A Model-Based Gaze Tracking System

Rainer Stiefelhagen, Jie Yang, Alex Waibel
Interactive System Laboratories
Carnegie Mellon University
University of Karlsruhe
stiefel@ira.uka.de, yang+@cs.cmu.edu, waibel@cs.cmu.edu

Abstract

In this paper we present a non-intrusive model-based gaze tracking system. The system estimates the 3-D pose of a user's head by tracking as few as six facial feature points. The system locates a human face using a statistical color model without any mark on the face and then finds and tracks the facial features, such as eyes, nostrils and lip corners. A full perspective model is employed to map these feature points onto the 3D pose. Several techniques have been developed to track the features points and recover from failure. We currently achieve a frame rate of 15+ frames per second using an HP 9000 workstation with a framegrabber and a Canon VC-C1 camera. The application of the system has been demonstrated by a gaze-driven panorama image viewer. The potential applications of the system include multimodal interfaces, virtual reality and video-teleconferencing.

1. Introduction

For many human computer interaction applications it would be helpful to know where a person is looking at, and what he/she is paying attention to. Such information can be obtained from tracking the orientation of a human head, or gaze. While current approaches to gaze tracking tend to be highly intrusive – the subject must either stay very still, or wear a special device like a head mounted camera – in this paper we present a non-intrusive gaze tracking system that allows the user to move freely in the field of view of the camera.

There have been several approaches to compute the gaze of a person. First to mention, hardware-intensive and/or intrusive methods, where the user has to wear special headgear, or methods that use expensive hardware like radar range finder [8]. Recently, there have been proposed non-intrusive gaze trackers using mainly software. Baluja and Pomerleau proposed a method to estimate the eye-gaze onto a computer monitor [6]. In their approach however, the user has to stay in an almost fixed position and is not allowed to turn his head, and special lighting is needed. Gee & Cipolla developed a system to track the rotation and position of the head by finding correspondences between facial feature points and corresponding points in a model of the head, using a weak perspective projection [3]. However, the system has to be initialized manually because the system cannot locate the face and the facial feature points automatically.

We present a software-based system in this paper. The system estimates the 3-D pose of a user's head by tracking as few as six facial feature points. The system locates a human face using a statistical color model without any mark on the face. The system is able to find and track facial feature points automatically, as soon as a person appears in the field of view of the camera, and turns his face toward the camera. The system is also able to recover from tracking failures. The system then finds and tracks the facial features, such as eyes, nostrils and lip corners. To compute the pose we use the POSIT-algorithm, recently proposed by DeMenthon [1]. This algorithm iteratively approximates a full perspective solution of the pose, given at least four 3D to 2D correspondences.

The remainder of this paper is organized as follows: In section 2 we describe the search for the face, the eyes, nostrils and lip-corners. In section 3, the tracking of these features is described. Section 4 describes how we find a best subset of the feature points and how to predict true positions of outliers. In the following section, we describe our method to detect tracking failure and how to recover from it. In section 6, we show experimental results.

2. Searching the Features

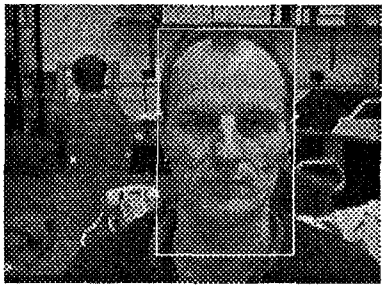
To search the facial features we use a top-down approach. First we search the facial area in the image, using a statistical color model, then the search of the facial features is restricted to certain areas inside the face.

0-8186-7728-7/96 $5.00

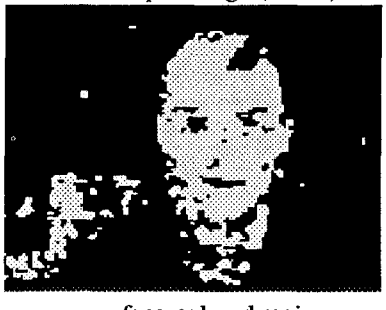
2.1. Searching the Face Using a Color Model

To find and track the face, we use a statistical color model, that consist of a two-dimensional Gaussian distribution of normalized face colors. The input image is searched for pixels with face colors. and the largest connected region of face-colored pixels in the camera image is considered as the region of the face. The color distribution is initialized so as to find a variety of face colors and is gradually adapted to the actual found face.

