Menu Selection by Facial Aspect

Vera Bakić and George Stockman
Department of Computer Science and Engineering
Michigan State University, East Lansing, MI 48824
{bakicve1,stockman}@cse.msu.edu
http://www.cse.msu.edu/~bakicve1/faces/

Abstract

A non-intrusive real-time program detects the face and face features of a moving workstation user at a rate of between 10 and 30 Hertz. Based on the face pose, it determines where on the display the subject is looking. Button selection can be done by opening the mouth. The long term goal is to provide a system for controlling a computer using head movements and gaze direction. A skin color model is used along with geometric knowledge about the face and weak assumptions about the lighting. Good results are reported with various subjects and conditions, including facial hair, 3D motion, and use of eyeglasses.

Keywords: HCI, face tracking, face recognition, menu selection

1 Introduction

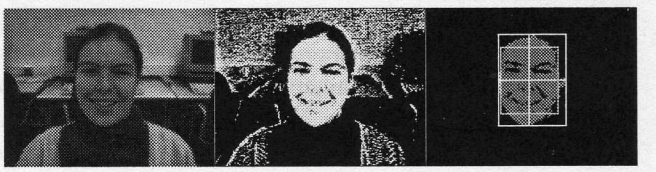
Recently, there has been a great deal of interest and progress in the analysis of faces. Potential applications are many, including recognition [14, 15, 13], gesture recognition [2, 16], face-spotting in images [11, 10, 6, 17], and gaze detection and HCI [9, 1].

The work described here is directed toward a general capability to detect and track a human face as it moves in a 3D workspace. Having achieved this capability, it can then be used to enable others. For example, knowing the approximate 3D head pose allows normalization for face recognition or for reduction of database search in the eigenface approaches to recognition [14, 13]. Or, 3D pose can be used directly for HCI or for evaluation of how humans explore computer displays or virtual environments.

At the higher level, ours is a programmed feature-based approach. A set of sample faces was studied to produce a skin color model that is used within a connected-components algorithm to extract a region that is likely to be a face. Various heuristics are used to eliminate false regions and to frame a real face; finally, additional heuristics use knowledge of faces and lighting to locate the eyes, eyebrows and nose. Location of these features determines gaze direction, and the user can move the screen pointer using just head movements and gaze direction. Button selection can be done by opening the mouth while the user is not moving. Section 2 describes the method for detecting the face features in real time, and location of the eyes, eyebrows and nose. Section 3 describes the overall architecture of the real-time face tracking program and reports results which show that the algorithm performs well with moderate control of the environment. Section 4 describes gaze direction determination. Section 5 describes one possible application of the system—menu selection using head orientation and gaze direction.

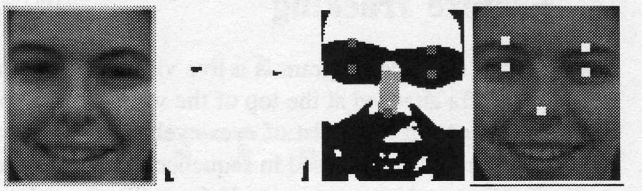
2 Feature Location

Our feature location algorithm consists of two main parts, described in the following subsections.



Original image Skin color class. Face region bound.

(a) Isolated face region in a sample image



Original Best threshold. Nose line Feature points

(b) Eyes, eyebrows and nose detection

Figure 1: Face and eyes/eyebrows/nose location

2.1 Face Detection

A skin color model is used within a connected components algorithm to extract a region that is likely to be a face. The input images are color images of red (R), green (G) and blue (B) components. The skin color model was derived from the 2-D plot of $R_{norm} = \frac{R}{R+G+B}$ vs. $G_{norm} = \frac{G}{R+G+B}$. The detection of face region boundaries in an input image is depicted in Figure 1 (a). Each pixel in the original image is classified according to the skin color model into skin and background pixels. The biggest connected component of class skin is assumed to be a face. The subimage determined by the bounding box of the face object is used in further processing.

