Gaze Tracking for Multimodal Human-Computer Interaction

Diplomarbeit

by
Rainer Stiefelhagen

at

Institut für Logik, Komplexität
und Deduktionssysteme
Fakultät für Informatik
Universität Karlsruhe (TH)
D-76128 Karlsruhe

Advisors:
Prof. Dr. Alex Waibel
Dr. Jie Yang

September 12, 1996

Hiermit versichere ich, die vorliegende Diplomarbeit persönlich und ohne unzulässige Hilfsmittel angefertigt zu haben. Alle verwendeten Quellen sind im Literaturverzeichnis aufgeführt.

Karlsruhe, den 15. Juli 1996

Contents

List of Figures 6

List of Tables 8

Acknowledgements 9

1 Introduction 10

2 Model Based Object Pose Estimation 13

2.1 Background 13

2.2 Camera Models [7] 14

2.2.1 Perspective Camera Model 14

2.2.2 Weak Perspective 17

2.3 From Weak Perspective to Perspective – the POSIT Algorithm [7] 18

2.4 Solving the Linear Equations [7] 19

2.4.1 Non Coplanar Object Points 20

2.4.2 Coplanar Object Points 20

2.5 Computing the Pose of the Head 21

3 Searching the Features **22**

3.1 Searching the Face Using a Color Model [24] 22

3.2 Searching the Pupils, Using iterative Thresholding and Geometric Restrictions 25

3.2.1 Iterative Thresholding of the Image 26

3.2.2 Geometric Constraints 28

3.3 Searching the Lip Corners 29

3.3.1 Predicting Locations of the Lip Corners 30

3.3.2 Use of Integral Projections to Locate the Lips 30

3.3.3 Looking for Maximum Contrast along the Line between the Lips 32

3.4 Searching the Nostrils 33

4 Tracking the Features **34**

4.1 Tracking the Face 35

4.2 Tracking Eyes 35

4.3 Tracking Lip Corners 35

4.4 Tracking Nostrils 36

5 Rejection and Prediction of Outliers **38**

5.1 Sample Consensus 39

5.2 Temporal Continuity Tracking 40

5.3 Prediction of the True Position of Outliers 41

6 Recovery from Tracking Failure **42**

6.1 Detection of Failure 42

6.2 Searching the Features with Search Windows According to the Previous Pose 43

7 Experimental Results **45**

7.1 Different Tracking / Pose Estimation Methods 45

7.2 Test Sequences 46

7.3 Feature Tracking Results 47

7.4 Pose Estimation Results 48

7.5 Discussion of Test Sequence "sequence 2" 50

8 A Multimodal Interface to control a Panorama Image Viewer **53**

9 Conclusions and Further Directions **54**

Bibliography **56**

List of Figures

2.1 Perspective camera model 16

3.1 Sample face 23

3.2 Face region 23

3.3 Face color distribution in chromatic color space 24

3.4 Different face color distributions 24

3.5 Application of the color model to a sample input image. The face is marked in the input image 25

3.6 Search area for eyes 26

3.7 Use of iterative thresholding to find the eyes. Note: In the real system, only the search area for the eyes has to be thresholded! 28

3.8 Iterative thresholding of search window for the eyes 28

3.9 Initial search areas for the lips, and found lip corners. The small rectangles mark the predicted positions of the lip corners. 30

3.10 Search of the vertical position of the line between the lips . . 31

3.11 Finding horizontal borders of the lips, using a vertical projection of the horizontal edge image of the lips 32

3.12 Search region for nostrils and found nostrils 33

4.1 Search windows in tracking-mode 34

4.2 Search for lip corners along the line between the lips 36

4.3 Predicted positions of second nostril 37

6.1 Different initialization of search windows according to the previous pose of the head 44

7.1 Sample images from evaluation sequence 2 47

7.2 Rotation errors 51

List of Tables

7.1 Average location error in pixel for sequence 1 47

7.2 Average location error in pixel for sequence 2 47

7.3 Average location error in pixel for sequence 3 48

7.4 Average location error in pixel for sequence 4 48

7.5 Average error of rotation in degrees for sequence 1. 49

7.6 Average error of rotation in degrees for sequence 2. 49

7.7 Average error of rotation in degrees for sequence 3. 49

7.8 Average error of rotation in degrees for sequence 4. 49

7.9 Sequence 1: Average error of translation in mm. 50

7.10 Plot of rotation parameters for test sequence 2 52

Acknowledgements

The work for this project was done during my visit at the Interactive Systems Lab at Carnegie Mellon University, Pittsburgh PA from October 1995 to July 1996.

First, I would like to thank Jie Yang for his advice and guidance during this project. It was really fun working toghether. I would also like to thank Alex Waibel for giving me the opportunity to work at CMU, which was a great experience. Thank is also due to Kin Chan for sharing some of his code and Anukol Kapur for help implementing the interface on the PC. Finally thanks to everybody at Interactive Systems Laboratories for technical support and discussions.

Chapter 1

Introduction

A multimodal human-computer interface is helpful to enhancing human-computer communication by the processing and combination of multiple communication modalities known to be helpful in human communicative situations. Many human-computer interaction applications require the information where a person is looking, and what he/she is paying attention to. This information provides communication cues to a multimodal interface. 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 be perfectly still, or wear a special device, in this report we present a non-intrusive model-based gaze tracking system.

A person's gaze direction is determined by two factors: the orientation of the head, and the orientation of the eyes. While the orientation of the head determines the overall direction of the gaze, the orientation of the eyes is determining the exact gaze direction and is limited by the head orientation. In this study, we focus on estimating the orientation of the head. The hereby obtained gaze estimates are precise enough for a lot of applications and the method allows gaze estimation, even when an observed person is rather far away from the camera and no high resolution images of the eye regions are available anymore. This would apply, for example, to situations where a person is allowed to freely move in a room. The eye gaze in addition could be further estimated on the top of the orientation of the head, to gain a more precise gaze estimation whenever high resolution images of the eye

regions are available. Many researches have been directed to obtaining the gaze of a person. The approaches can be classified as hardware or software based. Hardware-intensive and/or intrusive methods require the user to wear special headgear, or they use expensive hardware like radar range-finder [27]. Recently, there have been proposed non-intrusive gaze trackers using mainly software. Baluja and Pomerleau proposed a method to estimate the eye gaze on a computer monitor [23]. In their approach, the user however has to stay in an almost fixed position and is not allowed to turn his head. Also special lighting is needed. Cipolla & Gee 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 [17]. The system however has to be initialized manually because it cannot locate the face and the facial feature points automatically.

