Fast Head Tilt Detection for Human-Computer Interaction

Benjamin N. Waber, John J. Magee, and Margrit Betke

Computer Science Dept., Boston University
{bwabes,mageejo,betke}@cs.bu.edu

Abstract. Accurate head tilt detection has a large potential to aid people with disabilities in the use of human-computer interfaces and provide universal access to communication software. We show how it can be utilized to tab through links on a web page or control a video game with head motions. It may also be useful as a correction method for currently available video-based assistive technology that requires upright facial poses. Few of the existing computer vision methods that detect head rotations in and out of the image plane with reasonable accuracy can operate within the context of a real-time communication interface because the computational expense that they incur is too great. Our method uses a variety of metrics to obtain a robust head tilt estimate without incurring the computational cost of previous methods. Our system runs in real time on a computer with a 2.53 GHz processor, 256 MB of RAM and an inexpensive webcam, using only 55% of the processor cycles.

1 Introduction

Many existing face analysis systems require an upright face as input to function correctly and often assume that no head tilt occurs (e.g., [1–5]). A head tilt detection system could serve as a pre-processor to these systems by rotating the input image by the estimated head tilt angle and thus facilitate interaction in circumstances when the head is not upright. This would be particularly important for people with severe motion impairments, e.g., due to cerebral palsy or multiple sclerosis, who often have difficulties holding their heads straight. They could then use video-based assistive technology that assumes an upright face, such as EyeKeys [4], and thus gain access to communication software. Head tilt detection can also enhance human-computer interaction by providing an additional mechanism to select commands. Left and right head tilts could be used to rotate a 3D model, control a video game, tab through a web page, or select letters in a scan-based text-entry program.

Among previous face detection algorithms, CAMSHIFT [6] is particularly geared towards real-time human-computer interaction. It requires a color model of the face that it will be tracking, and it finds the center, elongation, and tilt of the face by determining the likelihood that pixels in the input image belong to the face based on a comparison of their color values with the prior face color model. Other face tracking methods also use color and motion information [7, 8].

Face detection systems that require extensive training use neural networks [9] or AdaBoost [10–12]. These systems are very effective in detecting upright faces, but give only coarse estimates of head tilt and put severe computational and memory strains on a computer that, if used for human-computer interaction, must at the same time provide computational resources for the application program.

Motion of the head, the foreground object, in a video sequence causes pixels to transition from background to foreground or foreground to background, depending on the direction of motion. We observed that an image representation of these transitions contains distinctly grouped pixels created by the motion of the head, in particular by the change of its occluding boundary. Our contribution is a method of estimating head tilt from the size, relative location, and orientation of these groups. A second contribution is our method to estimate head tilt from the motion of the center of the face. Assuming head rotation is parallel to the image plane, a circle can be fit to the sequence of face centroids as the head rotates. This second estimate of head tilt is combined via a weighting function with our estimate based on motion of the occluding boundary to yield an estimate that is comparable to the most powerful methods to date (e.g., [9, 12]).

Not only is our method a novel technique, but its real-time performance on common desktop computers with inexpensive cameras makes it immediately applicable to human-computer interaction, unlike some previous approaches (e.g., [9, 12]). Our method can be integrated as part of a larger interaction system; here we show its performance as a stand-alone system.

2 Head Tilt Estimation Algorithm

Our head tilt estimation algorithm has four steps:

1. Foreground and background segmentation,
2. Analysis of the motion of the occluding boundary of the face, which provides head tilt estimate θ_b,
3. Analysis of the motion of the center of the face to compute angle estimate θ_c,
4. Analysis of the confidence factor w that determines the weighting of the two estimates θ_b and θ_c in computing the final head tilt estimate $\theta = (1-w)\,\theta_b + w\,\theta_c$.

We assume that our algorithm has access to a foreground-detection method. Ideally, the segmented foreground image I_f contains the user's head alone, but our method can also handle foreground segmentations that include the user's neck and shoulders and regions in the background (e.g., it works for the poor foreground segmentation in Fig. 1C). In our experiments, we used a simple foreground estimation method that subtracted the current frame (Fig. 1B) from the initial frame without the user (Fig. 1A). Our method also relies on the mild assumptions that movements in the background are not correlated with the user's head motion and affect a smaller number of pixels in the video than the user's head motion.