2.2 Eyes, Eyebrows and Nose Location

Finding the eyes, eyebrows and nose is based on the knowledge of the face geometry (Figure 1 (b)). To find the eye blobs, gradual thresholding of smoothed red component intensity is done. For each thresholded image, the connected components algorithm is run to find dark blobs. Each two blobs that are candidates for the eyes are matched to find the nose. To find the eyebrows and nose, the image of the red component intensity thresholded at average intensity is used. The position of each eye is verified with existence of the top of the eyebrow, that is assumed to be just above the eye pupils. It is assumed that most of the lighting on the face is from above, and that the nose line is normal to the line between the eye pupils. The point where the nose line hits the below-threshold area is considered as the tip of the nose. Each match is evaluated based on additional heuristics, and the match with the highest score is selected as the eyes-nose match.

Figure 2 shows examples of feature detection for various pan, tilt and roll angle rotations, some of which are extreme. The program imposes a limitation of up to 45 degrees of roll angle. The eye features are lost if the head is panned completely to the left or right. As roll and pan angles increase, so does the error in detecting the tip of the nose.

3 Feature Tracking

The input to the base program is a live video stream from the eye-camera attached at the top of the workstation monitor, and the output is the list of eyes-eyebrows-nose coordinates. Frames are processed in sequence. To take advantage of movement history, we use a Kalman filter to estimate motion vectors of tracked feature points and future position of tracked feature points [7]. The predicted coordinates are used in two ways: (i) to verify the position of newly found features and (ii) if tracked features are lost to predict their location and thus smooth out tracking and avoid losing tracked features.

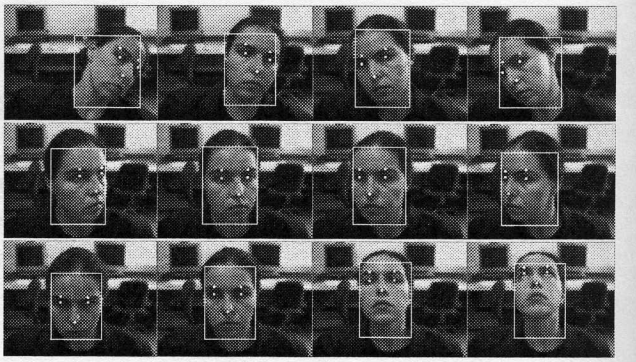


Figure 2: Results of eyes-nose detection for various roll, pan and tilt angles. The white rectangle shows face boundaries, and white dots show the locations of eyes, eyebrows and tip of the nose.

The system state diagram is depicted in Figure 3. Tracking starts in state `NO_FD` (no face detected). Once the face object is located (state `FD_FIRST`) in the input video stream, feature points are tracked using the approach described in Section 2.2 (state `FD_TRACK`). It is first attempted to find tracked features in the location of the previous frame. If that does not succeed, we will search for the new face position. We thus save time by not running the face detection algorithm which is the most time-consuming segment. Since the subject's motion is typically smooth, once we lock the face region, tracked features can be located in the same face bounding box for a number of video frames.

If tracked features cannot be found, prediction based on the Kalman filter is used to fill in the gap (state `FD_PREDICT`). Prediction is done for a limited number of frames. If tracked features are found while in the prediction state, we switch to the recovering state `FD_RECOVER` in which we only take measurements from the environment and make no prediction. The system stays in the recover state as long as it was in the prediction state. After the recovery period, if tracked features are found, we switch to the tracking state. If tracked features are not found while in the prediction or recover state, we start the face finding algorithm again (state `FD_NEWPOS`), since the assumption is that the state of the environment changed too much during the prediction period. If the face is not found, we switch to the initial state, while if the face is found, we switch to the intermediate continuation state `FD_CONT`. In that state, we attempt to find tracked features in the new location. If the features are found, we switch to the recover state. If the features are not found, we switch to the no match state `FD_NOMATCH`. In this state, we have a face object located in the image, however, face features are not found.

