A MODEL-BASED GAZE TRACKING SYSTEM

RAINER STIEFELHAGEN, JIE YANG and ALEX WAIBEL

Interactive System Laboratories

Carnegie Mellon University, USA

and

University of Karlsruhe, Germany

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 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.

Keywords: Gaze Tracking, Head Orientation, Human-Computer Interaction, Multimodal Interfaces, Facial Feature Extraction

1 Introduction

For many human computer interaction applications it would be helpful to know where a user is looking at and what his/her focus of attention is. 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 estimating 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 [1]. 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 [2]. In their approach, however, a user has to stay in an almost fixed position and is not allowed to turn his/her head, and a flash light is required. Gee & Cipolla [3] developed a system to track the rotation and position of a users' head by finding correspondences between facial feature points and corresponding points in a model of the head, using a weak perspective projection . However, the system has to be initialized manually because the system cannot locate the face and the facial feature points automatically.

We present a non-intrusive model-based gaze tracking 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. It 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/her face toward the camera. The system then finds and tracks the facial features, such as eyes, nostrils and lip corners. The system is also able to recover from tracking failures. We use the POSIT-algorithm proposed by DeMenthon & Davis [4] to compute the pose of the head. 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 formulate the gaze tracking problem as a pose estimation problem. Section 3 introduces methods for locating the face, eyes, nostrils and lip-corners. Section 4 describes the tracking of these features. Section 5 discusses how we find the best subset of the feature points and how to predict true positions of outliers. Section 6 describes methods to detect a tracking failure and how to recover from failure. Section 7 shows experimental results. Section 8 demonstrates an application of the gaze tracking system to multimodal human-computer interaction.

2 Gaze Tracking as Pose Estimation

A person's gaze direction is determined by two factors: the orientation of the head, and the orientation of the eyes. We limit our discussion to the head orientation in this paper. Then the gaze estimation can be formulated as a pose estimation problem. Pose estimation is to compute the 3D position and rotation of an object, based on a reference coordinate system. The position and rotation of the object can be described through a 3x3 rotation matrix $\mathbf{R}$ and translation vector $\vec{\mathbf{t}}$, which map the object coordinate system onto the reference coordinate system. Given a model of the head consisting of the 3D locations of the facial model points, such as eyes, lip corners and nostrils in the head coordinate system, and given the corresponding 2D positions of these model points in the camera image, the parameters for the rotation matrix $\mathbf{R}$ and translation vector $\vec{\mathbf{t}}$ can be computed.

Two issues are crucial for the pose estimation problem: finding feature points and computing pose. The problem of computing the object pose from 3D to 2D correspondences has been investigated extensively in the photogrammetry and computer vision literature. The approaches can be divided in two categories: closed-form solutions and numerical solutions. Closed-from solutions have been formulated for example in [5, 6, 7, 8]. However, these methods work only when the number of correspondences is limited. Whenever the number of correspondences exceeds four, iterative solutions are needed. Iterative solutions which make use of more feature points as can be dealt with in closed form solutions may be more robust with respect to noise and measurement errors because the pose information content becomes redundant. Such iterative methods were proposed for example by Lowe [9] and Yuan [10]. However, these techniques rely on the Newton-Raphson method, which presents two significant drawbacks: first, an initial pose has to be provided to start the iteration process; second, the pseudo-inverse matrix of a Jacobian matrix has to be computed in each iteration step, which is a computationally expensive operation.

Dementhon & Davis [4] proposed an iterative method – the POSIT algorithm – that works for an arbitrary number of point correspondences greater than three. The points may be either in general position (non-coplanar) or coplanar. Because no matrix inversions have to be computed in the iteration loop, the method is very fast and it is robust with respect to image measurements and to camera calibration errors. Furthermore it does not require an initial pose estimate. The method combines linear methods, necessary for weak perspective camera models, with non-linear methods, that are needed to compute the pose under a full perspective model. It approximates the full perspective solution, using linear computations [4, 8].