We present a software based system in this report. The system estimates the 3-D pose of a user's head by tracking as few as six facial feature points. Our 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 towards the camera. The system is also able to recover from tracking failures. To find facial feature points such as eyes, lip corners or nostrils in the image, we use a top-down approach. First, the system locates a face in the image. This is done by using a statistical color-model as described in [24]. Then, the facial features are searched inside the facial region. A full perspective model is employed to map these feature points onto the 3D pose. Several techniques have been developed to track the feature points and recover from failure.

Our system has achieved a frame rate of 15+ frames per second using an HP 9000 workstation with a framegrabber and a canon VC-C1 camera. On evaluation image sequences, we achieved 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. Average errors for feature location were as low as 2 - 3 pixel in x- and in y-direction.

To show the usefulness of the gaze tracker, we developed a multimodal interface to view panorama images. Therefore we used the gaze tracker to control scrolling through the 360 degree panorama images in Apple's Quick Time Movie Player, and voice-commands to control the zoom. The

interface client receives parameters describing the rotation of the head from the gaze tracker and parameters describing spoken commands from a speech recognizer. It then sends the appropriate mouse- or key-events to the image viewer. Both the interface and the image viewer are running on a PC, and communication is done via sockets. With such an interface, a user can fully control the panorama image viewer without using his/her hands. He can scroll through the panorama images by looking to the left and right or up and down, and he can control the zoom by speaking the commands "zoom in" or "zoom out".

Further work will include combining the system with eye-gaze tracking to obtain more accurate gaze estimations whenever the user is close to the camera and high resolution images are available, and introducing active camera control. Other potential applications of the system include virtual reality environments and tele-conferencing.

The remainder of this report is organized as follows. Section two gives a short overview of the problem of pose estimation from 2D to 3D correspondences and explains the algorithm that we used to compute the pose. In section three, the search of the face in the camera image and the search for the facial features such as eyes, lip corners and nostrils is described. Section four explains the tracking of these features. In section five, we describe two methods to find outliers in the set of found features, and how to predict their true positions. In the following section we adress the problem of detecting and recovering from tracking failure to build a more robust system. Section seven shows results using different tracking techniques, that we obtained with our system on prerecorded test sequences. In section eight we give a short overview of the multimodal interface and how we use the gaze tracker to control a panorama image viewer. Finally, in section nine follows a conclusion and we adress further research directions.

Chapter 2

Model Based Object Pose Estimation

2.1 Background

Pose estimation is to compute the 3D position and rotation of an object, based on a certain coordinate system.

In this study, we are interested in estimating a person's gaze by computing the position and rotation of his head. This can be formulated as a pose estimation problem, where we try to estimate the position and rotation of the person's head in the coordinate system that is attached to the camera. Given a model of the head, i. e., we know the 3D-positions of the facial model points, such as eyes, lip corners and nostrils in the object coordinate system, and given the corresponding 2D-positions of these points in the camera image, this pose estimation problem can be solved.

The problem of computing the object pose from 3D- (object points) to 2D-correspondences (image points) 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 work only when the number of correspondences is limited [5, 7]. Whenever the number of correspondences exceeds four, iterative solutions are needed [15, 9, 6]. A straightforward method is to compute

the elements of the perspective projection matrix [12, 13, 4], – which maps the feature points of the object onto their image projections in homogeneous coordinates – as solutions of a linear system. To find such a mapping, six correspondences have to be found. Alternative methods were proposed by Tsai [14], Lowe [9] and Yuan [15]. Tsai's method is especially useful when the intrinsic camera parameters, such as focal length, lens distortion and image center, are unknown. 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. Phong et al. [11] described a method, that works relatively good for a large number of correspondences. But, for three to ten correspondences, this method converges very slow and does not guarantee convergence.

Dementhon & Davis [1, 2] recently proposed a method that works for an arbitrary number of point correspondences greater than three. The points may be either in general position (non-coplanar) or coplanar. The method is very fast and robust with respect to image measurements and to camera calibration errors. It 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. Since this method has these advantages, we we will use it for our pose estimation problme. Some basic concepts are described in the rest of this section.

2.2 Camera Models [7]

2.2.1 Perspective Camera Model

In the setup as depicted in Figure 2.1, we denote by a 3-D point $P_i(X_i, Y_i, Z_i)$ in a frame that is attached to the object - the object frame. The origin this frame is the object point P_0. An object point P_i projects onto the image in p_i with camera coordinates x_i and y_i. The camera coordinates can be presented as:

$$x_i = \frac{\mathbf{i} \cdot \mathbf{P}_i + t_x}{\mathbf{k} \cdot \mathbf{P}_i + t_z}, \tag{2.1}$$

$$y_i = \frac{\mathbf{j} \cdot \mathbf{P}_i + t_y}{\mathbf{k} \cdot \mathbf{P}_i + t_z}, \tag{2.2}$$

where $\mathbf{P}_i$ is the vector from P_0 to P_i, and where the rigid body transformation from the object frame to the camera frame is given by:

$$T = \begin{pmatrix} & \mathbf{i}^T & & t_x \\ & \mathbf{j}^T & & t_y \\ & \mathbf{k}^T & & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} & R & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The relationship between the camera coordinates and the image coordinates may be obtained by introducing the intrinsic camera parameters:

$$u_i = \alpha_u x_i + u_c, \tag{2.3}$$

$$v_i = \alpha_v y_i + v_c, \tag{2.4}$$

where α_u and α_v are the vertical and horizontal scale factors and u_c and v_c are the image coordinates of the image center.

Dividing both the numerator and denominator of eqs. 2.1 and 2.2 by t_x, we can introduce the following notations:

- $\mathbf{I} = \mathbf{i}/t_z$ is the first row or the rotation matrix scaled by the z-component of the translation vector;
- $\mathbf{J} = \mathbf{j}/t_z$ is the second row of the rotation matrix scaled by the z-component of the translation vector;
- $x_0 = t_x/t_z$ and $y_0 = t_y/t_z$ are the camera coordinates of p_0 which is the projection of P_0 - the origin of the object frame, and
- $\epsilon_i = \mathbf{k} \cdot \mathbf{P}_i/t_z$.