A description of the use of the color model to find and track faces can be found in [7].



input image (color!)



face-colored regions

Figure 1. Application of the color model to a sample input image. The face is marked in the input image

2.2. Searching the Pupils, Using Iterative Thresholding and Geometric Restrictions

Assuming a frontal view of the face initially, we can search the pupils by looking for two dark regions within a certain area of the face, that satisfy certain geometric constraints.

For a given situation, these dark regions can be located using a fixed thresholding within the search area. However, the threshold value may change for different people and lighting conditions.

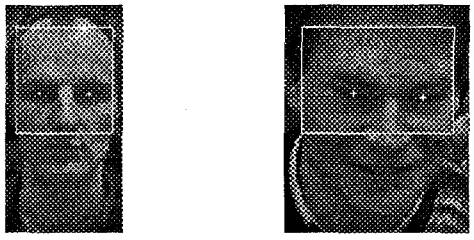


Figure 2. Search area for eyes

To use the thresholding method under changing lighting conditions, we developed an iterative thresholding algorithm. The algorithm iteratively thresholds the image, starting with a very low threshold k_0, until we find a pair of regions that satisfies our geometric constraints. Thresholding the image with increased values k_i eventually leads to more and bigger blobs, which constitute possible candidates for the eye regions, and finally a sufficient pair can be found.

$k_0 = 30$ $k_1 = 32$ $k_2 = 34$

Figure 3. Iterative thresholding of search-window for the eyes

Because the thresholding value is adjustable, this method is able to apply to various lighting conditions and to find the pupils in very differently illuminated faces robustly.

2.2.1 Geometric Constraints

Using knowledge about anthropometric measures such as approximate distance between eyes, and location of the eyes, and the assumption that we initially have a near-frontal view of the face, we have implemented the following constraints to choose and rank pairs of blobs.

First some initial constraints such as maximum size, maximum vertical extension and constraints on the position are imposed on the found regions.

Within each blob, that satisfies the initial restrictions, the darkest pixel is found and used as position of the eye candidate. These candidates are now checked pairwise. The pairs have to satisfy the following constraints: maximum and minimum horizontal distance, maximum vertical distance and symmetry. To measure symmetry according to the middle of the face, we use the following distance measure:

$$D(i,j) = |cand_i[x] - (w - cand_j[x])|,$$

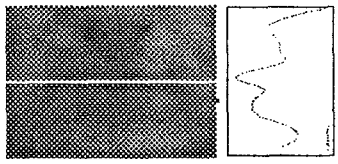


Figure 4. Integral projection of the search-window

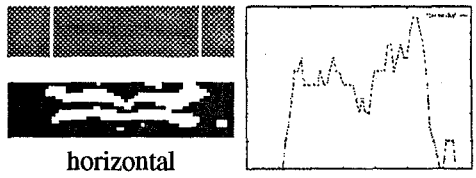


Figure 5. Finding horizontal borders of the lips, using a vertical projection of the horizontal edge-image of the lips

where $cand_i[x]$ and $cand_j[x]$ describe the horizontal position of the candidates, and w is the width of the search-region. D will be zero, if the candidates have the same distance from the border of the search-window and therefore lie perfectly symmetrically. As their distances to the boarder differ from each other, $D(i,j)$ increases linearly. If $D(i,j)$ exceeds a certain symmetry-distance D_{max}, the candidate pair (i,j) is rejected.

If more than one pair satisfies the above constraints, then the one with the least symmetry distance $D(i,j)$ is chosen.

2.3. Searching the Lip Corners

First, the approximate positions of the lip corners are predicted, using the positions of the eyes, the face-model and the assumption, that we have a near-frontal view. A generously big area around those points is extracted and used for further search.

Finding the vertical position of the line between the lips can be done by using a horizontal integral projection P_h of the greyscale image in the search region. P_h is obtained by summing up the greyscale values of the pixels in each row of the search area: $P_h(x) = \sum_{y=1}^{W} I(x,y)$ $, 0 \leq x \leq H$, where $I(x,y)$ is the intensity function of the search window, and W and H are the width and height of the search window, respectively. Because lip line is the darkest horizontally extended structure in the search area, its vertical position can be located where P_h has its global minimum. Figure 4 shows the search window for the lip-line and a rotated plot of the corresponding projection P_h. The vertical position, where P_h has its global minimum is marked in the image.