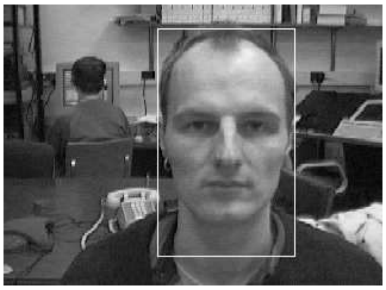
Since the POSIT algorithm has these advantages, and speed is essential for our task, we will use this algorithm to compute the head pose. The rest of the problem is how to find the feature points in a sequence of images, i.e., tracking problem. For feature-based tracking, a well-chosen feature can make the searching process much easier. Several factors, such as the existence of a preprocessing algorithm, the necessity, complexity and generality of the selected features, must be considered in selecting features for real-time tracking. Considering all these factors, we will track six facial feature points. In the rest of this paper, we will focus on developing fast and robust methods to locate and track facial features in a camera image in order to obtain a number of 3D to 2D correspondences to compute the pose.

3 Searching the Features

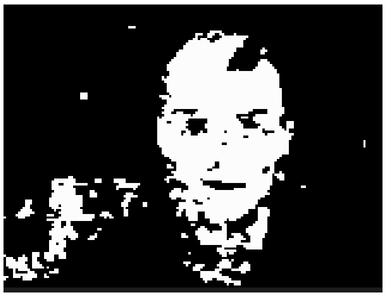
In order to search the facial features we use a top-down approach. First we search for a facial area in the image, using a statistical color model, then search for facial features within the facial area.

3.1 *Searching for a face*

To find and track the face, we use a statistical color model that consists of a two-dimensional Gaussian distribution of normalized skin colors. The input image is searched for pixels with skin colors. The largest connected region of skin-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. The interested reader is referred to [11].



input image (color!)



skin-colored regions

Figure 1. Application of the color model to a sample input image. The face is found in the input image (marked by a white rectangle)

3.2 *Searching for pupils*

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, which satisfy certain geometric constraints.

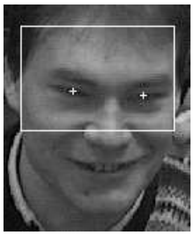


Figure 2. Search area for eyes

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.

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 the 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 can be applied to various lighting conditions and can find the pupils in very differently illuminated faces robustly.

3.2.1 *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 in the face. 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:

$$D(i,j) = |cand_i[x] - (w - cand_j[x])|, \tag{1}$$

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 pair with the least symmetry distance $D(i,j)$ is chosen.

3.3 ***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.

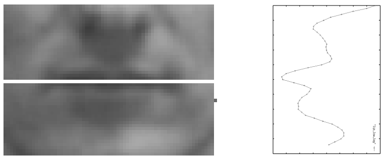


Figure 4. Integral projection of the search-window

Finding the vertical position of the line between the lips can be performed 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:

$$P_h(x) = \sum_{y=1}^{W} I(x, y) \quad , 0 \leq x \leq H, \tag{2}$$

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 the 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.

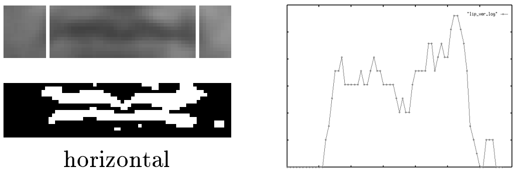
To obtain the horizontal boundaries of the lips, a smaller search area around the estimated vertical position of the line between the lips is extracted, and a horizontal edge operator is applied. The approximate horizontal boundaries of the lips can now be found, regarding the vertical integral projection P_v of this horizontal edge image. P_v is obtained by columnwise summing up the intensities of the pixels of the edge image.

$$P_v(y) = \sum_{x=1}^{H} E_h(x, y) \quad , 0 \leq y \leq W, \tag{3}$$

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 also described in [12, 13].



horizontal
edge-image

vertical projection

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

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

3.4 *Searching for the nostrils*

Similar to locating the eyes, the nostrils can be found by searching for two dark regions that satisfy certain geometric constraints. 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.

Figure 7. Search region for nostrils and found nostrils

4 Tracking the Features

For tracking, the features can be searched in small search windows around the last feature position. These search windows additionally are predicted using linear extrapolation over the two previous positions of those features. The widths of the local search windows are all automatically adjusted to the size of the face in the image.

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 4.3).