We estimate the motion of six variables (X and Y coordinates of both eyes and nose). In the Kalman filter equations, using previous and current values of the variables, we ma-

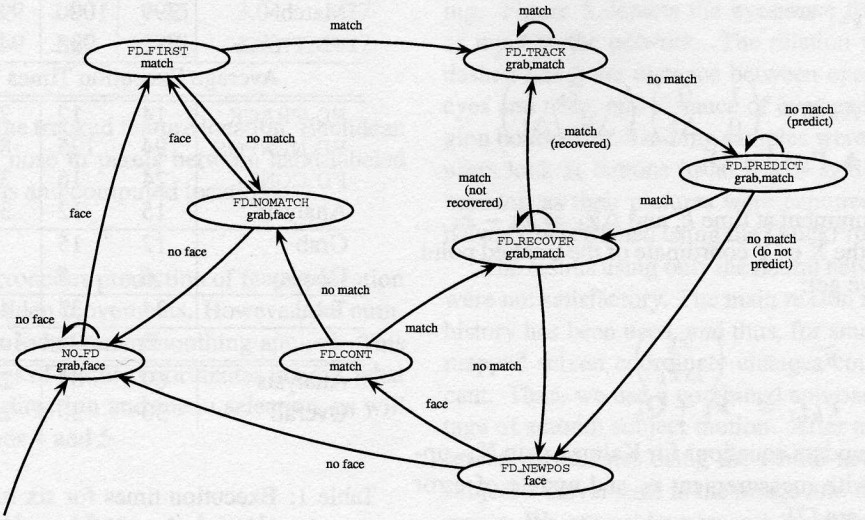


Figure 3: Tracking System State Diagram

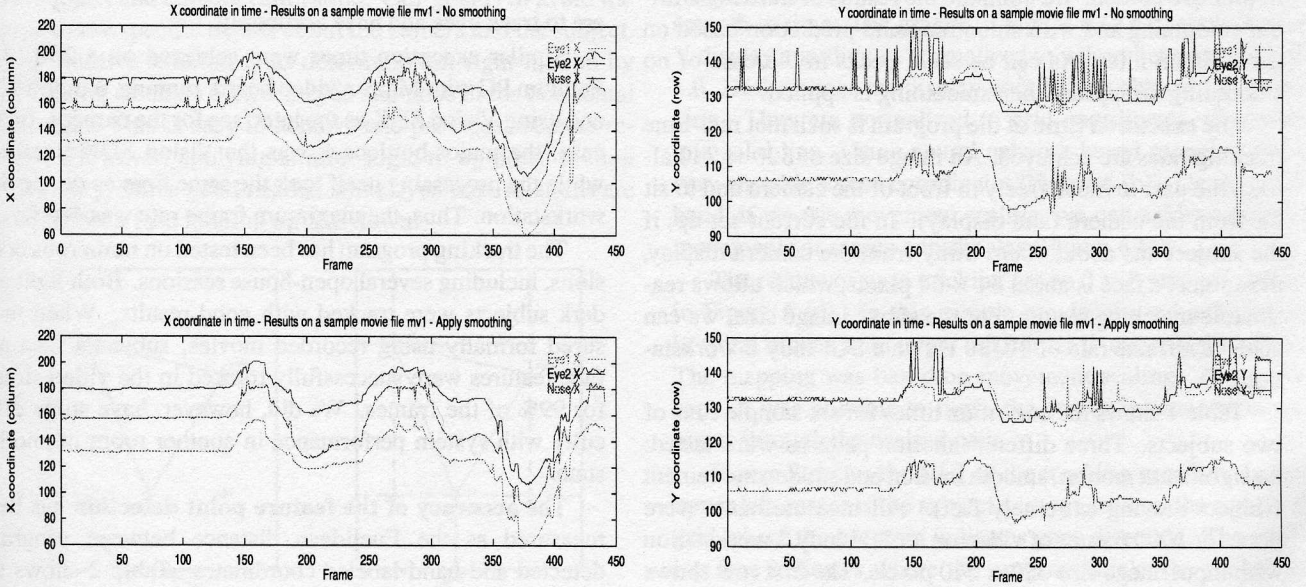


Figure 4: Comparison of X and Y coordinate tracking without (top) and with (bottom) use of Kalman filter for smoothing and prediction.