To obtain the horizontal boundaries of the lips, a smaller search area around the estimated vertical position of the line between the lips is extracted, and a horizontal edge operator is applied. The approximate horizontal boundaries of the lips can now be found, regarding the vertical integral projection P_v of this horizontal edge image. P_v is obtained by columnwise summing up the intensities of the pixels of the edge image. $P_v(y) = \sum_{x=1}^{H} E_h(x,y)$ $, 0 \leq y \leq W$, where $E_h(x,y)$ is the intensity function of the horizontal edge image, and W and H are the width and height of the search area, respectively.

The left and right boundaries of the lips can be located, where P_v exceeds a certain threshold t or falls below that threshold respectively. We choose t to be the average of the projection P_v. The vertical positions of the left and right lip corners can be found by searching for the darkest pixel along the columns at the left and right estimated boundaries of the lips in that search-region.

The use of integral projections to extract facial features is for example described in [2] or in Kanade's work on face recognition [5].

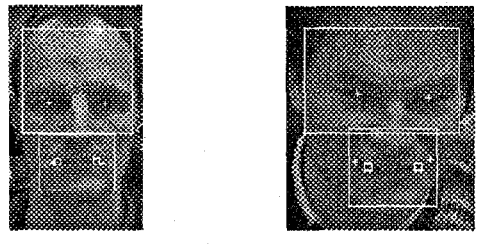


Figure 6. Initial search areas for the lips, and found lip-corners. The small rectangles mark the predicted positions of the lip-corners.

2.4. Searching the Nostrils

Similar to searching the eyes, the nostrils can be found by searching for two dark regions, that satisfy certain geometric constraints. Here the search region is restricted to an area below the eyes and above the lips. Again, iterative thresholding is used to find a pair of legal dark regions, that are considered as the nostrils.

3. Tracking the Features

For tracking, the features can be searched in small search windows around the last feature position. These search

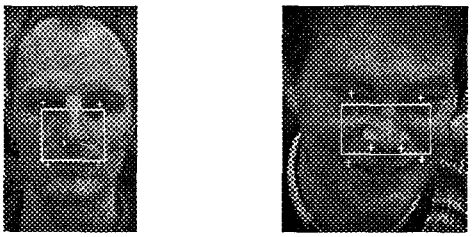


Figure 7. Search region for nostrils and found nostrils

windows additionally are predicted using linear extrapolation over the two previous positions of those features. The widths of the local search windows are all adjusted to the size of the facial regions.

In Figure 8 the search windows for all features are shown. The two white lines along the line between the lips indicate the search path along this line (see 3.3).

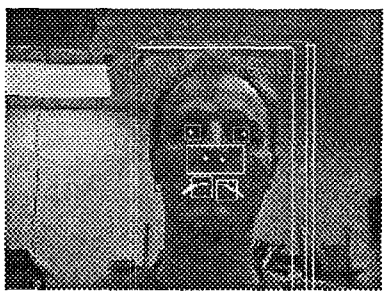


Figure 8. Search windows in tracking mode

3.1. Tracking the Face

In order order to beeing able to adjust parameters for the search windows of facial features to changing sizes of the face and to adjust parameters for the color model (see [7]), it is necessary to track the face in the image. Therefore, the face is searched in a search window around the last position of the face (see Figure 8). Because position and size of the face in the image will normally not change rapidly, it is not necessary to track the face in each frame. We tracked the face every 5 to 10 frames.

3.2. Tracking Eyes

For tracking the eyes, simple darkest pixel finding in the predicted search windows around the last eye positions is used.

3.3. Tracking Lip Corners

Tracking the lip corners consists of the following steps:

1. Predict the new positions of the lip corners
2. Search the darkest pixel in a search region right of the predicted position of the left corner and left of the predicted position of the right corner. The found points will lie on the line between the lips
3. Search the darkest path along the lip-line for a certain distance d to the left and right respectively, and choose positions with maximum contrast along the search path as lip corners

The search for the darkest pixel in the regions near the predicted lip corners ensures, that even with a bad prediction, a point on the lip-line is found, and the true positions of the lip corners can be found in the next step. Fig. 9 shows the two search windows for the points on the line between the lips. The two white lines mark the search paths along the darkest paths, starting from where the darkest pixel in the search windows have been found. The found corners are marked with small boxes.