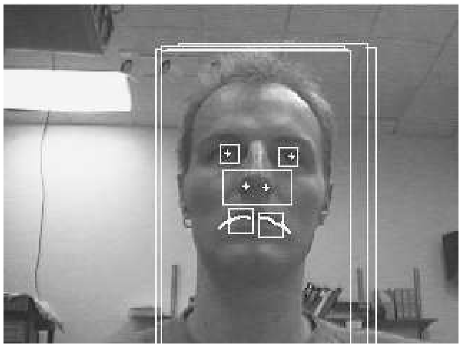


Figure 8. Search windows in tracking mode

4.1 *Tracking the face*

To track the face, the system searches in a search window at the predicted position, using the color model (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.

4.2 *Tracking eyes*

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

4.3 *Tracking lip corners*

Tracking the lip corners consists of the following steps:

(i) Predicting the new positions of the lip corners

(ii) Searching for 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

(iii) Searching for 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

Because the shadow between upper and lower lip is the darkest region in the lip-area, the searching for the darkest pixel in the search windows near the predicted lip corners guarantees, that even with a bad prediction of the lip corners, a point on the line between the lips is found. Then the true positions of the lip corners can be found in the next step. Figure 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

4.4 *Tracking the nostrils*

Tracking the nostrils is also achieved by iteratively thresholding the search region and looking for valid blobs. But whereas we have to search a relatively big area in the initial search, during tracking, the search window can be positioned at the predicted positions, and can be 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 performed 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 the nostril, that leads to the pose which implies smoother motion of the head compared to the pose obtained choosing the other nostril (see Section 5). The position of the other nostril can easily be predicted in the following frame using the current estimated pose, as shown in Figure 10, where the predicted location of the second nostril is marked with small rectangles.

5 Rejection and Prediction of Outliers

To increase the robustness and 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

Figure 10. Predicted nostrils (marked with box)

the same time, we use the most consistent subset of 2D to 3D point-correspondences to compute the pose, instead of using all found points.

To find the best subset we investigated two methods proposed by Gee & Cipolla [14]: 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 implying the smoothest motion is chosen.

To compute the pose using the POSIT-algorithm we need at least four correspondences, and the object points should preferably be non-coplanar [4]. We chose the considered subsets as follows:

- In case that 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 the subsets are considered, plus the complete set of six correspondences.

Because the pseudo-inverse matrices corresponding to these used subsets can be computed off-line, the pose for each subset can be estimated 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 onto 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.

6 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.

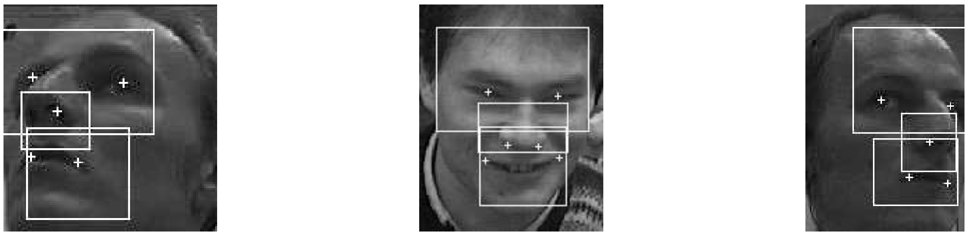


Figure 11. Adjusted Search Windows

6.1 *Detection of failure*

Tracking failure occurs, when one or more features couldn't be located or are mistakenly located at the wrong position. Whereas detection of the first case is trivial, detection of the second case is not easy.

In our system we use mainly two methods for detection of failure: first, after each feature is located, the system checks, 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. This distance can serve as a measure of confidence. If it is above a certain threshold, then the actual found features and pose are rejected, failure is considered and the features are being searched again.

6.2 *Searching the features based on the previous pose*

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 occurred, 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.

7 Results

To evaluate the system, we recorded several sequences to hard disk, and the facial feature positions were located manually. 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 with the manually marked sequence. Three tracking methods were investigated:

(i) All found points are used to compute the pose and no prediction of outliers is done (*no pred*-method).

(ii) The best subset of points is found using the sample consensus method (*SC*-method), positions of outliers are predicted.

(iii) 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 the results that we obtained with 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. In the 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.

Table 1. Average location error in pixel.

	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 2. Average rotation error in degrees for sequence 1.

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

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

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

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

7.1 *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 observed 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 occurred 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.