nipulate 2×2 matrices, and matrix operations like inversion are not time consuming. Time update ("predict") equations for the projected state of tracked point $\widehat{\mathbf{x}}_{k+1}^-$ and error covariance matrix $\mathbf{P}_{k+1}^-$ are:

$$\begin{aligned}\widehat{\mathbf{x}}_{k+1}^- &= \mathbf{A}_k\widehat{\mathbf{x}}_k = \begin{pmatrix}1 & 0\\ 0 & \Delta t\end{pmatrix}\begin{pmatrix}\widehat{x}_k\\ \Delta\widehat{x}_k/\Delta t\end{pmatrix}\\ \mathbf{P}_{k+1}^- &= \mathbf{A}_k\mathbf{P}_k\mathbf{A}_k^T + \mathbf{Q}_k\end{aligned}$$

where $\widehat{x}_k$ is measurement at time k, and $\Delta\widehat{x}_k = \widehat{x}_k - \widehat{x}_{k-1}$. In our case, $\widehat{x}_k$ is the X or Y coordinate of the tracked point.

For $\Delta t = 1$, we get:

$$\begin{aligned}\widehat{\mathbf{x}}_{k+1}^- &= \begin{pmatrix}\widehat{x}_k\\ \Delta\widehat{x}_k\end{pmatrix}\\ \mathbf{P}_{k+1}^- &= \mathbf{P}_k + \mathbf{Q}_k\end{aligned}$$

Measurement update equations for Kalman gain $\mathbf{K}_k$, update of estimate with measurement $\mathbf{z}_k$ and update of error covariance matrix are [7]:

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_k^-(\mathbf{P}_k^- + \mathbf{R}_k)^{-1}\\ \widehat{\mathbf{x}}_k &= \widehat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\widehat{\mathbf{x}}_k^-)\\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k)\mathbf{P}_k^-\end{aligned}$$

where we set $\mathbf{H}_k = \mathbf{I}$, and determine measurement error covariance matrix $\mathbf{R}_k$ and process noise $\mathbf{Q}_k$ empirically.

Figure 4 shows the results of feature tracking on a sample movie file: X and Y coordinates of eyes and nose points in time are plotted. We compare the results of tracking without smoothing and with smoothing and prediction based on the Kalman filter. As can be seen, peaks in the case of no smoothing disappear when smoothing is applied.

The **execution time** of the program is such that real-time tracking rates are achieved. An image size of 320×240 allows the user to move freely in front of the camera and to sit far from the camera (and display). In the current set-up, if the subject sits about $50cm$ away from the camera/display, the subject's face is about 64×64 pixels, which allows reasonable matching results. For the above image size, we can achieve a frame rate of 10–30 Hz on a SGI Indy 2 workstation.

Table 1 shows the execution times for six sample runs of two subjects. Three different motion patterns were tested: no significant motion, smooth motion and sudden movement (subject moving extremely fast). All measurements were done for 1000 frames of video on an SGI Indy 2 workstation with input image size 320×240 pixels. The first row shows total number of matches, and the second row shows number of frames spent in the tracking state. The execution time of the face detection algorithm (row FD_NEWPOS) is 80–90 msec, while the time needed to locate tracked features in the face (row FD_TRACK or FD_CONT) is 10–30 msec. The overhead induced by grabbing and displaying the image is not significant, and overall frame rate that can be achieved

	No Motion		Smooth Motion		Sudden Moves	
	R1	R2	R3	R4	R5	R6
Match	999	1000	998	999	958	799
FD_TRACK	992	998	941	924	623	431
Average Execution Times in milliseconds						
FD_TRACK	14	12	29	21	54	96
FD_NEWPOS	94	125	89	90	90	86
FD_CONT	24	16	37	15	60	53
Analysis	15	12	37	29	111	175
Grab	12	15	5	8	11	10
Display	6	5	7	5	6	5
Total	34	33	49	42	129	191
Frame Rates in Hz						
Analysis	69	81	27	35	9	6
Overall	30	30	20	24	8	5

Table 1: Execution times for six runs (two subjects). Program has been run for 1000 frames of video, on an SGI Indy 2 workstation with input image size 320×240.

for smooth motion is 20–30 Hz. In case of sudden movements, the subject's face position changes in every frame, thus the face detection algorithm dominates the execution times, thus the frame rate achieved is low. If we would use smaller images, frame rate would be higher; however, that would constrain the subject's movements significantly, if we are to track eyes and nose accurately.

