

“Look, Ma – No Hands!”

Hands-Free Cursor Control with Real-Time 3D Face Tracking

Kentaro Toyama
Microsoft Research
One Microsoft Way, Redmond, WA 98052
kentoy@microsoft.com

Abstract

This paper presents a real-time, non-invasive, vision system which performs hands-free cursor control by having users point their nose where they wish to place the cursor on a monitor screen. The vision system robustly tracks 3D face position and orientation in real time using a framework called Incremental Focus of Attention (IFA). IFA integrates tracking based on multiple cues (including color, intensity templates, and dark point features) which cooperate to track under adverse visual conditions. The pose recovered from tracking is then used to compute the intersection between the plane of the monitor screen and an imaginary ray extending forward from the user’s nose. Results show that naive users can position a cursor within a 1cm x 1cm square from a distance of 50cm from the monitor.

1 Introduction

Vision-based human-computer interfaces (VBI, for short) take a live video stream from a camera pointed at a user and interpret the user’s actions for the purposes of directing computation. This paper considers one form of VBI, where the user’s face is tracked in 3D as he points his nose at a point on the computer monitor (Figure 1). The tracked pose of a user’s face allows positioning of the cursor without the use of a mouse, joystick, or keyboard. (Note that button click operations will not be discussed, but that dwell, eye-blink or mouth movement detection, speech recognition, and other mechanisms are conceivable.)

One application of this technology is as a hands-free mouse substitute for disabled access. Users who have difficulty using a standard mouse could manipulate an on-screen cursor merely by moving their heads.

Another application of hands-free cursor control allows users to change the “focus window” in a window GUI without mouse movement. Imagine two GUI windows opened side-by-side on the desktop. Instead of having to laboriously switch the active window by moving and clicking on

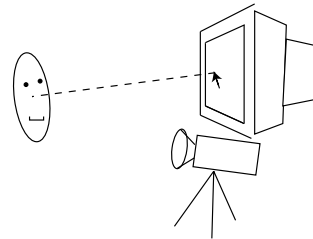


Figure 1. Cursor control with face tracking.

the mouse, the user could simply turn towards desired window, and keyboard input would flow into the appropriate document.

Finally, there are applications of hands-free cursor control for entertainment – painting programs, games, and so on.

Of course, these applications have inspired a host of hands-free mouse replacements. Hardware companies have developed headgear for tracking the 3D pose of a person’s head. In the accessibility community, several companies support products that perform head tracking or eye gaze tracking. These products are accurate and reliable, but all require either expensive dedicated hardware or structuring of the environment (special lighting, markings on the user’s face, etc.) to simplify the tracking process.

Ideally, head tracking would be performed cheaply and non-invasively, and computer vision techniques offer just such a possibility [1]. Using only a single desktop camera positioned as in a teleconferencing application, reasonable ambient illumination is sufficient for data gathering. And, as hardware to support live visual processing becomes more commonplace, computer vision becomes less and less expensive, both in monetary and computational cost.

In fact, recent advances in hardware have allowed vision researchers to develop real-time face tracking systems [3, 5, 6, 7, 9, 13, 14, 15]. (It should also be mentioned that a few researchers have attempted to track eye gaze using passive computer vision but with few dependable results [10, 16].) Although all such systems could be used for cursor control to some extent, most, having been developed for