8 An Application: Controlling an Image Viewer [15]

In order to show the applicability of our gaze tracking system to human-computer interaction, we developed a multimodal interface to view panorama images. A panorama image is made from photographs, video stills, or computer renderings. Most panoramas are made from photographs as they provide the most realistic images. The QTVR Player is a stand-alone application for Mac or PC that lets you experience virtual reality scenes and objects from your desktop. It usually allows the user to scroll through 360 degree panorama images by using the mouse and to zoom in and out using the keyboard. In order to make the user hands free, we habe developed and interface that uses gaze to control scrolling through the panorama images, and voice-commands to control the zoom. The interface receives parameters describing the rotation of the users' head from the gaze tracker and parameters for the spoken commands from a speech-recognizer. It then sends the appropriate mouse- or key-events to the image viewer. The interface and the image viewer are running on a PC, and communication is done via sockets.

With such an interface, users can fully control the panorama image viewer without using their hands as shown in Figure 14. They can scroll through the panorama

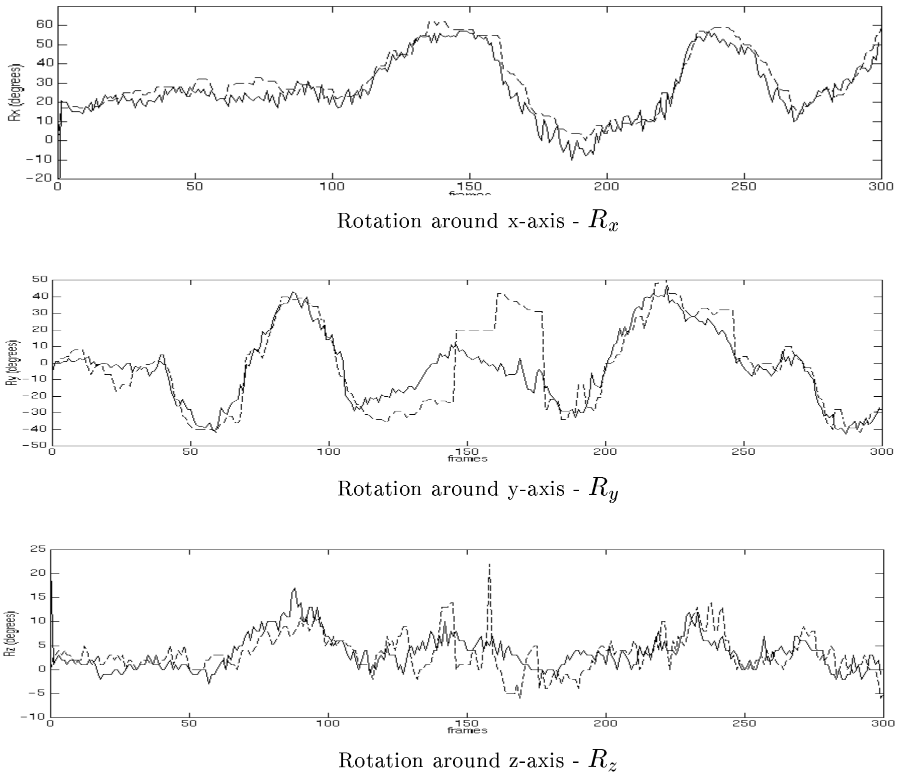


Figure 12. Estimated rotation angles with hand-labelled features (solid line) and with automatically tracked features (dashed line) on test sequence.

images in a natural way by looking to the left and right or up and down, and they can control the zoom by speaking commands such as "zoom in", "zoom out" or "zoom in three times". The concept of this interface can be extended to navigate in a virtual environment where the surrounding then can be rendered according to the users' gaze.

9 Conclusion

We have developed a non-intrusive real-time gaze tracking system, which estimates the gaze by computing the pose of the user's head. 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. Several algorithms to find and track facial features such as eyes, nostrils and lip corners have been developed. With the system, a user is allowed to move freely in the view of the camera and no special lighting or marks are needed. The system has achieved average rotation errors as low