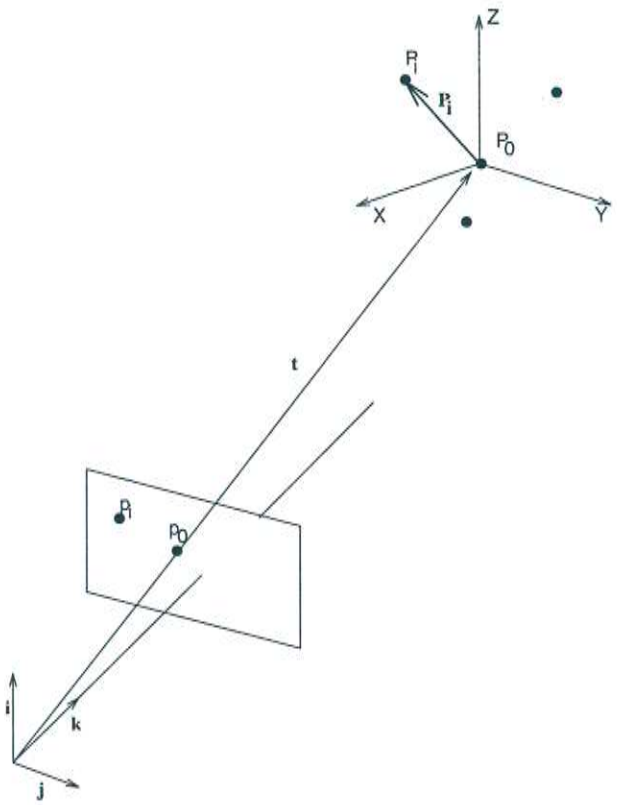


Figure 2.1: This figure shows the general setup. One of the object's points is selected as the origin of the object frame. Therefore, the pose parameters are the coordinates of this object point (P_0) in the camera frame and the orientation of the object frame with respect to the camera frame.

The perspective equations can then be rewritten as:

$$x_i = \frac{\mathbf{I} \cdot \mathbf{P}_i + x_0}{1 + \epsilon_i}, \tag{2.5}$$

$$y_i = \frac{\mathbf{J} \cdot \mathbf{P}_i + y_0}{1 + \epsilon_i}, \tag{2.6}$$

or:

$$x_i(1 + \epsilon_i) - x_0 = \mathbf{I} \cdot \mathbf{P}_i, \tag{2.7}$$

$$y_i(1 + \epsilon_i) - y_0 = \mathbf{J} \cdot \mathbf{P}_i. \tag{2.8}$$

Whenever the object is at some distance from the camera, the ϵ_i are small compared to 1, and an approximation of the perspective solution can be introduced: weak perspective.

2.2.2 Weak Perspective

The scaled orthographic projection (SOP), or *weak perspective*, is an approximation of full perspective.

With this approximation, we assume that for a given object in front of the camera, there is relatively little change in depth, i. e., the depths of the different object points P_i are not very different from each other, and are therefore all similar to the depth of the origin P_0 of the object coordinate system.

With SOP (weak perspective), the image of an object point P_i will be a point p_i^w in the image plane with the following coordinates:

$$x_i^w = fX_i/Z_0,\ y_i^w = fY_i/Z_0,$$

whereas with full perspective projection, we obtain a point p_i with coordinates:

$$x_i = X_i/Z_i,\ y_i = fY_i/Z_i.$$

The relation $s = f/Z_0$ is the scaling factor of the SOP. The point of reference P_0 has got the same image point p_0 under weak and under full perspective projection.

Weak perspective assumes that the object points lie in a plane parallel to the image plane passing through the origin of the object plane. This is equivalent to a zero-order approximation:

$$\frac{1}{1+\epsilon_i} \approx 1\ \forall i, i \in 1 \ldots n$$

With this approximation, eqs. 2.5 and 2.6 become:

$$x_i^w - x_0 = \mathbf{I} \cdot \mathbf{P_i}, \tag{2.9}$$

$$y_i^w - y_0 = \mathbf{J} \cdot \mathbf{P_i}, \tag{2.10}$$

where x_i^w and y_i^w are the camera coordinates of the weak perspective projection of the point P_i. By identification with eqs. 2.7 and 2.8 we obtain the relationship between the weak perspective and the perspective projections of P_i:

$$x_i^w = x_i(1 + \epsilon_i), \tag{2.11}$$

$$y_i^w = y_i(1 + \epsilon_i). \tag{2.12}$$

For an analysis of the quality of the SOP and further details, see [7].

2.3 From Weak Perspective to Perspective – the POSIT Algorithm [7]

In order to solve the pose problem, Dementhon & Davis [1] noticed that eqs. 2.9 and 2.10 are similar to eqs. 2.7 and 2.8 for which the ϵ_i are set to zero. We therefore can conclude:

- Whenever the ϵ_i are fixed, the pose equation 2.7 and 2.8 become linear in **I** and **J**. If at least four object points are provided, a solution can be found.
- It is possible to solve eqs. 2.7 and 2.8 *iteratively* by successive linear approximations.

In this case, the pose algorithm starts with a weak perspective camera model and computes an approximate pose. This approximated pose is improved iteratively as follows:

1. $\forall i, i \in \{1 \ldots n\}, n \geq 3, \epsilon_i = 0$;
2. Solve the over constrained linear system of equations 2.7 and 2.8 which provides an estimation of vectors **I** and **J**;

3. Compute the position and orientation of the object frame with respect to the camera frame:

$$\begin{aligned} t_z &= \frac{1}{2}\left(\frac{1}{\|\mathbf{I}\|} + \frac{1}{\|\mathbf{J}\|}\right) \\ t_x &= x_0 t_z \\ t_y &= y_0 t_z \\ \mathbf{i} &= \frac{\mathbf{I}}{\|\mathbf{I}\|} \\ \mathbf{j} &= \frac{\mathbf{J}}{\|\mathbf{J}\|} \\ \mathbf{k} &= \mathbf{i} \times \mathbf{j} \end{aligned}$$

4. For all i compute:

$$\epsilon_i = \frac{\mathbf{k} \cdot \mathbf{P_i}}{t_z}$$

If the ϵ_i don't change anymore, stop the procedure, otherwise go to step 2.

A geometric interpretation of this algorithm can be found in [1, 2].

2.4 Solving the Linear Equations [7]

The over constrained equation system of eqs. 2.7 and 2.8 can be written in matrix from as follows:

$$\underbrace{P}_{n\times 3}\underbrace{\mathbf{I}}_{3\times 1} = \underbrace{\mathbf{x}}_{n\times 1} \tag{2.13}$$

$$\underbrace{P}_{n\times 3}\underbrace{\mathbf{J}}_{3\times 1} = \underbrace{\mathbf{y}}_{n\times 1} \tag{2.14}$$

where P is a $n \times 3$ matrix formed by the 3-D coordinates of n vectors $\mathbf{P_1} \ldots \mathbf{P_n}$. Since the point P_0 is the origin of the objhect frame, this matrix can be written as:

$$P = \begin{pmatrix} X_1 & Y_1 & Z_1 \\ \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n \end{pmatrix}$$

To solve these linear equations, two cases have to be distinguished: non coplanar an coplanar sets of object points.

