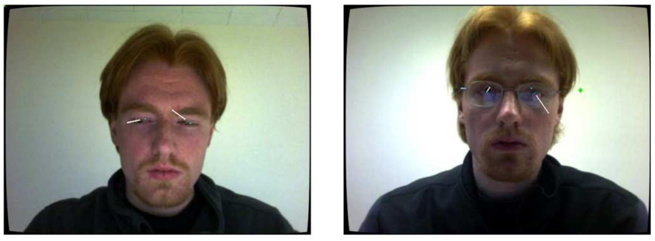


Fig. 22. (a) Irises are nearly obscured by the eyelids, causing the right eye gaze estimation to fail. (b) Heavy glare on the glasses causes an incorrect eye detection estimate.

APPENDIX A
SOLUTION FOR t_a AND t_b

Let t_a and t_b represent the lengths such that $At_a = E$ and $Bt_b = E'$, respectively. Our first constraint can be expressed as follows:

$$\|At_a - Bt_b\| = r. \tag{17}$$

We also know that the dot product of a vector with itself is its length squared; so, after multiplying out and gathering terms, we get

$$\begin{aligned}(At_a - Bt_b) \bullet (At_a - Bt_b) &= At_a \bullet At_a - At_a \bullet Bt_b \\ &\quad - At_a \bullet Bt_b + Bt_b \bullet Bt_b \\ &= t_a^2 - 2t_a t_b (A \bullet B) + t_b^2 \\ &= r^2. \end{aligned} \tag{18}$$

Let us now look at our other constraint. We want to find a vector going from P to At_a such that it intersects B at Bt_b. The intersection of $C = At_a - P$ and B is equivalent to the intersection of B with a plane with point P and normal $N = (A \times B \times C)$. So

$$t_b = \frac{N \bullet P}{N \bullet B}. \tag{19}$$

We work out N:

$$\begin{aligned} N &= A \times B \times C = A \times B \times (At_a - P) \\ &= (A \times B) \times At_a - (A \times B) \times P \\ &= t_a (A \times B \times A) - (A \times B \times P). \end{aligned} \tag{20}$$

Substituting this for N in (19)

$$\begin{aligned} t_b &= \frac{[t_a(A \times B \times A) - (A \times B \times P)] \bullet P}{[t_a(A \times B \times A) - (A \times B \times P)] \bullet B} \\ &= \frac{t_a(A \times B \times A) \bullet P - (A \times B \times P) \bullet P}{t_a(A \times B \times A) \bullet B - (A \times B \times P) \bullet B}. \end{aligned} \tag{21}$$

As it happens, however, $(A \times B \times P)$ is orthogonal to P, so the dot product of $(A \times B \times P)$ and P must be zero. Therefore

$$t_b = \frac{t_a(A \times B \times A) \bullet P}{t_a(A \times B \times A) \bullet B - (A \times B \times P) \bullet B}. \tag{22}$$

For simplicity, let

$$X = (A \times B \times A) \bullet P \tag{23}$$
$$Y = (A \times B \times A) \bullet B \tag{24}$$
$$Z = (A \times B \times P) \bullet B. \tag{25}$$

We can compute all of these ahead of time, so we now have t_b in terms of t_a:

$$t_b = \frac{t_a X}{t_a Y - Z}. \tag{26}$$

Substituting this into (18) yields

$$t_a^2 - 2t_a(A \bullet B)\left[\frac{t_a X}{t_a Y - Z}\right] + \left[\frac{t_a X}{t_a Y - Z}\right]^2 = r^2. \tag{27}$$

Multiplying by $(t_a Y - Z)^2$

$$(t_aY-Z)^2 t_a^2 - 2t_a(A\bullet B)t_aX(t_aY-Z) + t_aX^2 = r^2(t_aY-Z)^2. \tag{28}$$

Multiplying everything out

$$t_a^4Y^2 - 2YZt_a^3 + Z^2t_a^2 - 2(A\bullet B)XYt_a^3 + 2(A\bullet B)XZt_a^2 + t_a^2X^2 = r^2Y^2t_a^2 - 2r^2YZt_a + r^2Z^2. \tag{29}$$

Grouping like terms

$$(Y^2)t_a^4 + (-2YZ - 2(A \bullet B)XY)\, t_a^3 + \left(Z^2 + 2(A \bullet B)XZ - r^2Y^2 + X^2\right)t_a^2 + (2r^2YZ)t_a - r^2Z^2 = 0. \tag{30}$$

If we create the following variables:

$$S = Y^2 \tag{31}$$
$$T = -2YZ - 2(A \bullet B)XY \tag{32}$$
$$U = Z^2 + 2(A \bullet B)XZ - r^2Y^2 + X^2 \tag{33}$$
$$V = 2r^2YZ \tag{34}$$
$$W = -r^2Z^2. \tag{35}$$

We have after substituting into (30)

$$St_a^4 + Tt_a^3 + Ut_a^2 + Vt_a + W = 0. \tag{36}$$

We can compute X, Y, Z, S, T, U, V, and W ahead of time, leaving us with a quartic formula for t_a, which is solvable. Given t_a, we can use (26) to get t_b.