Similar execution times were achieved on a 200 MHz Pentium PC with Matrox video board, running Windows NT and using Vision SDK as the interface for the camera. In this case, the major bottleneck was the Vision SDK interface, while the processing itself took the same time as on the SGI workstation. Thus, the maximum frame rate was 10 Hz.

The tracking program has been tested on numerous occasions, including several open-house sessions. Both light and dark subjects were tracked with good results. When measured formally using recorded movies, subject's face and face features were successfully tracked in the video stream for 99% of the frames. We did, however, have some difficulty with system performance in another room in another state.

The **accuracy of the feature point detection** has been measured as the Euclidean distance between program-detected and hand-labeled coordinates. Table 2 shows the accuracy measured on 3 movies (two light and one dark subjects, total length of 835 frames). The program detection was typically 1–4 pixels away from real locations. Location of eyes was 1–3 pixels away, while location of nose was 3–4 pixels away from hand-labeled location. These results were obtained when outlier matches were removed from the statistics. Errors are slightly higher when smoothing is ap-

	No smoothing			Apply smoothing		
	Eye 1	Eye 2	Nose	Eye 1	Eye 2	Nose
S1	1.42	1.44	3.42	2.23	2.29	4.09
S2	1.16	2.92	2.93	1.35	3.04	3.27
S3	1.21	1.21	3.52	1.58	1.53	3.87

Table 2: Accuracy of the tracked feature location. Euclidean distance for eyes and nose in pixels between hand-labeled eyes and nose locations and computed locations.

plied. This is due to erroneous prediction of feature location in case of subject's sudden movements. However, the number of outlier matches is lower if smoothing applied. This ensures smooth changes of feature coordinates in time, what is important for gaze direction and menu selection, as will be discussed in Sections 4 and 5.

4 Gaze direction determination

Once the coordinates of 3 points on the face are known, we can determine the subject's gaze direction. Currently, the program tracks only the user's head, unlike systems that track user's eye movements [5, 8].

The use of the P3P solution [4, 9] for 3D head pose based on the image features and a stored 3D triangle formed by the current user's features requires some knowledge about the subject and camera calibration. Our goal is to avoid the need to use specific details about the camera and the subject. One intuitive solution is to determine left-right motion by measuring distance between eyes and nose in the horizontal direction. This is easy to achieve and provides accurate results. However, applying similar logic to up-down motion does not produce good results, since eyes-nose relations are similar for a face looking up and down.

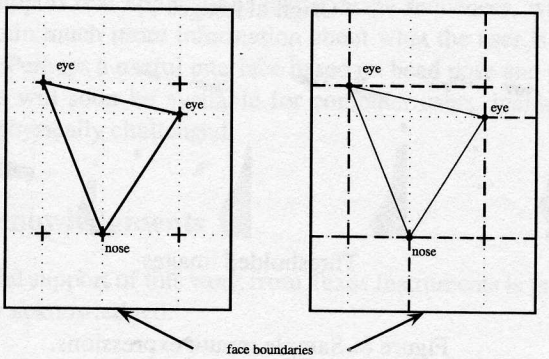


Figure 5: Relations between eyes and nose used as input to neural network to determine gaze direction. Dashed lines represent the relations we use as NN input.

To avoid this, we use an artificial neural network [12] to estimate the gaze direction. The input to the network are eyes and nose coordinates and their relations, and the output is normalized screen coordinates where the subject is looking. Figure 5 depicts the eyes-nose relations that we use as input to the network. The relation we use (depicted in dashed lines) are distance between eyes, distance between eyes and nose, and distance of eyes and nose from face region boundaries. Training samples were obtained by having users look at buttons in an 8×8 grid on the workstation monitor, as their pictures were captured. Then, the neural network was trained using the QuickProp program [3].