2.4.1 Non Coplanar Object Points

If the object points are not coplanar, the rank of P is 3 and therefore the solutions for $\mathbf{I}$ and $\mathbf{J}$ are simply giben by:

$$\begin{aligned} \mathbf{I} &= B\mathbf{x} \\ \mathbf{J} &= B\mathbf{y} \end{aligned}$$

whereas $B = (P^T P)^{-1} P^T$ is the pseudoinverse matrix of matrix P Notice, that the pseudo-inverse matrix B can be computed off-line and hence the estimation of $\mathbf{I}$ and $\mathbf{J}$ is particularly efficient.

2.4.2 Coplanar Object Points

If the object points are coplanar, P has rank 2 and additional constraints have to be used for the solution of the equation system. See for example [3] or [7] for a discussion.

In that case, two possible solutions for $\mathbf{I}$ and $\mathbf{J}$ are obtained. These solutions correspond to the well known reversal ambiguity associated with an affine camera model.

In each iteration, therefore two poses are obtained, which have the same translation vector.

2.5 Computing the Pose of the Head

As already mentioned above, we formulated the gaze estimation problem as a pose estimation problem and try to compute the pose of the user's head by finding correspondences between facial model points and their image points. Using Dementhon's and Davis' POSIT-algorithm, it is now necessary to find at least four 3D- to 2D-correspondences to compute the pose, and the model-points should preferably be non-coplanar to each other. In this study we therefore try to search and track the following six features in the image: the eyes, the lip corners and the nostrils. They form a non-coplanar set of feature points.

In the following sections we will now discuss how to search and track these features in the image.

Chapter 3

Searching the Features

In order to track facial features, these features have to be located in the image first. This initial search is more complicated than tracking the features, because no information is available about previous feature locations that could help to restrict the search-regions.

To search the facial features such as eyes, lip corners and nostrils, we use a top-down approach. First we search the facial area in the image, using a statistical color model. Once the face is found, the search of the facial features can be restricted to certain areas inside the face.

3.1 Searching the Face Using a Color Model [24]

Our approach to find and track a face in the camera-image, is to use color-information.

If we analyze the face region in an image (see Figure 3.1 and Figure 3.2), we can discover the skin color distribution clusters in a small area of the chromatic color space (Figure 3.3):

We have further found that distributions of skin colors of different people are clustered in chromatic color space. Although skin colors of different people appear to vary over a wide range, they differ much less in color than in brightness. In other words, skin colors of different people are very close

Figure 3.1: Sample face

Figure 3.2: Face region

but they differ mainly in intensities. The following Figure shows a skin color distribution of forty people with different skin colors in the chromatic color space. The distribution was obtained by analyzing faces of different races, including Asian, African American, and Caucasian.

By closely investigating the face color cluster, we have discovered that the distribution has a regular shape. A close view of skin color distributions is shown in the following. Figure 3.4 (a) and (b) are color distributions of a face under different lighting conditions and 3.4 (c) is the color distribution of two person's faces.

It is obvious that the human face colors of different people under different lighting conditions in the chromatic color space have similar Gaussian distributions. Therefore, a face color distribution can be represented by a

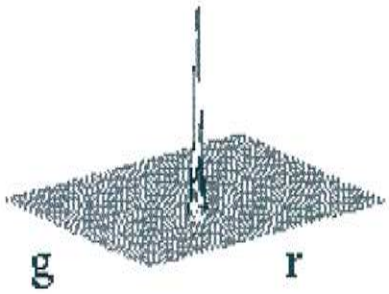


Figure 3.3: Face color distribution in chromatic color space

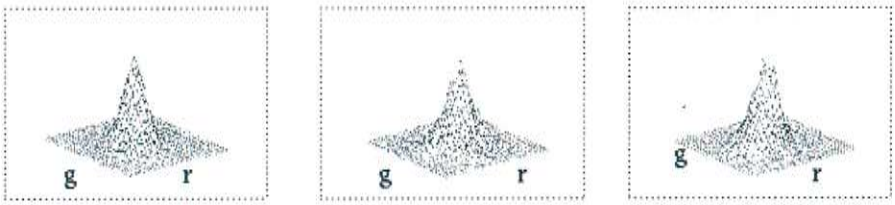


Figure 3.4: Different face color distributions

Gaussian model.

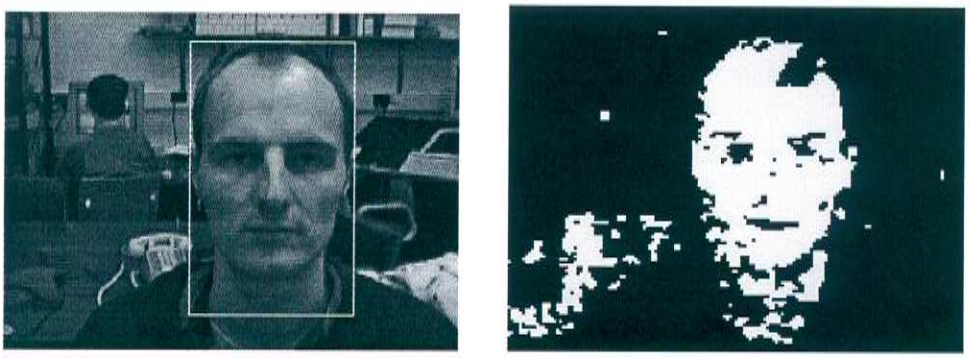
Since the model only has six parameters, it is easy to estimate and adapt them to different people and environments.

The input image is searched for pixels with face colors. Pixels with colors near the means of the color distribution are considered as possible facial areas. The largest connected region of face colored pixels in the camera image is considered as the region of the face.

Initially, a general color distribution with a relatively big variance is

used, to be able to find faces with different face colors, and under different lighting conditions. This initial color distribution can be obtained once by collecting images of different people with varying face colors and under different lighting conditions .

Figure 3.5 shows a sample input image (Note that we use a color image originally!) and the application of the face color classifier to it.



input image (color!) face-colored regions

Figure 3.5: Application of the color model to a sample input image. The face is marked in the input image

Once a face is found in the image, the color distribution is constantly adapted to the actual face colors to increase robustness of the search even under changing lighting conditions (for example, caused by the person moving to different positions in a room).

A thorough description of the use of the statistical color model to find and track the face can be found in [24]

After the face has been found, the facial features can be searched in a restricted area within the face.

3.2 Searching the Pupils, Using iterative Thresholding and Geometric Restrictions

Within the face, the pupils are two dark regions that satisfy certain geometric constraints, such as position inside the face, symmetry according to

the facial symmetrie axis and minimum and maximum width between each other. We assume, that the user is initially looking straigth into the camera, and that we therefore have a frontal or near-frontal view of the face. Then the search area for the pupils can be restricted to lie within the upper half, and within some margins away from the borders of the facial area. We can now search the pupils by looking for two dark regions within the search area, that satisfy the mentioned geometric constraints.