ACKNOWLEDGMENT

The authors would like to thank Mr. X. Zhang for his help on the experiments and video recording/editing.

REFERENCES

[1] Z. Zeng, M. Pantic, G. Roisman, and T. Huang, "A survey of affect recognition methods: Audio, visual, and spontaneous expressions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 1, pp. 39–58, Jan. 2009.

[2] S. Zhai, C. Morimoto, and S. Ihde, "Manual and gaze input cascaded (magic) pointing," in *Proc. CHI'99*, 1999, pp. 246–253.

[3] D. W. Hansen and Q. Ji, "In the eye of the beholder: A survey of models for eyes and gaze," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 478–500, Mar. 2010.

[4] S. Wang, X. Xiong, Y. Xu, C. Wang, W. Zhang, X. Dai, and D. Zhang, "Face-tracking as an augmented input in video games: Enhancing presence role-playing and control," in *Proc. CHI'06*, 2006, pp. 1097–1106.

[5] G. G. Mateos and S. F. Muñoz, "Tierra inhospita: Exploring a virtual world with your face," in *Proc. ACM SIGCHI Int. Conf. Advances in Computer Entertainment Technology (ACE'05)*, 2005, pp. 383–384.

[6] C. Harrison and A. K. Dey, "Lean and zoom: Proximity-aware user interface and content magnification," in *Proc. CHI'08*, 2008, pp. 507–510.

[7] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Comput. Vis. Image Understand.*, vol. 81, pp. 231–268, Mar. 2001.

[8] K. Tollmar, D. Demirdjian, and T. Darrell, "Navigating in virtual environments using a vision-based interface," in *Proc. Nordic Conf. Human-Computer Interaction (NordiCHI'04)*, 2004, pp. 113–120.

[9] M.-Y. Chen, L. Mummert, P. Pillai, A. Hauptmann, and R. Sukthankar, "Controlling your tv with gestures," in *Proc. Int. Conf. Multimedia Information Retrieval (MIR'10)*, 2010, pp. 405–408.

[10] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-based hand-gesture applications," *Commun. ACM*, vol. 54, pp. 60–71, Feb. 2011.

[11] D. Beymer and M. Flickner, "Eye gaze tracking using an active stereo head," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR'03)*, Jun. 2003, vol. 2, pp. 451–458.

[12] B. Noureddin, P. D. Lawrence, and C. F. Man, "A non-contact device for tracking gaze in a human computer interface," *Comput. Vis. Image Understand.*, vol. 98, pp. 52–82, Apr. 2005.

[13] T. Ohno and N. Mukawa, "A free-head simple calibration, gaze tracking system that enables gaze-based interaction," in *Proc. Symp. Eye Tracking Research & Applications (ETRA'04)*, 2004, pp. 115–122.

[14] K. Nguyen, C. Wagner, D. Koons, and M. Flickner, "Differences in the infrared bright pupil response of human eyes," in *Proc. Symp. Eye Tracking Research & Applications (ETRA'02)*, 2002, pp. 133–138.

[15] K. Bernardin, H. K. Ekenel, and R. Stiefelhagen, "Multimodal identity tracking in a smart room," *Personal Ubiq. Comput.*, vol. 13, pp. 25–31, Jan. 2009.

[16] J.-G. Wang and E. Sung, "Gaze determination via images of irises," *Image Vision Comput.*, vol. 19, no. 12, pp. 891–911, 2001.

[17] D. Xia and Z. Ruan, "IR image based eye gaze estimation," in *Proc. ACIS Int. Conf. Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'07)*, 2007, vol. 1, pp. 220–224.

[18] C. Colombo, D. Comanducci, and A. D. Bimbo, "Robust tracking and remapping of eye appearance with passive computer vision," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 3, pp. 2:1–2:20, Dec. 2007.

[19] J. Wang, L. Yin, and J. Moore, "Using geometric properties of topographic manifold to detect and track eyes for human-computer interaction," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 3, pp. 3:1–3:20, Dec. 2007.

[20] D. Schnieders, X. Fu, and K.-Y. Wong, "Reconstruction of display and eyes from a single image," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR'10)*, Jun. 2010, pp. 1442–1449.