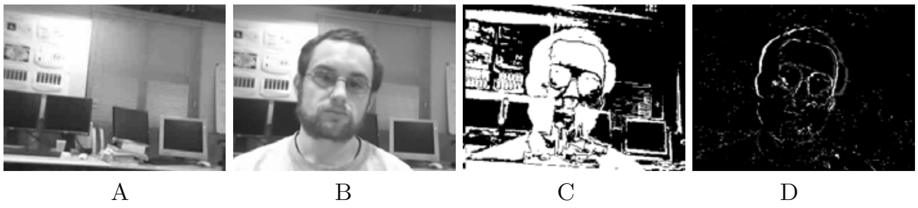

Fig. 1. Images used by our system. A) Background image. B) Example input frame. C) Foreground image $I_{f,t}$ computed from differencing images in A and B (here noisy due to camera shift). D) The signed difference $I_{f,t} - I_{f,t-1}$. The gray pixels represent $I_{b\to f}$, the white pixels represent $I_{f\to b}$

Foreground pixels in I_f are set to 1, while background pixels are set to 0. A representation of motion is obtained by subtracting the foreground image of the previous frame from the foreground image of the current frame, taking care to preserve the sign of the result (Fig. 1D). The image is then separated into two frames: binary image $I_{b\to f}$ to represent pixels that changed from belonging to the background at time $t-1$ to belonging to the foreground at time t, and $I_{f\to b}$ to represent pixels in the foreground at time $t-1$ and in the background at time t.

2.1 Angle Estimation Based on Occluding Boundary of Face

We now explain how head tilt angles are estimated in the binary motion images $I_{b\to f}$ and $I_{f\to b}$. Three filtering steps are performed on both $I_{b\to f}$ and $I_{f\to b}$ to find connected components that represent the motion of the occluding boundary of the face well, in particular, at the sides of the face. First, connected components too large or too small to represent the face are removed (Fig 2A). Then components too far from the centroids of the respective components of $I_{b\to f,t-1}$ and $I_{f\to b,t-1}$ are removed (Fig. 2B). Furthermore, components whose orientation is too far from the estimated orientation of the face in the previous frame are removed (Fig. 2C). To perform this last filtering step, each remaining component is processed as follows:

Pixels with high curvature at the top and bottom of the filtered components are removed since they do not represent the motion of the sides of the face well (Fig. 2D). For each connected component i, its lowest pixel $\mathbf{x}_{i,o}$ is used as the origin of a local polar coordinate space. We ignore additional portions of the connected component above the origin with lower, but still relatively high curvature within this connected component (orange in Fig. 2D) to allow for more accurate angle estimation. The angle $\alpha_{i,j}$ representing pixel $\mathbf{x}_{i,j}$ is thus the angle between the line $\mathbf{x}_{i,j} - \mathbf{x}_{i,o}$ and the horizontal axis. If the face is tilting left, which appears as a right rotation in the image, the pixel with the largest angle $\theta_{L,i}$ is roughly equivalent to the head tilt angle estimated in the previous frame, and the pixel with the smallest angle $\theta_{R,i}$ can be used to describe the head tilt in the current frame:

$$\theta_{L,i} = \max \{\, \alpha_{i,j} \mid \forall\, j \text{ in component } i \,\}, \tag{1}$$

$$\theta_{R,i} = \min \{ \alpha_{i,j} \mid \forall j \text{ in component } i \}. \tag{2}$$

Angle $\theta_{R,i}$ is a particularly good approximation of the head tilt if component i represents the motion of the right side of the face (shown left) in $I_{f\to b}$ (Fig. 2D) or the motion of the left side of the face (shown right) in $I_{b\to f}$. Vice versa, if the face is tilting right, angle $\theta_{L,i}$ is a particularly good approximation of the head tilt if component i represents the motion of the right side of the face (shown left) in $I_{b\to f}$ or the motion of the left side of the face (shown right) in $I_{f\to b}$.

If angles $\theta_{L,i}$ or $\theta_{R,i}$ are not within a small γ_T of the head tilt estimate from the previous frame, then component i is discarded. The threshold γ_T was experimentally set to 10 degrees and found to work quite well.