The results using only the neural network based mapping were not satisfactory. The main reason is that no movement history has been used, and thus, for small changes in input, mapped screen coordinate changes could be quite significant. Thus, we use a combined approach that takes advantage of smooth subject motion. After an initial estimate of screen coordinates using the neural network, we scale the subject's movements in the image into display cursor movements. We can achieve smooth movements of the cursor as long as the subject's head is moving smoothly. The logic is similar to that used for an ordinary mouse. In the long run, this approach enables us to adapt to individual users needs, where each user would decide how much he or she wants to move the head to move the cursor.

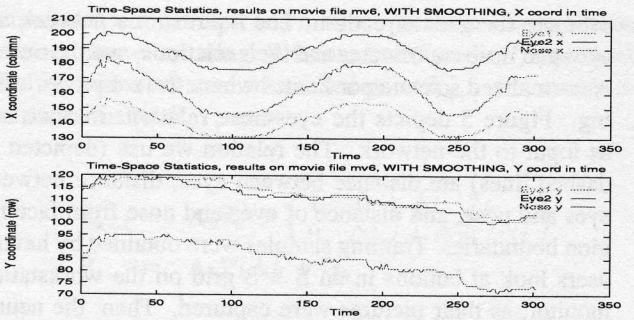
The results of gaze tracking for a 4×4 grid are depicted in Figure 6. The subject looks in spiral moves from the upper left corner to the lower left corner. As can be seen in Figure 6 (a), the X and Y coordinates of tracked points are smooth in time. The graphs in Figure 6 (b,c,d) show X and Y display coordinates normalized to values between 0.0 and 1.0. Dashed lines are display coordinates obtained by mapping. They are normalized to grid coordinates, displayed in a solid line. Using neural network based mapping gives very unstable screen coordinates (Figure 6 (b)). On the other hand, if we use movement scaling, we can achieve smooth movements of screen coordinates (Figure 6 (c)).

The results of gaze tracking for an 8×8 grid are shown in Figure 6 (d). In this sample movie, the subject looked in spiral moves from upper left corner to upper right corner. The mapping was based on movement scaling. As can be seen, it was possible to move the cursor in smooth movements through the desired path.

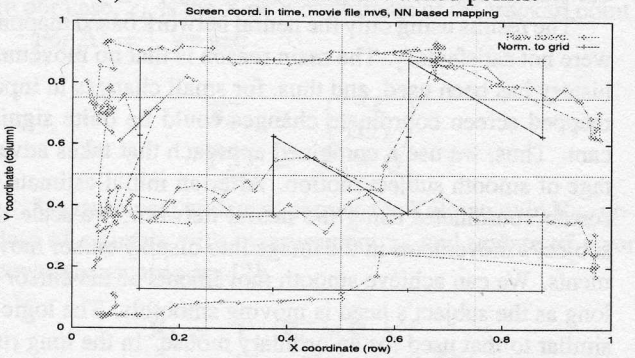
Figure 7 shows the GUI of the tracking program. In the upper left corner is the subject's image, and on the right is a grid where the program indicates gaze direction. The smiling face indicates where the program thinks the subject is looking, in this case towards row 2 column 2.

5 Application to Button Selection

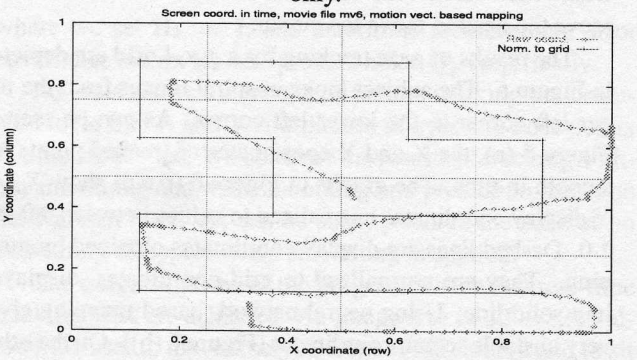
In this Section, we describe one possible application of our gaze tracking program. The application is simple button se-



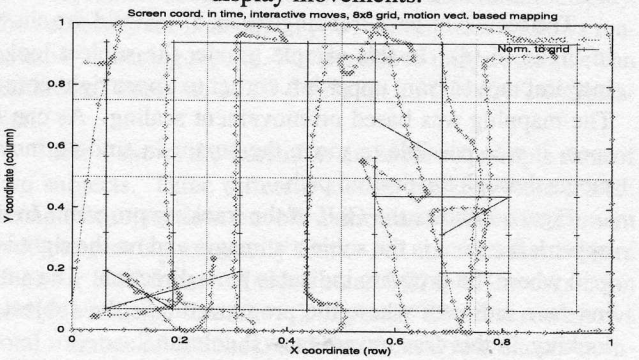
(a) X and Y coordinates of tracked points.