[21] E. Pogalin, A. Redert, I. Patras, and E. A. Hendriks, "Gaze tracking by using factorized likelihoods particle filtering and stereo vision," in *Proc. Int. Symp. 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, 2006, pp. 57–64.

[22] J. Xie and X. Lin, "Gaze direction estimation based on natural head movements," in *Proc. Int. Conf. Image and Graphics (ICIG'07)*, 2007, pp. 672–677.

[23] H. Wu, Y. Kitagawa, T. Wada, T. Kato, and Q. Chen, "Tracking iris contour with a 3D eye-model for gaze estimation," in *Proc. Asian Conf. Comput. Vision—Volume Part I (ACCV'07)*, 2007, vol. Part I, pp. 688–697.

[24] H. Yamazoe, A. Utsumi, T. Yonezawa, and S. Abe, "Remote and head-motion-free gaze tracking for real environments with automated head-eye model calibrations," in *Proc. IEEE CVPR Workshop Human Communicative Behavior Analysis (CVPR4HB)*, Jun. 2008, pp. 1–6.

[25] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 677–695, Jul. 1997.

[26] Y. Wu and T. S. Huang, "Vision-based gesture recognition: A review," in *Proc. Int. Gesture Workshop Gesture-Based Communication in Human-Computer Interaction (GW'99)*, 1999, pp. 103–115.

[27] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM Trans. Graphics*, vol. 28, pp. 63:1–63:8, Jul. 2009.

[28] C. Manders, F. Farbiz, J. Chong, K. Tang, G. Chua, M. Loke, and M. Yuan, "Robust hand tracking using a skin tone and depth joint probability model," in *Proc. IEEE Int. Conf. Automatic Face & Gesture Recognition (FG'08)*, Sep. 2008, pp. 1–6.

[29] A. Utsumi, N. Tetsutani, and S. Igi, "Hand detection and tracking using pixel value distribution model for multiple-camera-based gesture interactions," in *Proc. IEEE Workshop Knowledge Media Networking (KMN'02)*, 2002, pp. 31–36.

[30] M. Lee, D. Weinshall, E. Cohen-Solal, A. Colmenarez, and D. Lyons, "A computer vision system for on-screen item selection by finger pointing," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR'01)*, 2001, vol. 1, pp. 1026–1033.

[31] C. Colombo, A. D. Bimbo, and A. Valli, "Visual capture and understanding of hand pointing actions in a 3-D environment," *IEEE Trans. Syst., Man, Cybern. B*, vol. 33, no. 4, pp. 677–686, Aug. 2003.

[32] N. Jojic, B. Brumitt, B. Meyers, S. Harris, and T. Huang, "Detection and estimation of pointing gestures in real-time stereo sequences," in *Proc. IEEE Int. Conf. Automatic Face & Gesture Recognition (FG'00)*, 2000, pp. 468–475.

[33] Y. Yamamoto, I. Yoda, and K. Sakaue, "Arm-pointing gesture interface using surrounded stereo cameras system," in *Proc. Int. Conf. Pattern Recognition (ICPR'04)*, 2004, vol. 4, pp. 965–970.

[34] M. Kolsch and M. Turk, "Analysis of rotational robustness of hand detection with a Viola-Jones detector," in *Proc. Int. Conf. Pattern Recognition (ICPR'04)*, 2004, vol. 3, pp. 107–110.

[35] T. T. Nguyen, N. D. Binh, and H. Bischof, "An active boosting-based learning framework for real-time hand detection," in *Proc. IEEE Int. Conf. Automatic Face & Gesture Recognition (FG'08)*, Sep. 2008, pp. 1–6.

[36] L. Cinque, M. Cupelli, and E. Sangineto, "Fast viewpoint-invariant articulated hand detection combining curve and graph matching," in *Proc. IEEE Int. Conf. Automatic Face & Gesture Recognition (FG'08)*, Sep. 2008, pp. 1–6.

[37] M. Kolsch and M. Turk, "Robust hand detection," in *Proc. IEEE Int. Conf. Automatic Face & Gesture Recognition (FG'04)*, May 2004, pp. 614–619.

[38] K. Oka, Y. Sato, and H. Koike, "Real-time fingertip tracking and gesture recognition," *IEEE Comput. Graphics Appl.*, vol. 22, no. 6, pp. 64–71, Nov./Dec. 2002.

[39] C.-B. Park, M.-C. Roh, and S.-W. Lee, "Real-time 3D pointing gesture recognition in mobile space," in *Proc. IEEE Int. Conf. Automatic Face & Gesture Recognition (FG'08)*, Sep. 2008, pp. 1–6.