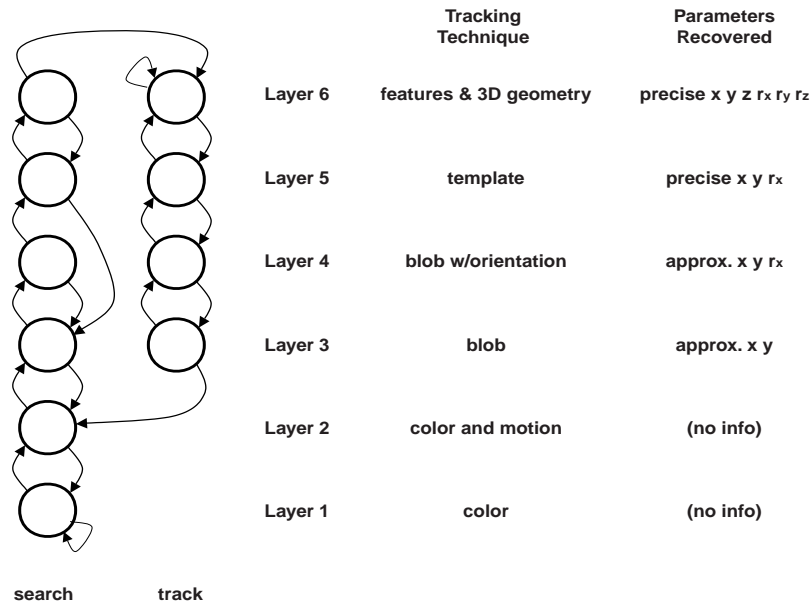


Figure 2. States for face tracking.

others tasks such as lipreading, are not well-suited for the cursor control task. Some do not compute full 3D, 6-degree-of-freedom pose; cursor control would therefore have to be based on head position rather than (position and) orientation. Others severely limit allowable translational motion. Still others become confused if multiple faces appear in the field of view. And lastly, many systems lack the robustness required for practical application – tracking is unable to handle or recover from situations such as occlusion or exit from the field of view.

This paper presents a cursor control system that uses face tracking based on Incremental Focus of Attention (IFA) [20]. The next section describes the face tracking subsystem, which is able to track the 3D pose of a single user’s face at 30Hz. It rapidly recovers from moments of tracking failure caused by visual disturbances and does so without being distracted by other faces in the field of view. Section 3 discusses the geometry of cursor control, based on recovered pose from face tracking. Finally, in Section 4, experiments illustrate the robustness of IFA face tracking and the precision with which cursor control can be achieved.

2 Face Tracking

In order for cursor control by face pose to be practical, face tracking must be robust. In particular, because users cannot be tracked perfectly *all* of the time (for example, when they leave the field of view), tracking systems must be *post-failure robust* and capable of recovering from any visual disturbance that disrupts normal tracking [19].

2.1 Incremental Focus of Attention

Reliable face tracking is accomplished by implementing a system based on Incremental Focus of Attention (IFA), an architecture for incorporating different tracking algorithms into a robust real-time tracking system [18, 20]. If designed for the face tracking task, IFA allows tracking of 3D pose when conditions are favorable (*i.e.*, all facial features are visible) and additionally performs rapid recovery from moments of tracking failure. A brief description of the IFA face tracking system follows (a more rigorous presentation of IFA appears elsewhere [20]).

IFA casts search heuristics and tracking algorithms as *layers* (Figure 2) which perform search space reduction, where the search space is the space of all poses assumable by the face, and live images are used as hints to prune the search in several steps. For example, one layer of the system uses color information: By examining the image and identifying those regions of the image which include skin-colored pixels, the system can narrow its search to those portions of the space of facial poses where projections of the face onto the image are consistent with observed skin-colored pixels. Successful ascension of the layers incrementally shrinks the search space until only a small, compact set of viable poses remain. This set can be considered the final estimate and its margin of error.

Associated with the layer hierarchy is a finite state machine that controls layer execution. States cause corresponding layers to execute at any given moment, and layers cause transitions between states based on their outcome. Certain layers, for example, act as confirmation gateways, where if a reduced search space does not appear to contain