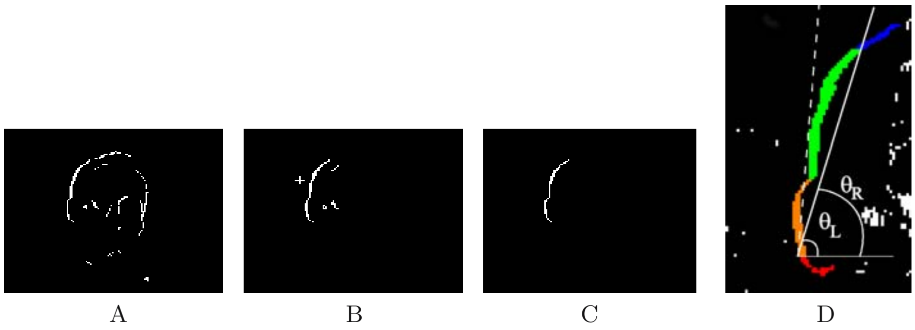

Fig. 2. Results of filtering steps on image $I_{f\to b}$ in Fig. 1D. A) Size filter. B) Filter on distance from centroid (cross) in previous frame. C) Filter on on difference to estimated angle in previous frame. Here, but not generally, only one component remained. D) Zoomed-in sub-image of $I_{f\to b}$ containing the filtered component in C (colored pixels) and the discarded components (white). The pixels at the top and bottom of the filtered component (blue, red, and orange) are disregarded in estimating the orientation of the occluding boundary. Angle $\theta_{L,i}$ is approximately the head tilt in the previous frame (dashed line) and angle $\theta_{R,i}$ the head tilt in the current frame (solid line)

At this point in the process, our method has eliminated all components that do not represent the motion of the sides of the face well and created a list of angles $\theta_{L,i}$ and $\theta_{R,i}$ for both $I_{b\to f}$ and $I_{f\to b}$. Our algorithm compares these two lists by only considering those pairs of angles $\theta_{b\to f}, \theta_{f\to b}$ that are within a threshold τ of each other. Threshold τ was experimentally set to 20 degrees. If there is more than one pair remaining, pairs are eliminated if their size difference is too large. In particular, a component that is less than 2/3 the size of its paired component can be safely eliminated. If, however, no components meet this criterion and there is only a single candidate component remaining in either $I_{b\to f}$ or $I_{f\to b}$, then no pairs are eliminated in this way. This deals with cases when components of motion that should be considered are broken into smaller pieces as a result of poor foreground estimation. If there are still multiple pairs remaining, the pair $\theta_{b\to f}, \theta_{f\to b}$ that has the largest combined area is chosen.

The estimate of head tilt based on the analysis of the occluding boundary of the sides of the face is then computed by the average

$$\theta_b = \frac{\theta_{b\to f} + \theta_{f\to b}}{2}. \tag{3}$$

We also define a measure of confidence p_t that $\theta_{b,t}$ represents head tilt in the current frame t:

$$p_t = 1 - \frac{|\theta_{b\to f} - \theta_{f\to b}|}{\tau}. \tag{4}$$

The confidence is high if the angle estimates based on binary motion images $I_{b\to f}$ and $I_{f\to b}$ are similar and low otherwise.

2.2 Angle Estimation Based on Circular Face Motion

The angle estimate θ_b can be further refined by comparing the location $\mathbf{c}_{t,\text{meas}}$ of the center of the face in current frame t, measured in foreground image I_f, with its predicted location $\mathbf{c}_{t,\text{circle}}$. The prediction is based on a motion model that assumes that the center of the face follows a circular arc in the video as the face tilts sideways, and the center of rotation is a point on the neck (Fig. 4A). A pixel $\mathbf{x}_0$ is used as the center of rotation. It has the same x-coordinate as the center of the face $\mathbf{c}_{1,\text{meas}}$ when the face is in an upright position. The initial radius of the circle is the distance $\| \mathbf{c}_{1,\text{meas}} - \mathbf{x}_0 \|$ between the initial center of the face and its projection (Fig. 4A). The location of $\mathbf{c}_{t,\text{circle}}$ on the circular arc is determined by the intersection of the line between $\mathbf{x}_0$ and $\mathbf{c}_{t,\text{meas}}$ with the arc. If a large distance $\| \mathbf{c}_{t,\text{circle}} - \mathbf{c}_{t,\text{meas}} \|$ between predicted and measured face center occurs in subsequent frames, the radius is updated by the distance $\| \mathbf{x}_0 - \mathbf{c}_{t,\text{meas}} \|$.