(b) Gaze tracking using neural network based mapping only.



(c) Gaze tracking using scaling of movements in picture to display movements.



(d) Gaze tracking on an 8×8 grid.

Figure 6: Results of gaze tracking on a sample movie using neural network only and using scaling of movements in the picture to display movements.

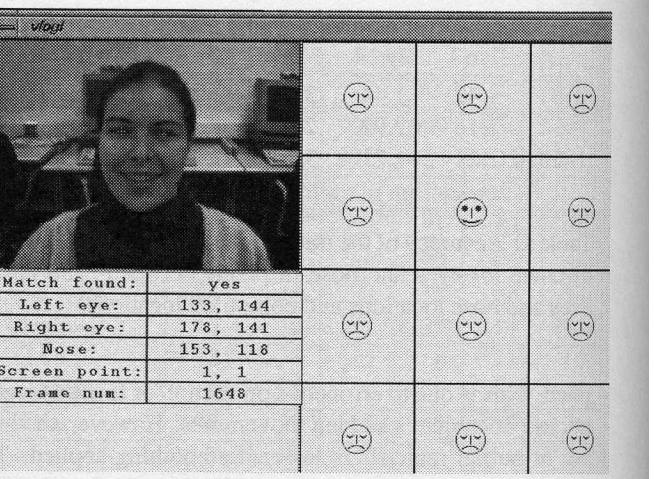


Figure 7: Snapshot of the face tracking program GUI: the smiling face shows in which area on the screen the subject is looking.

lection. Buttons are arranged in an $N \times M$ grid, and the subject's task is to select a button using head movements, gaze direction and face expression. We used an "open mouth" expression as the selection signal. The reason we use mouth expression instead of, say, eye blinking, is that mouth movements are done voluntarily. Eye blinking is not always done consciously, and we blink frequently to moisten our eyes. Thus, that is not the best way to control the computer.

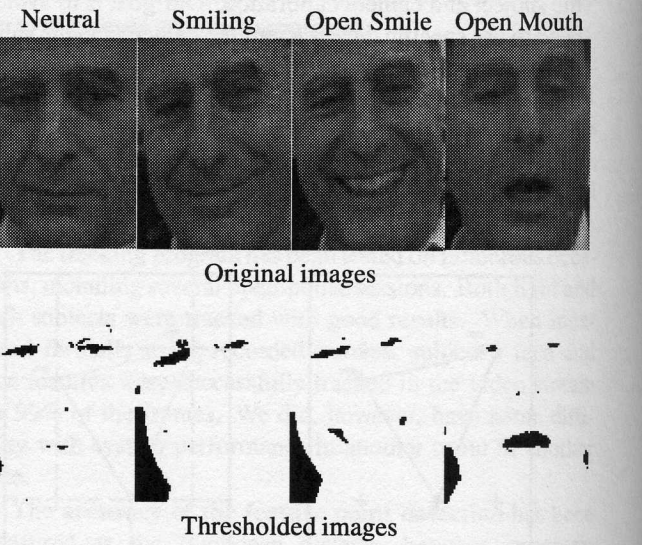


Figure 8: Sample mouth expressions.

To determine whether the mouth is opened or not, we look for a big, dark ellipsoidal blob just below the nose. The best threshold image from Figure 1 (b) is used to find the mouth. Figure 8 shows examples of blobs for several face expressions: neutral, smiling (with mouth closed), open

smiling (with mouth opened), and open mouth. For these four basic expressions, it can be clearly seen that the open mouth expression is distinct from the other three because of the dark blob.