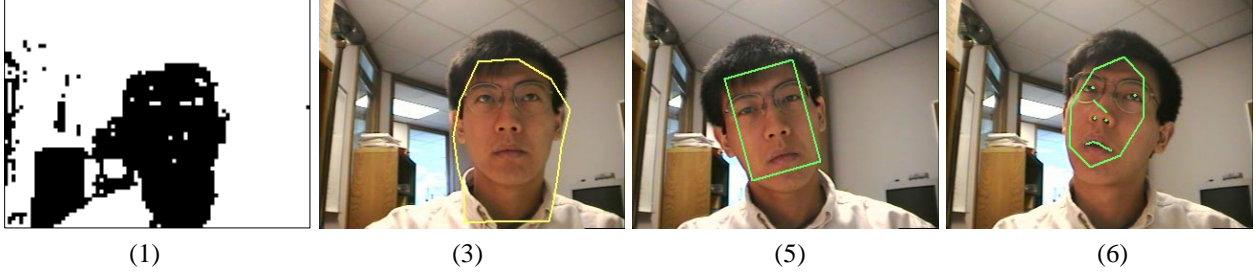


Figure 3. Face tracking in different layers.

the intended target, the system is directed to lower layers where a larger search space is re-examined.

2.2 IFA for Faces

Six layers compose the current face tracking system (see Figure 3 for four of the layers in action). For efficiency, actual IFA search spaces are not maintained explicitly. In fact, Layers 1 and 2 only maintain a list of image pixels which correspond to color or motion, and Layers 3 through 6 only return a single value for each state parameter they recover. These outputs can be interpreted as implicit representations of search spaces. For example, when Layer 3 returns values x' and y' as approximate x and y coordinates of the face, the output can be thought to correspond to a subset of the entire search space for which x and y are constrained by $\sqrt{((x' - x)^2 + (y' - y)^2)} \leq \epsilon$ and the remaining parameters are completely unconstrained (ϵ is Layer 3's margin of error). In the following, all variables, k , indicate a positive constant set empirically.

Layer 1 selects regions of the search space corresponding to pixels exhibiting skin color or motion. Skin color is defined to be those RGB values where

$$\begin{aligned} k_{rg}^- &< r/g < k_{rg}^+ \\ k_{rb}^- &< r/b < k_{rb}^+. \end{aligned}$$

This color model describes a pyramidal wedge in RGB space which accepts skin colors of all races under very approximate white light. Although adaptation of the color model to specific individuals is a possibility [13], this layer is meant as an initial attention-focusing scheme only; a liberal color model is preferred to one which might produce false negatives under varying illumination.

Layer 2 is a color- and motion-based selector that is biased toward regions nearest the last observed position of the target. The search spirals outward and selects state sets consistent with those on those pixels which exhibit both skin color (as in Layer 1), and large motion,

$$\frac{dI(\mathbf{x})}{dt} > k_m,$$

where $I(\mathbf{x})$ represents the image intensity at pixel \mathbf{x} .

Layer 3 uses “radial spanning” to find the approximate size and shape of a single cluster of skin-colored pixels. Radial spanning is a fast cluster detection algorithm which extends initializes pixel probes at the center of the last known position of the face and extends them radially outward until they find non-skin-colored pixels [17]. Probes are affected by forces as follows:

$$\begin{aligned} F_i &= F_i^{out} + F_i^{in} + F_i^{int}, \text{ where} \\ F_i^{out} &= k_{out}, \\ F_i^{in} &= -k_{in}, \text{ if pixel at } \mathbf{x}_i \text{ is not skin color,} \\ F_i^{int} &= k_{int} * \frac{(2\mathbf{x}_i - \mathbf{x}_{p(i)} - \mathbf{x}_{s(i)}) \cdot \mathbf{v}_i}{|2\mathbf{x}_i - \mathbf{x}_{p(i)} - \mathbf{x}_{s(i)}|}. \end{aligned}$$

Here, i indexes a predetermined number of probes (16 are used for face tracking), \mathbf{x}_i is Probe i 's pixel location, \mathbf{v}_i is Probe i 's expansion direction, and \mathbf{p} and \mathbf{s} return the predecessor and successor indices of i . The algorithm is similar to “draping” [22] and to balloons [4], but it is more efficient, and it avoids spurious edges when tracking faces by constraining shape to star-shaped polygons. Approximate position in the plane parallel to the camera image is tracked.