The estimate of head tilt based on the average of the observed and predicted positions of the face center is

$$\theta_{c,t} = \frac{\| \mathbf{c}_{t,\text{meas}} - \mathbf{c}_{t,\text{circle}} \|}{2}. \tag{5}$$

As a measure of confidence in the angle estimate, we use the ratio e_t of distance between the observed and predicted center positions to the maximum possible distance T_t that the two points could be apart, i.e.,

$$e_t = 1 - \frac{\| \mathbf{c}_{t,\text{meas}} - \mathbf{c}_{t,\text{circle}} \|}{T_t}, \tag{6}$$

where T_t is the distance to the farthest corner of the image from $\mathbf{c}_{t,\text{circle}}$.

2.3 Weighted Angle Estimation

The head tilt estimate θ_b, computed by analyzing the occluding boundary of the face, and the estimate θ_c, computed by analyzing the circular motion of the face, can be combined to compute the weighted angle estimate

$$\theta_t = (1 - w_t)\, \theta_{b,t} + w_t\, \theta_{c,t}, \tag{7}$$

where the weighting factor

$$w_t = \frac{e_t^2}{e_t^2 + p_t^2} \tag{8}$$

is computed from e_t and p_t, which represent the respective confidences in estimates θ_b and θ_c. To prevent one measure from being the sole source of our final angle estimate, we set a lower bound of 0.2 for the weight of $\theta_{b,t}$ and 0.1 for the weight of $\theta_{c,t}$.

A summary of the head tilt estimation algorithm is given in Figure 3.

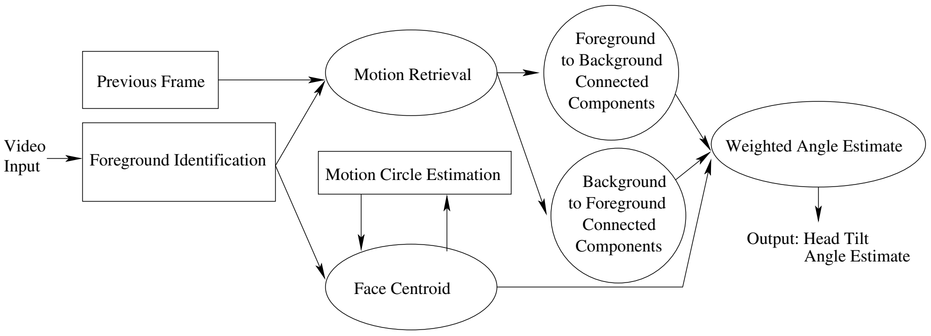
Previous Frame
Motion Retrieval
Foreground to Background Connected Components
Video Input
Foreground Identification
Motion Circle Estimation
Weighted Angle Estimate
Background to Foreground Connected Components
Output: Head Tilt Angle Estimate
Face Centroid

Fig. 3. Flowchart of head tilt estimation method

3 Human-Computer Interaction Experiments

To evaluate our method we performed three experiments. In experiment 1, we compared our system's head tilt estimation to that of two volunteers, who labeled each frame of five 320-frame video segments with their estimate of head tilt angle (Fig. 4B). Each of these video segments contained 5–7 separate head tilts with motion varying from slow and smooth to quick and erratic for a total of 31 distinct head tilts across 1320 images. In two of the video sequences, people were walking through the background. To verify that all the steps in our algorithm are necessary, we also tested our method in three other configurations: with the weighting component removed, with the centroid angle estimation removed, and with only centroid angle estimation.

In the second experiment, we tested users' ability to play the BlockEscape game [4] using our interface. We tested 5 users who each played 4 games. The game BlockEscape was developed as a tool to test the performance of human-computer interfaces. In the BlockEscape game, the screen contains a number of horizontal walls that are separated vertically by a fixed distance (Fig. 4C). Each wall has one or more holes of various widths. At the beginning of the game, a block appears on top of the screen. The goal of the game is for the user to lead the block through the holes in the walls until it reaches the bottom of the screen. The game was implemented so that the user only needs to initiate the block's horizontal movement by issuing "move block left" or "move block right" commands. The block continues to move in the initiated direction until the user issues a command for movement in the opposite direction. The block's vertical motion is automatic – when a hole is reached, the block "falls through" to the next wall or bottom of the screen. When a wall reaches the user's block, it pushes the block upwards. The user wins if he or she leads the block through the holes in the walls to the bottom of the screen and loses if a wall pushes the block up to the top of the screen. During playing, usage statistics, in particular, the departure of the user-controlled block from an optimal path, were computed based on the positions of the block, walls, and holes.