Figure 9. Search along the line between the lips

3.4. Tracking the Nostrils

Tracking the nostrils is also done by iteratively thresholding the search region and looking for 'legal' blobs. But whereas we have to search a relatively big area in the initial search, during tracking, the search window can be positioned around the previous positions of the nostrils, and can be chosen much smaller. Furthermore, the initial threshold can be initialized with a value that is a little lower than the intensity of the nostrils in the previous frame. This limits the number of necessary iterations to be very small.

However, not always both nostrils are visible in the image. For example, when the head is rotated strongly to the right, the right nostril will disappear, and only the left one will remain visible. To deal with this problem, the search for two nostrils is only done for a certain number of iterations. If no pair of nostrils is found, then only one nostril is searched by looking for the darkest pixel in the search window for the nostrils.

To decide, which of the two nostrils was found, we choose that nostril, that leads to the more consistent pose estimation. This decision can be made, by choosing the one pose that implies the smoother motion (see section 4). The position of the nostril that was not found, can easily be predicted in the following frame using the current estimated pose (see figure 10).

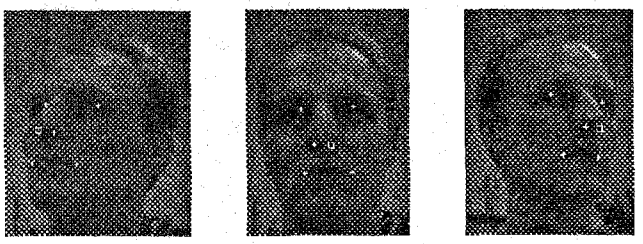


Figure 10. Predicted nostrils (marked with box)

4 Rejection and Prediction of Outliers

To increase the robustness as well as the accuracy of the system, we try to find outliers in the set of found feature points, and predict their true position in the next frame. At the same time, we use a most consistent subset of 2D to 3D point-correspondences to compute the pose, instead of using all found points.

To find a best subset we investigated two methods proposed by Cipolla [4]: Sample consensus tracking and temporal continuity tracking. Using the first method, that subset is chosen that leads to the best back-projection of model-points into the image-plane. Using the second method, the subset that leads to the pose that implies the smoothest motion is chosen as the best subset.

To compute the pose using the POSIT-algorithm we need at least four correspondences, and the object points should preferably be non-coplanar [1]. We chose the considered subsets as follows:

- In case, we only found one nostril, only the two subsets are considered, where the left or the right nostril is missing, respectively. This forces the system to choose, which of the two nostrils was found.
- In case both nostrils were found, the six subsets, where one feature is missing in each of them, are considered, plus the complete set of six correspondences.

Because the computation of the pseudo-inverse matrices corresponding to the used subsets can be computed off-line, the computation of the pose for each subset goes very fast.

Once the best subset of features is found, the true position of an out-lier can be easily predicted by projecting its model point into the image, using the computed pose. This prediction allows the system to recover from tracking errors and leads to a more robust tracking of the feature points.

5. Recovery from Tracking Failure

Tracking facial features on a freely moving person is a difficult task and once in a while tracking failure will occur. In order to build a robust useful gaze tracking system, the system has to be able to detect tracking failure and to recover from it automatically.

5.1. Detection of Failure

In our system we used mainly two methods for detection of failure: First, after each feature is located, it is checked, if its position lies within the found face region. If not, obviously some error occurred, and the features are searched again. Second, after all features are found, the model points are projected back onto the image plane, using the found pose, and the average distance between the back-projected model points and the actual found points is computed. If this average distance is above a certain threshold, then the actual found features and pose are rejected and failure is considered.

5.2. Searching the Features with Search Windows According to the Previous Pose

Once the system detected tracking failure, it switches to the search mode, and searches the features again. However, if failure occurs during tracking, we cannot assume a frontal view of the face anymore, because failure could have occurred at any possible rotation of the head, and the initial search might not work anymore. This problem can be solved by initializing the search windows and the geometrical restrictions according to the previously found pose. If failure occured, while the person was looking to the right, we then shift the search window for the eyes more to the right in the facial area, and more to the left, if the person was looking to the left.

Figure 11 shows the search windows for cases, where the person was looking to the left, near frontal or to the right in the image. Only the search windows for the eyes are shifted according to the pose. The subsequent search windows for lips and nostrils are adjusted according to the found position of the eyes or lips respectively.