Layer 4 is the same as Layer 3, with an additional computation of the principal axis of the cluster; approximate in-plane position and orientation are tracked.

Layer 5 performs linearized sum-of-squared-differences (SSD) tracking [7], which iteratively matches live images to a stored template acquired during a manual initialization step that can be performed offline:

$$\mathbf{X}_{t+1} = \mathbf{X}_t - (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T (\mathbf{I}(\mathbf{X}_t, t+1) - \mathbf{I}(\mathbf{0}, t_0)),$$

where \mathbf{X} is a vector $[x \ y \ \theta]^T$, $\mathbf{I}(\mathbf{X}, t)$ is a vector representing the image at time t translated and rotated according to \mathbf{X} , \mathbf{J} is the Jacobian of \mathbf{I} with respect to \mathbf{X} , and $\mathbf{I}(\mathbf{0}, t_0)$ is the original template. Fine position and in-plane orientation are tracked. This layer works only for approximately frontal poses, but in those poses, it ensures that a particular individual, and not just any skin-colored blob, is tracked by accepting only those matches which have a low SSD residual value.

Layer 6 tracks 5 point features of the face including the eyes, the nostrils, and the mouth. The eyes and nostrils are tracked using small search windows with first moment computations on the inverse of pixel intensity to update feature and window positions (similar in spirit to the feature trackers used in [6]). The upper lip is tracked using a snake-like contour tracking algorithm attracted to intensity valleys in the image [11]. These features are then used to determine the 3D transformation from a face reference frame to the camera reference frame by finding a least-squares fit between an approximate geometric model of facial features and the tracked features under weak perspective:

$$\mathbf{T} = (\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T \mathbf{Q},$$

where \mathbf{T} is the recovered transformation matrix, \mathbf{P} is a 3x5 matrix of the concatenated image feature positions with the “z” parameter held constant, and \mathbf{Q} is the 4x5 concatenation of the five model points in homogeneous coordinates. Simple algebra on \mathbf{T} , with the assumption that head orientation is in the hemisphere facing the camera, allows full recovery of 6-DOF pose.

3 Cursor Control

Once 6-DOF face pose is recovered, positioning of the cursor is conceptually simple. One can extend a ray from the user’s nose that is normal to the plane of the face and find the intersection of the ray with the plane of the monitor screen (see Figure 1).

3.1 Known Monitor Plane

If we know the monitor plane in the camera frame, this problem is trivial. In particular, let the plane of the monitor screen be given by

$$\mathbf{a}^T \mathbf{x} = [a \ b \ c \ 1] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0, \quad (1)$$

where \mathbf{a} represents the plane parameters and \mathbf{x} is in homogeneous camera coordinates. Let $\mathbf{n}^f = [u_n \ v_n \ w_n \ 1]^T$ be the coordinate of the nose in the face reference frame, $\mathbf{d}^f = [u_d \ v_d \ w_d \ 0]^T$ be the normalized direction of the line perpendicular to the facial plane, and \mathbf{T}_f^c be the transformation from the face frame to the camera frame determined by face tracking. Then the ray projecting from the nose that is normal to the face (this ray will henceforth called the *face normal*) can be described, in camera coordinates, as

$$\begin{aligned} \mathbf{T}_f^c (\mathbf{n}^f + \lambda \mathbf{d}^f) &= \mathbf{T}_f^c \mathbf{n}^f + \lambda \mathbf{T}_f^c \mathbf{d}^f \\ &= \mathbf{n}^c + \lambda \mathbf{d}^c, \end{aligned} \quad (2)$$

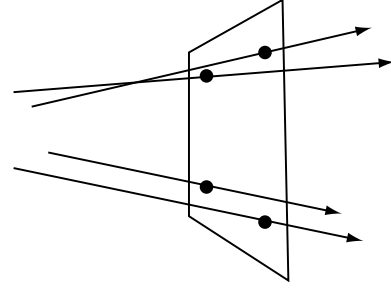


Figure 4. A plane with 4 rays passing through 4 specified points.

where $\lambda \geq 0$, and \mathbf{n}^c and \mathbf{d}^c are in camera coordinates.