Figure 3.6: Search area for eyes

In order to robustly locate the pupils, we have developed an iterative thresholding method.

3.2.1 Iterative Thresholding of the Image

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. For different values of the threshold and different lighting conditions, the found regions will vary a lot as well in size as in number. For example, by thresholding a very bright image of a face with the same threshold as a very dark or shady image, we might obtain no blobs at all in the bright image and a lot of blobs in the dark image.

To use the thresholding method under changing lighting conditions, we developed an iterative threshold algorithm. The algorithm iteratively thresholds the image, starting with a very low threshold, until we find a pair of regions that satisfies our geometric contstraints.

Algorithm 1:

1. Set threshold $k = k_0$
2. threshold the search area with threshold k
3. apply some initial constraints to find candidates among blobs (see 3.2.2)
4. check and rank the candidate regions according to geometric constraints (see 3.2.2)
5. if none of the pairs satisfies the constraints: increase threshold k ($k = k + c$) and go to step 2)

   else: choose highest ranked pair of candidate regions as eye regions. Stop.

The positions of the pupils are found, looking for the darkest pixels in the obtained two eye regions.

The initial threshold k_0 is chosen in such a way, that no blob at all will appear in the search region. Thresholding the image with an increased value will eventually lead to more and bigger blobs, which constitute possible candidates for the eye regions, and finally a sufficient pair can be found. In fact, in most of the cases, the first two candiates that appear are already the actual pupils, because they are the darkest objects in the search region and other facial features such as nostrils or hair will lead to blobs that usaually are not accepted as candidates, because the are either too big, or lie along the borders of the search region.

Figures 3.7 and 3.8 show processes of the iteratively thresholding of the facial areas. Note, that in the real system however, only the search area for the eyes is thresholded, as shown in Figure 3.8. It can be observed from Figure 3.8, that the first accpetable blobs are already the pupils. The white regions in the upper left corner belong to hair and are rejected as candidates, because they are lying along the border of the search-area.

Because the thresholding value is not fixed, this method is able to apply to various lighting conditions and to find the pupils in very differently illuminated faces robustly.

$k = 30$ $k = 40$ ... $k = 80$

Figure 3.7: Use of iterative thresholding to find the eyes. Note: In the real system, only the search area for the eyes has to be thresholded!

$k = 30$ $k = 32$ $k = 34$ - eyes found

Figure 3.8: Iterative thresholding of search window for the eyes

3.2.2 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.

- **Position:** Only blobs that lie completely inside the search window are accepted. Blobs that are connected with the border of the search window usually belong to hair and are therefore rejected.
- **Minimum Size:** Only Blobs that are bigger than 2 pixels were considered. This is done mainly to eliminate blobs that are produced by noise in the input image.
- **Maximum Size of blobs:** The maximum size of the blobs was set to exclude big blobs that sometimes appear at the borders of the search area due to hair, or in case of non-frontal views of the face due to dark background in the image

- **Maximum vertical extension of blobs:** Blobs that are very extended vertically are excluded, knowing that the eye-regions are rather horizontally extended.

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:

- **horizontal distance:** A maximum and minimum horizontal distance is set
- **vertical distance:** A maximum vertical distance is set, assuming, that the head isn't bended to the sides very much initially.
- **Symmetry:** Candidates have to lie approximately symmetrically to the facial symmetric axis. The following symmetry-measure $D(i,j)$ was used:

  $D(i,j) = |cand_i[x] - (w - cand_j[x])|$, 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. A maximum "symmetrie-distance" D_{max} is set. If $D(i,j)$ exceeds D_{max}, the candidate pair (i,j) is rejected.

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

Note that all the parameters to implement the constraints, mentioned above are chosen relative to the actual size of the face. The method is therefore independent of the size of the face in the image and it eliminates the need to normalize the size of the face.

3.3 Searching the Lip Corners

The basic idea to find the lip corners is to stepwise more and more refine the location of the lips, by first looking for the vertical position for the line

between the lips, then finding rough horizontal boundaries of that line in a smaller search region, and finally searching the exact position of the corners.

3.3.1 Predicting Locations of 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. In Figures 3.9 the initial search windows for eyes and lips are marked for two faces. The rectangles mark the predicted positions of the lip corners and the crosses mark the found eyes and lip corners respectively.

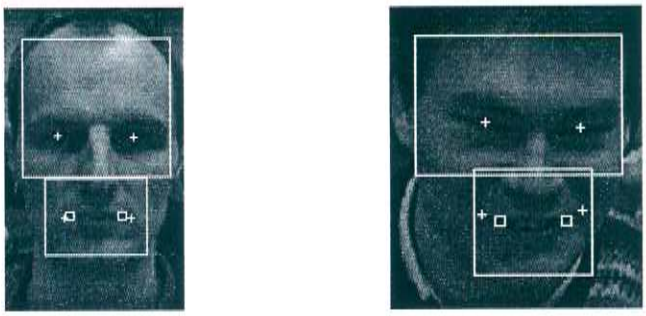
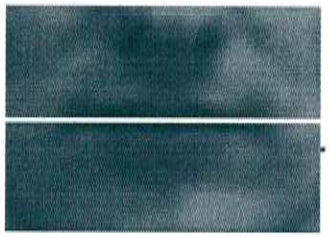


Figure 3.9: Initial search areas for the lips, and found lip corners. The small rectangles mark the predicted positions of the lip corners.

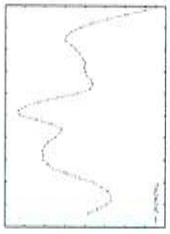
3.3.2 Use of Integral Projections to Locate the Lips

To find the vertical position of the line between the lips, we are looking for a dark and horizontally extended region. This can be found by using a horizontal integral projection P_h of the greyscale image in the search region. The horizontal integral projection P_h is obtained by summing up the greyscale values of the pixels in each row of the search area:

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



search area



horizontal
projection

Figure 3.10: Search of the vertical position of the line between the lips

where $I(x, y)$ is the intensity function of our search window, and W and H are the width and height of the search area, respectively.

Because the line is the darkest horizontally extended structure in the search area, the vertical position of the lip line can now be located where P_h has its global minimum. Figure 3.10 shows the search window for the lip-line and a rotated plot of the corrseponding Projection P_h. The vertical position, where P_h has its global minimum is marked in the image.