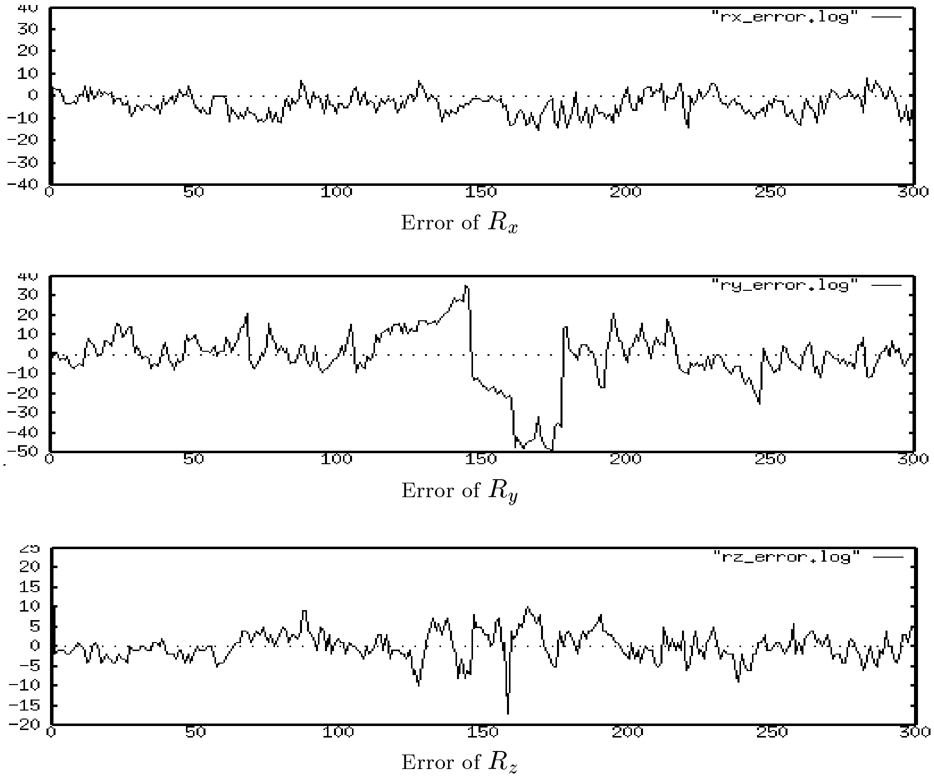


Figure 13. Rotation errors

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. The usefulness of the gaze tracker to human-computer interaction has been demonstrated by driving a multimodal interface to watch panorama images.

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. A. Simon, M. Hebert, and T. Kanade. Real-time 3-D pose estimation using a high-speed range sensor. In *International Conference of Robotics and Automation Proceedings*, pages 142–147, May 1994.

Figure 14. Controlling a panorama image viewer

[2] Shumet Baluja and Dean Pomerleau. Non-intrusive gaze tracking using artificial neural networks. Technical Report CMU-CS-94-102, Carnegie Mellon University, 1994.

[3] Andrew H. Gee and Roberto Cipolla. Non-intrusive gaze tracking for human-computer interaction. In *Proc. Mechatronics and Machine Vision in Practise*, pages 112–117, 1994.

[4] Daniel F. DeMenthon and Larry S. Davis. Model based object pose in 25 lines of code. In *Proceedings of Second European Conference on Computer Vision*, pages 335 – 343. Springer Verlag, May 1992.

[5] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[6] R.Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and image Processing*, 47(1):33–44, July 1989.

[7] L.G Roberts. Machine perception of three-dimensional solids. In J. Tippet et al., editor, *Optical and Electrooptical Information Processing*. MIT Press, 1965.

[8] Radu Horaud, Stéphane Christy, and Fadi Dornaika. Object pose: The link between weak perspective, para perspective, and full perspective. Technical Report No. 2356, INRIA, September 1994.

[9] D. G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis an Machine Intelligence*, 13(5):441–450, May 1991.

[10] J. S.-C. Yuan. A general photogrammetric method for determining object position and orientation. *IEEE Transactions on Robotics and Automation*, 5(2):129–142, April 1989.

[11] Jie Yang and Alex Waibel. A real-time face tracker. In *Proceedings of WACV*, pages 142–147, 1996.

[12] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 15(10), October 1993.

[13] Takeo Kanade. Picture processing by computer complex and recognition of human faces. Technical report, Kyoto Univ., Dept. Inform. Sci., 1973.

[14] Andrew H. Gee and Roberto Cipolla. Fast visual tracking by temporal consensus. Technical Report CUED/F-INFENG/TR-207, University of Cambridge, February 1995.

[15] Rainer Stiefelhagen and Jie Yang. Gaze tracking for multimodal human-computer interaction. In *Proceedings of International Conf. on Acoustics, Speech, and Signal Processing*, April 1997.