We then find the λ which satisfies Equation 1:

$$\begin{aligned} \mathbf{a}^T (\mathbf{n}^c + \lambda \mathbf{d}^c) &= 0 \\ \lambda &= -\frac{\mathbf{a}^T \mathbf{n}^c}{\mathbf{a}^T \mathbf{d}^c}. \end{aligned} \quad (3)$$

Knowing λ , we know the intersection of the face normal with the monitor, and we can simply place the cursor at that position. Additional filtering of the output also decreases jitter due to noise from face tracking (see Section 3.3).

3.2 Unknown Monitor Plane

Without a known monitor-camera relationship, we can either perform calibration to determine the monitor plane, or we can make approximations based on certain assumptions about the camera-monitor relationship.

3.2.1 Calibration

For proper calibration, some user input is required. Users must be prompted to point their noses in the direction of specified points on the monitor screen. Supposing we use n such points, then the problem of finding the monitor plane can be reduced to the following geometry question: Given n 2D points, \mathbf{S}_i , on a plane and n rays, \mathbf{R}_i (in 3D), what Euclidean transformation of the plane allows each \mathbf{R}_i to pass through the transformed point \mathbf{S}_i on the plane (see Figure 4)?

If the rays \mathbf{R}_i happen to lie in a pencil of lines through a point, then this problem reduces to the standard camera calibration problem with coplanar data points. Methods for solving this problem for four or more rays are well-known [8, 21]. Unfortunately, there is no guarantee that users will accommodate the system so conveniently, as they may translate during the calibration process. Instead, we will consider a standard least-squares formulation of the problem.

Assume we are given n rays, \mathbf{R}_i parametrized by λ_i ,

$$\mathbf{R}_i(\lambda(i)) = \mathbf{n}_i + \lambda_i \mathbf{d}_i.$$

Then, the task is to estimate the plane parameters \mathbf{a}^* such that the sum of squared differences between the distances between ray intersections with the plane and corresponding distances between points P_i is minimized:

$$\arg \min_{\{\mathbf{a}\}} \sum_{i=1}^n \sum_{j=i+1}^n (\|\mathbf{R}_i(\lambda_i) - \mathbf{R}_j(\lambda_j)\| - \|\mathbf{S}_i - \mathbf{S}_j\|)^2,$$

where the λ values are dependent on \mathbf{a} and are computed as in Equation 3. This problem can be solved iteratively by guessing an initial plane and using steepest descent techniques on the plane parameters.

The method described above does not work well in practice for several reasons: First, ray estimation is noisy and this form of plane-based calibration is known to be ill-conditioned [21]. So, small errors in pose estimates from face tracking translate to significant errors in calibration. As a result, n must be large for reliable calibration, but larger n implies greater setup time for the user. Even if pose estimation were accurate, however, there remains a greater source of error: human estimates of where one’s nose is pointing are wildly inconsistent. An informal test on four computer users asked to “point” at a marked pixel on the monitor from a distance of 50cm showed that over the course of 10 trials, even the same person would point at pixels as far as 7cm from the target after a “calibration” step (briefly showing where they were pointing) and do so with great variation (standard deviations of up to 3.5cm).

3.2.2 Approximation

Aside from the difficulty in determining accurate monitor plane estimates, there are several reasons why good calibration is unnecessary – even undesired – for the cursor control task. For example, the monitor plane coefficients can be approximated using knowledge of the environment. Desktop cameras are likely to be located close to the monitor with image planes roughly parallel to the monitor (significant deviation would not allow tracking of facial features); a reliable approximate calibration is already known. Then, too, for tasks such as cursor control, users are provided immediate visual feedback of the control process via the graphic of the cursor itself, allowing them to adaptively adjust their own pose as appropriate. This is an instance of one of the standing tenets of UI design, which suggests that appropriate closed-loop feedback alone goes a long way toward intuitive interfaces [12]. Another UI notion is that preparation prior to using the interface should be minimal; an interactive calibration step should be avoided altogether because of its inconvenience.