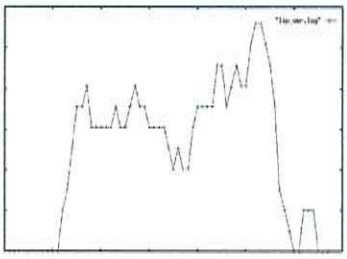
To obtain the horizontal boundaries of the lips, a smaller search area around the estimated vertical position of the line between the lips is extracted, and a horizontal edge operator is applied.

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.

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

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. To obtain the horizontal edge image, we used a horizontal Sobel operator. Figure 3.10 shows the search area around the line between the lips, their horizontal edge image and the corresponding vertical Projection of the horizontal edge image. The left and right borders of the lip-line is marked in the input image.

horizontal edge image



vertical projection

Figure 3.11: Finding horizontal borders of the lips, using a vertical projection of the horizontal edge image of the lips

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

The use of integral projections to extract facial features is for example described in [16] or in Kanade's work on face recognition [21]

3.3.3 Looking for Maximum Contrast along the Line between the Lips

To obtain the final position of the left and right lip corner, we then search along the darkest path (the line between the lips) to the sides for a certain distance d and compute the contrast at each position along the lip-line. Be-

cause there is a sudden change from dark to bright intensities along the line between the lips, where the corners are located, the final lip corner positions can considered to be where the contrast along the lip-line has a maximum. Again, this method is independent from lighting conditions, because no fixed thresholds are used. As long as the consecutive search regions are chosen big enough, this method is only dependent on the parameter d, which has to be chosen, so that the search along the lip-line will go further than the actual lip corners, and not too far to search out of the facial region, which is in fact not a problem with near-frontal or frontal views of the face. As well the parameter d as the other parameters to specify search areas are chosen according to the actual size of the found face.

3.4 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.

Figure 3.12 shows the search area for the nostrils for two different faces with the found positions of the nostrils marked.

Figure 3.12: Search region for nostrils and found nostrils

Chapter 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 adjusted to the size of the facial-regions.

In Figure 4.1 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)

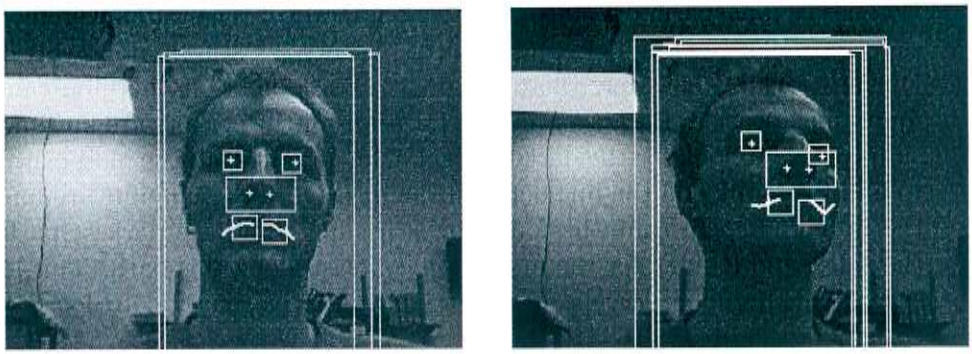


Figure 4.1: Search windows in tracking-mode

4.1 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 [24]), 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 4.1). 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 positions is used.

4.3 Tracking Lip Corners

While for the search of the lip corners, we could apply methods as integral projections and edge detection, this is not anymore feasible for tracking, because these operations take too much time.

These expensive operations in the search mode of the system were necessary, to locate the line between the lips, to start the search along this line. During tracking however, this line, or at least points on this line can be found much easier. Because previous positions of the corners are known, we simply have to look for the darkest pixel in an aera next to the old positions, to find two points on the line between the lips. Once these points are found, the lip corners can again be found by searching along the darkest path to the sides, and looking for the maximum contrast along this path. This local search can be done very fast, and it is an effective method to locate the corners of the lips.

Tracking the lip corners consists of the following steps:

1. Predict the new positions of the lip corners through linear extrapola-

tion over the previous locations

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 lip-line, near the corners

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 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. 4.2 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.

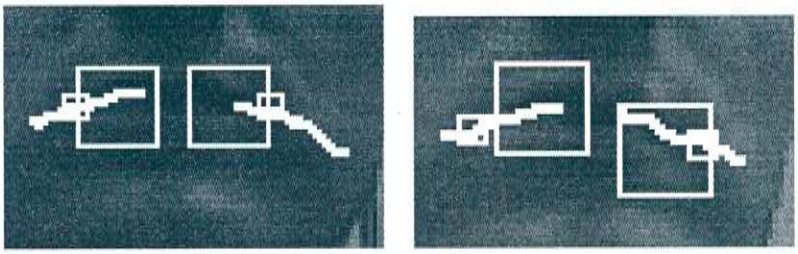


Figure 4.2: Search along the darkest path, starting from the darkest points in the two search windows (see text). Corners are found, where the maximum contrast along the search path is found.

4.4 Tracking Nostrils

Our approach to track the nostrils is basically the same as to search the nostrils, namely to search for two dark regions, that satisfy some geometric constraints. This 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 iterations that are necessary to find the nostrils, and the nostrils usually can be found within very few iterations.

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, and if no nostrils were found, we then continue searching just for one nostril by looking for the darkest pixel in the search window. To decide which of the two nostrils we have found, we use a sample consensus method, which will be described in section 5.1. If only one nostril was found, only the position of this nostril (together with eyes and lip corners) is used to compute the pose. The position of the other nostril can easily predicted in the following frame, making it easier to find both nostrils again.

Figure 4.3 shows some example images, where only one nostril could be found. The found nostril position there are marked by a small cross and the predicted locations of the other nostril are marked by a small box.

Figure 4.3: Predicted positions of second nostril

Chapter 5

Rejection and Prediction of Outliers

Pose estimation is typically an over-constrained problem, i. e., there are usually more correspondences available, than those are necessary to solve the equation systems for computing the pose. Furthermore, automatic finding and tracking of features in an image is error prone, and time after time, there will be outliers in the set of found features.

To avoid degradation of the pose estimation due to outliers in the found feature set, it might therefore be a better idea not to use all the found correspondences, but to choose a best subset of correspondences to compute the pose. Finding and rejecting outliers in the set of correspondences will not only lead to better pose estimations, but also enables the system to recover from tracking failures of single features, by predicting the true positions of outliers, and using the predicted positions to initialize tracking in the next frame.

A good way to choose a best subset, can be done by using a modified version of the RANSAC paradigm (RAndom SAmple Consensus), which was introduced by Fischer & Bolles [5].