[40] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, pp. 137–154, May 2004.

[41] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 681–685, Jun. 2001.

[42] T. Hung, "Real-time face detection and applications," M.S. thesis, Binghamton Univ., Binghamton, NY, 2008.

[43] Seeing machines faceapi. [Online]. Available: http://www.seeingmachines.com/product/faceapi.

[44] D. Li, D. Winfield, and D. Parkhurst, "Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches," in *Proc. IEEE CVPR Workshop Vision for Human-Computer Interaction (V4HCI)*, Jun. 2005, p. 79.

[45] V. Vezhnevets and A. Degtiareva, "Robust and accurate eye contour extraction," in *Proc. GraphiCon'03*, 2003, pp. 81–84.

[46] F. Coutinho and C. Morimoto, "Free head motion eye gaze tracking using a single camera and multiple light sources," in *Proc. Brazilian Symp. Computer Graphics and Image Processing (SIBGRAPI'06)*, Oct. 2006, pp. 171–178.

[47] R. Bailey, A. McNamara, N. Sudarsanam, and C. Grimm, "Subtle gaze direction," in *Proc. ACM SIGGRAPH 2007 posters*, 2007.

[48] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. New York: Cambridge Univ. Press, 1992.

[49] J. Orozco, F. X. Roca, and J. Gonzàlez, "Real-time gaze tracking with appearance-based models," *Mach. Vision Appl.*, vol. 7, pp. 1–12, Oct. 2009.

Michael J. Reale (S'10) received the B.S. degree in computer science from SUNY Oneonta, NY, in 2007 and the M.S. degree from the State University of New York at Binghamton in 2009. He is now pursuing the Ph.D. degree.

He works in the Graphics and Image Computing (GAIC) lab at the State University of New York at Binghamton. His research interests include eye tracking and gaze estimation, human–computer interaction interfaces, expression recognition, and computer graphics, as well as GPGPU programming for computer vision.

Shaun Canavan (S'07) received the B.S. degree in computer science and the M.C.I.S. degree in computer information systems from Youngstown State University, Youngstown, OH, in 2006 and 2008, respectively. He is currently pursuing the Ph.D. degree in computer science with the Graphics and Image Computing (GAIC) lab at the State University of New York at Binghamton.

His research interests include statistical shape analysis of 2-D and 3-D shapes, face recognition, and human–computer interaction.

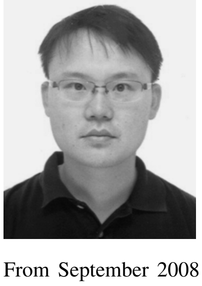
Lijun Yin (M'00–SM'10) received the M.Sc. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 1992 and the Ph.D. degree in computer science from the University of Alberta, Edmonton, AB, Canada, in 2000.

Before he joined the State University of New York at Binghamton in 2001, he was with Chyron Corp., Long Island, NY, and InfiMed Inc., Syracuse, NY, as a Computer Graphics and Image Scientist. He is currently an Associate Professor and the Director of the Graphics and Image Computing (GAIC) Laboratory in the Computer Science Department, State University of New York at Binghamton. He was a summer visiting faculty in the Air Force Research Lab at Rome, NY, in 2005. His research interests include visual information processing, computer vision, biometrics, face and gesture analysis, recognition, animation, and applications to human–computer interaction. He has over 90 publications in referred journals and conferences. His research has been supported by the National Science Foundation, the New York State Office of Science, Technology and Academic Research (NYSTAR), Air Force Research Lab, and the SUNY Upstate Medical Center.

Dr. Yin received the prestigious NYSTAR's James Watson Young Investigator Award in 2006. He is currently serving as an Editorial Board Member for the *Journal of Image and Vision Computing*.

Kaoning Hu received the B.S. degree and the M.S. degree in electronics engineering from Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2008, respectively. He is now pursuing the Ph.D. degree in computer science with the Graphics and Image Computing (GAIC) lab at the State University of New York at Binghamton.

His research interests are hand tracking, gesture recognition, human–computer interaction interfaces, and computer graphics.



Terry Hung received the B.S. and M.S. degrees in biomedical engineering from CYCU, Taiwan, and the M.S. degree in computer science at the State University of New York at Binghamton in 2008.

He worked in the Graphics and Image Computing (GAIC) Lab at the State University of New York at Binghamton. His research interests have included data mining, machine learning, image processing, face detection, and medical device design. He created the product Golf instant Replayer (Golf IR) in 2005, which was marketed by Sun Scientific Corp. From September 2008 to the present, he has been a Senior Control System Engineer in Corning Inc., Taichung City, Taiwan.