These facts suggest that a good approximate method for estimating direction of the face normal is sufficient for the cursor control task. The simplest such approximation is to make the assumption that the camera image plane and the monitor plane are coincident, with identical origins. That

is, we let $a = b = 0$ and $c = 1$, in Equation 1. Then, the intersection point of the face normal and the approximated monitor plane can be immediately computed as described in Section 3.1.

3.3 Filtering

Because pose estimation is noisy as a result of the additive effects of noise from the underlying feature tracking algorithms, we perform adaptive filtering to smooth the motions of the cursor and make it easier to direct. More specifically, Kalman filters [2] are applied separately on each coordinate of the cursor position (*i.e.*, a block-diagonal covariance matrix). The motion models contain state parameters for displacement and velocity; the noise model parameters were chosen empirically.

4 Results

All experiments were run on a single-processor, 266MHz Pentium II PC equipped with a Sony single-CCD color camera and Matrox Meteor framegrabber.

Constructed as described in Section 2, the face tracking system allows for highly robust, real-time tracking. Under normal visual conditions, tracking proceeds at the top layer, with recovery of complete 6-DOF pose at 30Hz (see Movie 1¹).

The system is also robust to a variety of visual disturbances: When partial occlusions of the face occur, for example, the system falls to Layer (5), where SSD tracking tracks 2D position and orientation. Turning the face over 30 degrees to the left or right causes the system to fall to Layers (3) and (4) where only approximate position is tracked. And, events such as full occlusion or exit from the field of view will cause the system to descend all the way to Layers (1) and (2). In all such cases, cursor movement is halted. If the disturbance is temporary, then when an approximately frontal view of the target face is presented without occlusion, tracking and cursor positioning will recover within a few frames (between 33 and 200 milliseconds depending on the type of disturbance and number of skin-colored clusters in the background) (see Movie 2). Fortunately, in those instances when a disturbance is long term (*e.g.*, the user being away from the computer or turning to the side to speak to someone), the system does not need to perform cursor control.

Finally, we performed an initial series of experiments on four subjects, of which one had extensive familiarity with the system. The other three were exposed to the system for only several minutes before undertaking the test sequence. At a distance of 50cm from the monitor and approximately

¹All movie files are available in MPEG format from www.research.microsoft.com/toyama/research.htm.

40cm from the camera, which was placed below and in front of the monitor, all users were able to hold the cursor for one second within a 1cm x 1cm square with filtering (Section 3.3) and in an 5cm x 5cm square without. The effective angular noise with filtering is equivalent to ± 1.15 degrees. Greater accuracy could be achieved by modifying noise parameters in the filter, but this would come at the cost of a noticeable lag in cursor response.

5 Discussion

Robust, real-time 3D tracking of facial position and pose from a desktop camera can be applied to accurately position a cursor on a monitor. Even with only approximate camera-monitor calibration, this system provides an intuitive, user-friendly interface for controlling cursor movement to within 1cm on a monitor using head motion alone.

The system still requires additional work prior to practical implementation. First, dependence on preselected facial features such as the eyes and the lips must be eliminated, since they are not visible or easily tracked on all faces. Similarly, moustaches and thick beards cause difficulty for lip tracking. All of these features also suffer from the fact that they move independently of the entire face, causing unwanted jitter in pose tracking. Methods which automatically detect and track stable points on the face should therefore be used to perform the top-level feature-based tracking [9, 23]. Such improvements will also contribute to better resolution of cursor positioning making more precise control possible.

Another difficulty comes from the system's reliance on frontal faces for recovery. This drawback results in moments without cursor control, as the system tries in vain to recover track of the face. Clearly, a mechanism for tracking and reinitialization of non-frontal faces is necessary.