Our last experiment tested users' ability to navigate web pages with our head tilt estimation method. Test subjects were repeatedly shown web pages consisting of three links and directed to select a particular sequence of links (on average, the 2nd link on the page). A left head tilt directed the web browser to highlight the next link, while a right head tilt directed the browser to follow the currently selected link. The same sequence of links was used for all participants. We recorded the number of links each participant successfully followed before following an incorrect link.

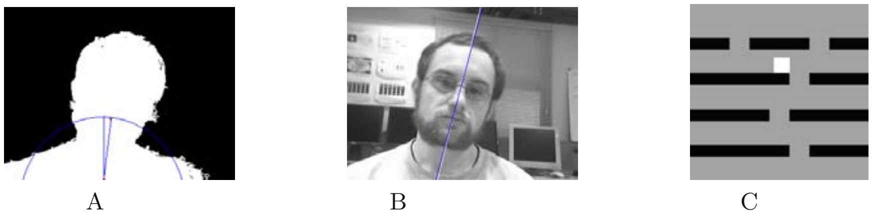

Fig. 4. A) Centroid angle detection binary image showing the centroid circle, circle center, and the original and current frame centroid positions. B) Experimental system output. Our system's head tilt estimate is represented by the white line, while the blue line represents the average of the two volunteers' estimates. C) Screenshot of the game BlockEscape. As the white block falls towards the bottom of the screen, the player navigates it through the holes in the black walls, which move upwards, by initiating "move block left" or "move block right" commands

4 Results of Human-Computer Interaction Experiments

Our system runs in real time on a computer with a 2.53 GHz processor, 256 MB of RAM and in these experiments used less than 55% of the processor cycles. The comparison of our system's head tilt estimations to those made by two volunteers is given in Table 1. A screenshot of the system's estimate and that of the volunteers shown in Fig. 4B. If we consider human observation to be ground truth, our system has exhibited a good performance. Each component of our method, angle estimation based on the analysis of the motion of the occluding boundary of the face, analysis of the moving center of the face, and the weighting scheme, is vital to the success of our interface. It is the combination of these components that yields a valuable HCI tool, and removing one of them yields a decline in performance (Table 1).

Table 1. Accuracy-Evaluation Experiment: Angle differences between four versions of our method and human observation, using 2 volunteers and five 320-frame video segments. The estimates of degrees were rounded to whole numbers because precision beyond this level in human observations is unlikely

	Weighted Estimate θ		**Equally Weighted Estimate** $\theta_b + \theta_c$		**Boundary-based Estimate** θ_b		**Center-based Estimate** θ_c	
Subject	Median Ang. Diff.	Std. Dev.	Median Ang. Diff.	Std. Dev.	Median Ang. Diff.	Std. Dev.	Median Ang. Diff.	Std. Dev.
1	6	7	9	8	14	9	15	10
2	8	6	11	9	13	9	17	9
Avg.	7	6	10	8	14	9	16	10

The results for our BlockEscape experiment are summarized in Table 2. The subjects were able to use our head-tilt interface with similar success as the EyeKeys [4] or Camera Mouse [13] interfaces. All but one subject using our interface won their last three games. The other won their last two games, which indicates that practice improved the quality of the human-computer interaction. If we eliminate each subject's first game from the analysis, the average path deviation is 1.4 and the median is 0.6.

Table 2. Game-Interaction Experiment: Five users played 4 games of BlockEscape, issuing commands by head tilt and with three other interfaces [4]. Rows 1–3: The number of deviations of the block from the optimal path during gameplay, which are assumed to be due to false detections of the interface rather than misjudgments of the player

Interaction Method	Head Tilt	EyeKeys	Camera Mouse	Keyboard
Path Deviation: Median	2.2	2.5	0	0
Average	2.6	2.9	2.3	0
Std. Dev.	2.8	4.0	2.7	0
Wins	16/20 (80%)	10/12 (83%)	10/12 (83%)	12/12 (100%)

In the games that resulted in losses, the users tilted their heads too far sideways in executing a command. This motion and the return motion to the neutral upright state both took time. As a result, it took the subjects too long to issue the sequence of commands needed to navigate the block through the holes in the moving walls before the game ended. After gaining experience with the interface, the users soon became aware of this issue and played much better.

In the third experiment, users were able to follow an average of 5.9 links to new web pages, indicating that 11.8 head tilts were correctly detected for each incorrect detection. Individual results are in Table 3. These results indicate that our method may be used in a real-world context of surfing the Internet.