5.1 Sample Consensus

The basic idea of the RANSAC paradigm is to find a set of points that are consistent with a single pose, and to reject the remaining outliers. The original RANSAC approach meant to choose random subsets of the data to compute the pose, until a sufficiently good subset was found. However, this is only necessary, if the amount of data is really big, and not all possible subsets can be computed. With only a few feature points, like in our case, it is perfectly feasible to use all possible subsets to compute the pose and find a best subset.

In our case, we have six model points. To compute the pose using the algorithm proposed by DeMenthons & Davis [2], we need at least four correspondences, and the objecet points should preferably be non-coplanar (see section 2). Because the lip corners and eyes lie in one plane, we therefore should have at least one nostril in our feature subset to have a non coplanar set of object points. We furthermore assume, that we have not lost more than one feature in one frame. The considered subsets accordingly were chosen as follows:

1. 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.
2. 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.

Note, that the pseudoinverse matrices, that are necessary to compute the pose, have to be computed in advance for each of the model subsets.

The following steps are now done for all the considered subsets:

1. choose one subset to compute the pose
2. use the estimated pose to back-project the used model points onto the image plane
3. measure the average distance (MSE) of back-projected points and found feature points in the image

Finally, the subset with the least error (distance) is chosen to be the best subset. This subset produced a pose, where the back-projected points matched the actual found points the best. The use of only two subsets in case that we have only found one nostril, forces the system to decide, which of the two nostrils was found.

Our approach is slightly different from the method proposed in [18], where (in step 2) the whole model is projected back onto the image plane, and the number of back-projected points, that lie within a certain circle around the feature-points in the image, is used to find the best subset. This requires however the determination of a parameter r to define the circle around the points in the image, on which the choice of the best subset depends, and which has to be determined empirically.

5.2 Temporal Continuity Tracking

As proposed by Gee & Cipolla [18] the sample consensus approach can be modified, so that the candidate-sets are chosen according to the smoothness of the implied motion, instead of the best match of back-projected model points.

Then step 3) has to be modified as follows:

- measure the smoothness of motion, that is implied by the pose, that was computed with the current subset

Now, the subset, that implied the smoothest motion is chosen as the best subset. Assuming that we have high sampling rates and therefore only little motion from frame to frame, and assuming zero-mean Gaussian distributions for the linear and angular velocities, we can select the pose which maximizes

$$exp\left(-\frac{\|\mathbf{v}\|^2}{2\sigma_l^2}-\frac{\|\omega\|^2}{2\sigma_a^2}\right),$$

where $\mathbf{v}$ is the linear velocity implied by the new pose, ω is the angular velocity, and σ_l and σ_a are the standard deviations of $\|\mathbf{v}\|$ and $\|\omega\|$ respectively.

As can be seen on our results on evaluation image sequences (see. 7), this method always led to better results than using the sample consensus method in our system.

5.3 Prediction of the True Position of Outliers

Once a consistent subset of features is found, the true position of an outlier can be easily predicted in the next frame. The outlier's model point simply has to be projected onto the image plane using the computed pose.

This prediction makes it possible for the system to recover from tracking errors, without going back to initial search, and leads to a much more robust tracking of the feature points. We have also found that the prediction is very helpful to find disappeared nostrils again.

Chapter 6

Recovery from Tracking Failure

Tracking facial features in a camera image is no easy task, and once in while tracking failure will occur. Furthermore, if we want to give the user the possibility to move freely in front of the camera, and maybee even turn his head away from the camera or dissappear at all for a while, failure will necessarily occur at some point.

In order to build a robust gaze tracking system, it is therefore necessary to beeing able to detect tracking failure and to recover therefrom automatically. If this cannot be accomplished, the gaze tracking system would fail completely after the very first failure, and its applicability would be very limited.

6.1 Detection of Failure

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

In our system we use mainly two methods for detecting failure: First, after each feature islocated, 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 (see 5.1) 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, than the actual found features and pose are rejected and failure is considered.

6.2 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 using all the methods that we already described for the initial search in section 3, where we assumed a frontal or near-frontal view of the face, and initialized the search windows for the eyes accordingly. This search will however fail if the person is looking strongly to the left or right. First because the eyes may be outside the search window, and second, because the restrictions on symmetry do no longer apply. If failure occurs during tracking, we cannot assume a frontal view of the face anymore, because failure could have ocurred at any possible rotation of the head. This problem can be solved by initializing the search windows and the geometrical restrictions according to the previously found pose. For example, if failure occured, while the person was looking to the right, we then can 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 6.1 shows the search windows for the 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.

However, if the features cannot be found after a couple of iterations, it is not longer reasonable to assume, that the head is still in a similar pose as the last pose obtained. We therefore only try for a certain small number of frames to find the features with the search windows initialized according

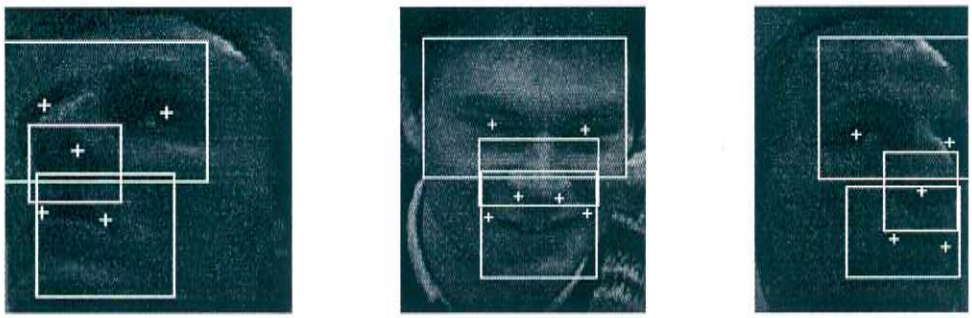


Figure 6.1: Different initialization of search windows according to the previous pose of the head

to the last pose. If, after this number of frames, the features haven't been found again, we then switch in each frame to another initialization of the search windows to find the features again. With this method we usually can recover from tracking failures quite fast.

Chapter 7

Experimental Results

To measure the performance of the feature tracking as well as the pose estimation of the system, we recorded several image sequences of one person to hard disk. In each frame of the sequences, we marked the facial features by hand. To obtain reference poses for each frame, we computed the pose with the pose estimation algorithm, given the hand-labelled positions of the facial features. Then, the gaze tracker was run on the pre-recorded sequences, the facial features were automatically tracked, and the pose computed. These automatically tracked positions and pose parameters were then compared to the results, that were obtained by using the hand-labelled image sequences. While running the gaze tracker on the image sequences, 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 tracker didn't consider the features as lost (tracking failure). However, sometimes the gaze tracker mistakenly considered the features as correctly found, whereas in reality, it lost one or more of the features. The results for these "erroneous" frames added considerably to the average errors for location of the features and rotation results. See also the discussion for the results of "sequence 2" below.