The selection is done only when the subject is still for 5 or more frames. "Being still" is defined by the difference between the previous and current position of feature points being not more than 12 pixels. When a still state is discovered, we check for the state of the mouth. When the subject is moving, we do not check for the mouth state.

6 Conclusion

Ten years ago, Ohmura *et al.* [9] implemented a 10 Hz system which tracked 3 blue cosmetic spots on the face. Later, Ballard [1] was able to detect 3 facial points using only intensity imagery and no cosmetics; however, the lighting was harsh and the cycle time slow. Regarding the use of head-mouse, current systems are intrusive to the user: they either require special devices (infrared sensors) [5] or use head-mounted devices. The algorithm presented here meets its major requirement of real-time operation in a friendly environment, although implemented on a lightly loaded workstation. Implementation on a DSP should provide an inexpensive alternative for future applications. The standard SGI or PC eye-camera is used and the user is free to move in the workspace. Correct detections are made in most frames with modest environmental control, even for subjects with eye glasses. The algorithm has been tested on diverse subjects unseen in training with good results similar to those documented in the tables. Feature detection accuracy is adequate for choosing from typical displays of icons.

Our major interest is to explore protocols using image sequences to enhance communication between man and machine. We believe that the current performance is sufficient to support real applications using image sequences, which contain much more information about what the user is doing. Perhaps a useful interface based on head pose and gestures will soon be available for computer users, including the physically challenged.

Acknowledgments

Partial support of this work from Texas Instruments is gratefully acknowledged.

References

[1] P. Ballard and G. Stockman. Controlling a computer via facial aspect. *IEEE Transactions on System, Man and Cybernetics, Vol. 25, No. 4*, pages 669–677, April 1995.

[2] I. Essa and A. Pentland. A vision system for observing and extracting facial action parameters. *Proceedings of CVPR*, pages 76–83, 1994.

[3] Scott E. Fahlman. An empirical study of learning speed in back-propagation networks. Technical Report CMU-CS-88–162, Carnegie Mellon University, School of Computer Science, 1988.

[4] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. of ACM, Vol. 24*, pages 381–395, June 1981.

[5] T.E. Hutchinson, Jr. K.P. White, W.N. Martin, K.C. Reichert, and L.A. Frey. Human-computer interaction using eye-gaze input. *IEEE Transactions on System, Man and Cybernetics, Vol. 19, No. 6*, pages 1527–1534, November/December 1989.

[6] T. Jebara and A. Pentland. Parameterized structure from motion for 3D adaptive feedback tracking of faces. *Proceedings of CVPR*, pages 144–150, 1997.

[7] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*, pages 35–45, March 1960.

[8] Jr. K.P. White, T.E. Hutchinson, and J.M. Carley. Spatially dynamic calibration of an eye-tracking system. *IEEE Transactions on System, Man and Cybernetics, Vol. 23, No. 4*, pages 1162–1168, July/August 1993.

[9] K. Ohmura, A. Tomono, and Y. Kobayashi. Method of detecting face direction using image processing for human interface. *SPIE Visual Communication and Image Processing, Vol. 1001*, pages 625–632, 1988.

[10] N. Oliver, A. Pentland, and F. Bérard. LAFTER: lips and face real time tracker. *Proceedings of CVPR*, pages 123–129, 1997.

[11] H. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. Technical Report CMU-CS-95–158R, Carnegie Mellon University, School of Computer Science, 1995.

[12] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge, MA, and London, England, 1986.

[13] D. Swets and J. Weng. Using discriminant eigenvectors for image retrieval. *IEEE Transactions on PAMI, Vol. 18, No. 8*, pages 831–836, August 1996.

[14] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience, Vol. 3, No. 1*, pages 71–86, 1991.

[15] R. Uhl and N. da Vitoria Lobo. A framework for recognizing a facial image from a police sketch. *Proceedings of CVPR*, 1996.

[16] Y. Yacoob and L. Davis. Computing spatio-temporal representations of human faces. *Proceedings of CVPR*, pages 70–75, 1994.

[17] J. Yang and A. Waibel. A real-time face tracker. *Proceedings of WACV*, pages 142–147, 1996.