6. Results

To evaluate the system, we recorded several sequences to hard disk, and the facial feature positions were marked by

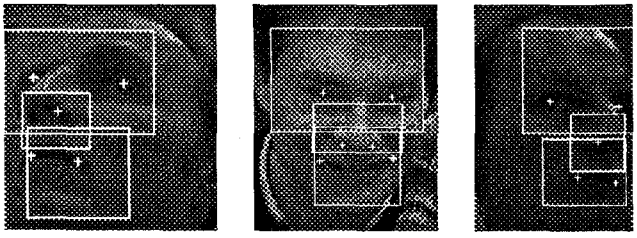


Figure 11. Adjusted Search Windows

	eyes		lips		nostrils		all features
method	X	Y	X	Y	X	Y	eucl. dist.
TC	3.2	2.5	3.2	2.1	2.0	2.5	4.1
SC	3.9	3.0	3.5	2.9	3.5	3.2	5.2
no pred	2.6	2.7	2.8	1.9	3.8	2.7	4.4

Table 1. Average location error in pixel.

method	R_x error	R_y error	R_z error
TC	5.5	7.6	2.2
SC	7.4	11.8	2.3
no pred	5.6	10.7	2.1

Table 2. Average rotation error in degrees for sequence 1.

method	T_x error (mm)	T_y error (mm)	T_z error (mm)
TC	7	4	63
SC	6	5	100
no pred	5	4	59

Table 3. Sequence 1: Average translation error in mm.

hand. With these manually marked positions, the reference pose for each frame was computed. Then, the gaze tracker was run with the pre-recorded sequences, and the obtained results were compared to the results, obtained by manually marking the sequence. Three tracking methods were investigated:

1. all found points are used to compute the pose and no prediction of outliers is done (*no pred*-method).
2. A best subset of points is found using the sample consensus method (*SC*-method), positions of outliers are predicted.
3. The best subset is chosen using the temporal continuity method (*TC*-method), positions of outliers are predicted.

While running the gaze tracker on the image sequence, the system lost the features during several frames, but recovered automatically from tracking failure. The average error of each parameter was computed just on those frames, where the gaze tracking system didn't consider the features as lost.

Table 1 to 3 show typical results that we obtained with one one of the sequences. Table 1 shows average errors in pixel for locating each feature. Table 2 and 3 show the average rotation and translation errors in millimeter for the same sequence. On our test sequences we achieved rotation errors as low as 5 degrees for rotation around the x- and y-axis and 1 degree for rotation around the z-axis.

In all our test sequences, using the temporal continuity tracking method lead to the best pose estimation results. Furthermore, using this method lead to a reduction of tracking failure of up to 60 % compared to using no prediction of outliers.

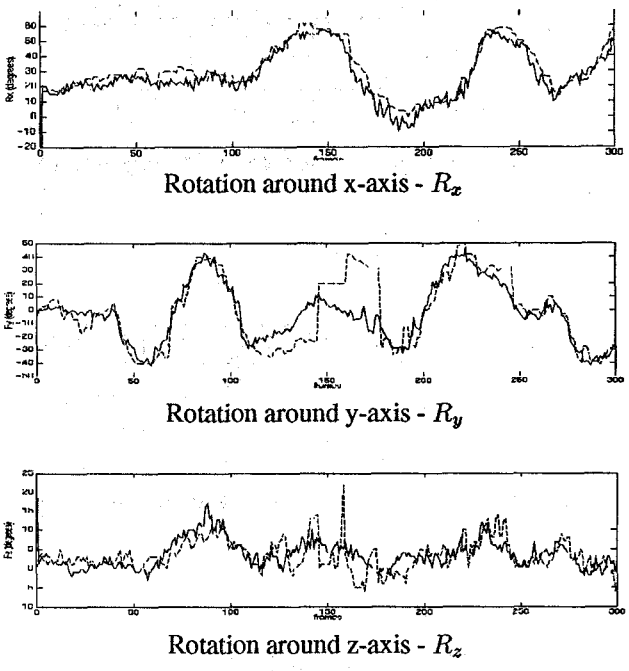
6.1. Discussion of Sample Test Sequence

Figure 12 shows plots of the rotation parameters R_x, R_y and R_z for test sequence "sequence 2". The solid lines indicate the reference rotation parameters, obtained with hand-labelled features and the dashed line shows the results obtained with our gaze tracker. Figure 13 shows a plot of the corresponding errors in R_x, R_y and R_z.