Finally, if so desired, some method of emulating mouse clicks through facial movement (perhaps through nod or blink detection) would complete the current system's ability to replace the standard desktop mouse through tracking of face movement alone.

References

- [1] P. Ballard and G. C. Stockman. Controlling a computer via facial aspect. *IEEE Trans. Sys. Man and Cybernetics*, 25(4):669–677, 1995.
- [2] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.
- [3] S. Birchfield and C. Tomasi. Elliptical head tracking using intensity gradients and color histograms. In *Proc. Computer Vision and Patt. Recog.*, 1998.
- [4] L. D. Cohen. On active contour models and balloons. *CVGIP: Image Understanding*, 53(2):211–218, March 1991.
- [5] A. Colmenarez, R. Lopez, and T. S. Huang. 3D head pose computation from 2D images: Templates versus features. In *Proc. Int'l Conf. on Image Proc.*, 1995.
- [6] A. Gee and R. Cipolla. Fast visual tracking by temporal consensus. *Image and Vision Computing*, 14(2):105–114, March 1996.
- [7] G. Hager and P. Belhumeur. Occlusion insensitive tracking of image regions with changes in geometry and illumination. Technical Report DCS-TR-1122, Yale University, 1996.
- [8] R. M. Haralick. Determining camera parameters from the perspective projection of a rectangle. *Patt. Recog.*, 22:223–230, 1989.
- [9] T. S. Jebara and A. Pentland. Parametrized structure from motion for 3D adaptive feedback tracking of faces. In *Proc. Computer Vision and Patt. Recog.*, 1997.
- [10] M. J. Jones and T. Poggio. Multidimensional morphable models. In *Proc. Int'l Conf. on Computer Vision*, pages 683–688, 1998.
- [11] Y. Moses, D. Reynard, and A. Blake. Robust real time tracking and classification of facial expressions. In *Proc. Int'l Conf. on Computer Vision*, pages 296–301, 1995.
- [12] D. A. Norman. Cognitive engineering. In D. A. Norman and S. W. Draper, editors, *User Centered System Design*, pages 31–61. Erlbaum, 1986.
- [13] N. Oliver, A. Pentland, and F. Berard. LAFTER: Lips and face real-time tracker. In *Proc. Computer Vision and Patt. Recog.*, pages 123–130, 1997.
- [14] E. Petajan and H. P. Graf. Robust face feature analysis for automatic speechreading and character animation. In *Proc. Int'l Conf. on Autom. Face and Gesture Recog.*, pages 357–362, 1996.
- [15] G. Potamianos, E. Cosatto, H. P. Graf, and D. B. Roe. Speaker independent audio-visual database for bimodal ASR. In *Proc. Euro. Tutor. and Res. Wkshp. on AV Speech Proc.*, 1997.
- [16] R. Stiefelhagen, J. Yang, and A. Waibel. Tracking eyes and monitoring eye gaze. In *Proc. Wkshp on Perceptual UI*, Banff, Canada, 1997.
- [17] K. Toyama. Radial spanning for fast blob detection. In *Proc. Int'l. Conf. on Comp. Vision, Patt. Recog., and Image Proc.*, 1998.
- [18] K. Toyama and G. Hager. Incremental Focus of Attention for robust visual tracking. In *Proc. CVPR*, pages 189–195, 1996.
- [19] K. Toyama and G. Hager. If at first you don't succeed... In *Proc. AAAI*, pages 3–9, Providence, RI, 1997.
- [20] K. Toyama and G. Hager. Incremental focus of attention for robust vision-based tracking. *Int'l J. of Computer Vision*, 1999.
- [21] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Trans. Robot. and Autom.*, 3(4):323–344, 1987.
- [22] M. Turk. Visual interaction with lifelike characters. In *Proc. Automatic Face and Gesture Recognition*, 1996.
- [23] C. S. Wiles, A. Maki, N. Matsuda, and M. Watanabe. Hyper-Patches for 3D model acquisition and tracking. In *Proc. Computer Vision and Patt. Recog.*, pages 1074–1080, 1997.