Table 3. Web Browsing Experiment: Number of web links correctly followed until a link was followed by mistake

Participant	Trial 1	Trial 2	Trial 3	Trial 4	**Mean**
1	3	5	7	2	4.3
2	6	8	9	8	7.8
3	7	7	6	9	7.3
4	4	3	5	2	3.5
5	8	7	7	4	6.5

5 Discussion and Future Work

Through our experimental results, we have shown that our system can act as a fast, responsive, and usable interface on its own. Our results are very promising since our system operated in less than ideal conditions: we used background subtraction for crude foreground estimation, and there were objects moving in the background. Since our method is not bound to a specific foreground model, and any background and foreground information collected by a program can be used as an input to our system, computational load may be significantly reduced, which is quite important in the context of human-computer interaction.

In our experiments, users were able to play the BlockEscape game and browse web pages effectively, demonstrating that our method works well in a real world context. Our method also offers new opportunities for people with disabilities. Used as a stand-alone interface or with another computer vision system, our technique could help facilitate a richer interactive experience for quadriplegic users who can control their head motion by enabling them to browse the web, enter text, or play a computer game. This is an important aspect of our contribution, since people with disabilities are dependent upon interface systems for their interaction with computers.

Using the eyes as possible cues to our method could make it even more robust. The eyes can be detected with a variety of methods (e.g., [4]), and we could use their axis of orientation to provide additional estimates to our system. To combine these estimates, the use of democratic integration [14] rather than equation 7 may be desirable in this case, since democratic integration can weigh system components relative to their current performance in an extremely effective fashion. These ideas could extend to tracking other facial features, such as nostril tracking. Other extensions would be to handle out of plane head rotation by fitting conic sections rather than circles. However, modeling such head turns is not as relevant to human-computer interaction, since users typically do not wish to interact with a computer that is not the focus of their attention.

In summary, we have introduced an efficient, accurate method for fast head tilt angle estimation. Our system operates on video data in real time with minimal computational requirements. Experiments with our system have shown that it is easy to use as an input and control device on its own and it has the potential to become an important part of a robust human-computer interaction system.

Acknowledgment

The authors would like to thank the subjects for their enthusiasm and patience in testing our method. Funding was provided by the National Science Foundation (IIS-0308213, IIS-039009, IIS-0093367, P200A01031, and EIA-0202067).

References

1. I.A. Essa and A.P. Pentland. Coding, analysis, interpretation, and recognition of facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):757–763, July 1997.
2. J.-J. J. Lien, T. Kanade, J. Cohn, and C.-C. Li. Automated facial expression recognition based on FACS action units. In *Proceedings of the Third IEEE International Conference on Face and Gesture Recognition*, pages 390–395, April 1998.
3. M.J. Lyons. Facial gesture interfaces for expression and communication. In *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 598–603, October 2004.
4. J.J. Magee, M.R. Scott, B.N. Waber, and M. Betke. Eyekeys: A real-time vision interface based on gaze detection from a low-grade video camera. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04)*, volume 10, pages 159–166, July 2004.
5. A. Raouzaiou, N. Tsapatsoulis, V. Tzouvaras, G. Stamou, and S. Kollias. A hybrid intelligent system for facial expression recognition. In *Proceedings of the European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (Eunite 2002)*, September 2002.
6. G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, Q2:15pp, 1998.
7. B. Schiele and A. Waibel. Gaze tracking based on face color. In *International Workshop on Automatic Face- and Gesture-Recognition*, pages 344–349, 1995.
8. K. Schwerdt and J. L. Crowley. Robust face tracking using color. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, Grenoble, France, March 2000.
9. H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *Proceedings of the 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, page 38, June 1998.
10. G. Shakhnarovich, P. A. Viola, and M. Baback. A unified learning framework for real time face detection and classification. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 16–26, May 2004.
11. J. Sochman and J. Matas. AdaBoost with totally corrective updates for fast face detection. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 445–450, May 2004.
12. B. Wu, H. Ai, C. Huang, and S. Lao. Fast rotation invariant multi-view face detection based on real AdaBoost. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 79–84, May 2004.
13. M. Betke, J. Gips, and P. Fleming. The Camera Mouse: Visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(1):1–10, March 2002.
14. J. Triesch and C. von der Malsburg. Democratic integration: Self-organized integration of adaptive cues. *Neural Computation*, 13(9):2049–2074, September 2001.