It can be seen, that for about the first one hundred and ten frames, the pose estimation is very close to the reference parameters. Then tracking failure occurs. Because no gross error occurred from the beginning of the failure – one eye was just found slightly off the real position – the system did not detect tracking failure immediately. At around frame 150 serious tracking failure occured and the system detected tracking failure. The tracker then starts searching for the features again, and fully recovers at frame 178. The features were then tracked again accurately and the pose estimates are very close to the reference parameters until frame 240. Here another failure occurs, but the system is able to recover after only three frames. This shows the ability of the system to recover from tracking failure.

7. Conclusion

We have developed a non-intrusive model-based gaze tracking system, which estimates the gaze by computing the pose of the user's head. The system achieves average rotation errors as low as 5 degrees for rotation around the x- and y-axis and as low as 1 degree for rotation around the z-axis and a frame rate of 15+ frames per second.

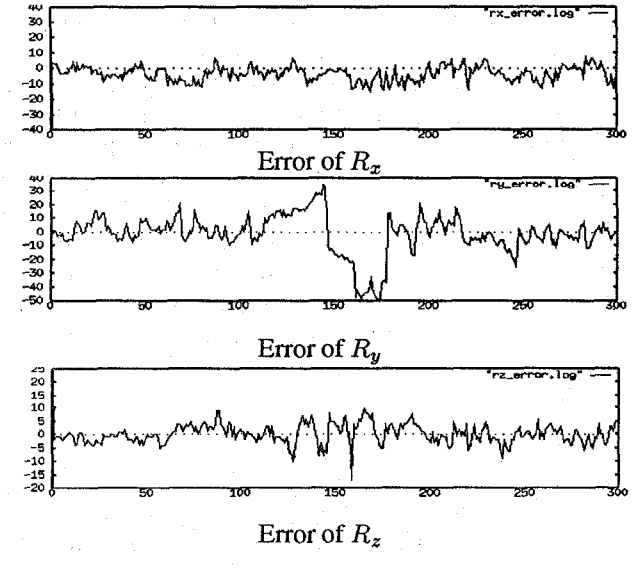


Rotation around x-axis - R_x

Rotation around y-axis - R_y

Rotation around z-axis - R_z

Figure 12. Estimated rotation angles with hand-labelled features (solid line) and with automatically tracked features (dashed line) on test sequence.



Error of R_x

Error of R_y

Error of R_z

Figure 13. Rotation errors

With our system, the user is allowed to move freely in the view of the camera and no special lighting or marks are needed. The system automatically finds and tracks the face and the facial feature points in the image and is able to recover from tracking failure automatically.

8. Acknowledgements

We thank our colleagues in Interactive Systems Laboratories for their technical supports to this project. This research was sponsored by the Advanced Research Projects Agency under the Department of the Navy, Naval Research Office under grant number N00014-93-1-0806.

References

[1] D. F. DeMenthon and L. S. Davis. Model based object pose in 25 lines of code. In G. Sandini, editor, *Computer Vision - ECCV 92, Proceedings Second European Conference on Computer Vision, Santa Margherita Ligure, May 1992*, pages 335 - 343. Springer Verlag, May 1992.

[2] R. Brunelli, T. Poggio. Face Recognition: Features versus Templates. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 10, October 1993.

[3] Andrew Gee and Robert Cipolla. Non-Intrusive Gaze Tracking for Human-Computer Interaction. *Proc. Mechatronics and Machine Vision in Practise, p. 112-117, Toowoomba, Australia, 1994*

[4] A. H. Gee and R. Cipolla, Fast Visual Tracking by Temporal Consensus. *Technical Report CUED/F-INFENG/TR-207, University of Cambridge, February 1995*

[5] T. Kanade, Picture processing by computer complex and recognition of human faces. Tech. Rep., Kyoto Univ., Dept. Inform. Sci., 1973.

[6] S. Baluja, D. Pomerleau. Non-Intrusive Gaze Tracking Using Artificial Neural Networks. CMU Tech. Report CMU-CS-94-102, 1994.

[7] J. Yang, A. Waibel. Tracking Human Faces in Real-Time. CMU Tech. Report CMU-CS-95-210, Nov. 1995.

[8] D. A. Simon, M. Hebert, T. Kanade. Real-time 3-D Pose Estimation Using a High-Speed Range Sensor. International Conference of Robotics and Automation Proceedings, May '94, San Diego.