7.1 Different Tracking / Pose Estimation Methods

Three different modes of the gaze tracking system were evaluated:

	sequence 1	sequence 2	sequence 3	sequence 4
# frames	400	300	50	200
size of image	120 x 140	140 x 200	320 x 240	320 x 240
size of face (ca.)	80 x 105	100 x 120	120 x 150	140 x 180

Basic method In this mode, in each frame, all six feature points were used to compute the pose, and no prediction of outliers was done. This method is referenced as *no-pred* in the following tables.

Sample Consensus : A best subset was chosen to compute the pose, according to the best match of back-projected points (see section 5.1), and the positions of outliers were predicted. These predicted positions of outliers were used for evaluation, as actual found positions, like the other non-predicted positions. This method is referenced as *SC-pred* in the following tables.

Temporal Continuity Tracking : A best subset was chosen, according to the smoothest implied motion (see 5.2). Here, also the positions of outliers were predicted and used as found positions for evaluation. This method will be referenced as *TC-pred* in the following tables.

7.2 Test Sequences

We recorded four image sequences with different lengths and different image sizes to hard disk to evaluate the gaze tracking system. Due to slightly different positions of the zoom of the camera and different distances from the face to the camera, the average size of the face in the camera image was different for the sequences.

Figure 7.1 shows some sample images from sequence 2.

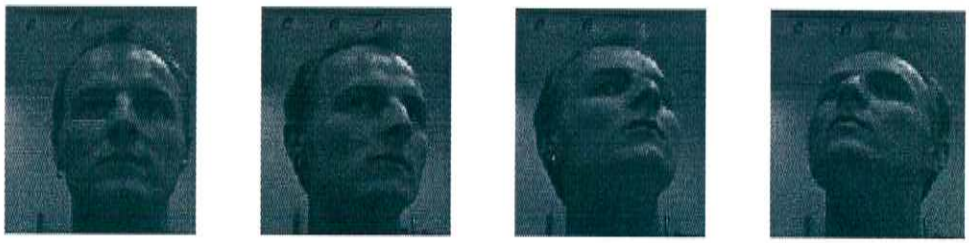


Figure 7.1: Sample images from evaluation sequence 2

	eyes		lip corners		nostrils		all features
method	X err.	Y err.	X err.	Y err	X err.	Y err.	avg. eucl. dist.
TC-pred	3.2	2.5	3.2	2.1	2.0	2.5	4.1
SC-pred	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 7.1: Average location error in pixel for sequence 1

7.3 Feature Tracking Results

Tables 7.1 to 7.4 show the obtained results for locating the features. For each method and each feature, the average distance in x- and in y-direction in pixel is shown, as well as the average euclidian distance (in pixel) for all six feature points.

As can be seen by comparing the average Euclidean distance for all six features, for the three different tracking-methods, the continuity-tracking

	eyes		lip corners		nostrils		all features
method	X err.	Y err.	X err.	Y err	X err.	Y err.	avg. eucl. dist.
TC-pred	5.3	2.6	3.3	2.0	1.9	2.3	4.7
SC-pred	4.0	2.4	3.4	2.0	2.8	2.6	4.6
no-pred	3.6	2.4	3.1	1.8	3.4	2.4	4.5

Table 7.2: Average location error in pixel for sequence 2

	eyes		lip corners		nostrils		all features
method	X err.	Y err.	X err.	Y err	X err.	Y err.	avg. eucl. dist.
TC-pred	4.6	2.1	3.8	2.0	1.8	2.3	4.5
SC-pred	8.0	5.1	9.4	6.8	10.3	5.6	11.5
no-pred	4.4	1.8	3.7	1.5	5.8	3.1	5.5

Table 7.3: Average location error in pixel for sequence 3

	eyes		lip corners		nostrils		all features
method	X err.	Y err.	X err.	Y err	X err.	Y err.	avg. eucl. dist.
TC-pred	4.6	2.8	5.0	2.6	3.2	2.6	5.5
SC-pred	6.2	2.5	4.2	1.7	6.5	2.9	6.6
no-pred	4.2	3.5	4.6	2.1	6.9	2.1	6.4

Table 7.4: Average location error in pixel for sequence 4

method always obtained the best results, and produced the least error. This is also reflected in the accuracy of the pose estimation, as shown below.

7.4 Pose Estimation Results

Tables 7.5 to 7.8 show the average rotation errors around the x-, y- and z-axis, that were obtained with the different methods. Here, the rotation around the x-axis, R_x, corresponds to looking up or down, rotation around the y-axis, R_y, corresponds to looking to the left or right, and rotation around the z-axis, R_z, corresponds to bending the head to the the left or right side.

In correspondence to the results for the accuracy of feature localization as shown above, the continuity-tracking method lead to the best results for all sequences.

Our best results were obtained with sequence 3, where no tracking failure occurred at all. Here we obtained average rotation errors as low as 5.2, 5.0

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

Table 7.5: Average error of rotation in degrees for sequence 1.

method	R_x error	R_y error	R_z error
TC-pred	4.9	8.6	2.7
SC-pred	6.3	9.7	2.5
no-pred	5.1	10.4	2.4

Table 7.6: Average error of rotation in degrees for sequence 2.

method	R_x error	R_y error	R_z error
TC-pred	5.2	5.0	1.4
SC-pred	8.3	10.9	5.5
no-pred	7.6	8.9	1.5

Table 7.7: Average error of rotation in degrees for sequence 3.

method	$R_x$error	R_y error	R_z error
TC-pred	3.7	8.7	2.1
SC-pred	6.4	20.1	2.3
no-pred	5.9	15.1	2.1

Table 7.8: Average error of rotation in degrees for sequence 4.

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

Table 7.9: Sequence 1: Average error of translation in mm.

and 1.4 degrees for R_x, R_y and R_z respectively.

Table 7.9 show the mean errors for the components of the translation vector $\mathbf{T} = (T_x, T_y, T_z)$ in mm for one sequence.. The mean error of components T_x and T_y varied between only 2 and 12 mm for all different sequences and under all different methods. The error for the depth T_x varied between 30 and 100 mm. The mean distance between the face and the camera was about 60 cm and varied roughly between 50 and 70 cm.

7.5 Discussion of Test Sequence "sequence 2"

Table 7.10 shows plots of the rotation parameters R_x, R_y and R_z for 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. Table 7.2 shows 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 immediatly. At around frame 150 serious tracking failure occured, which can easily be seen in the diverging plots for R_y and R_z, and the system detected tracking failure. The tracker than starts searching for the features again, and fully recovers at frame 178. Then the features where accurately tracked again 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 clearly shows the

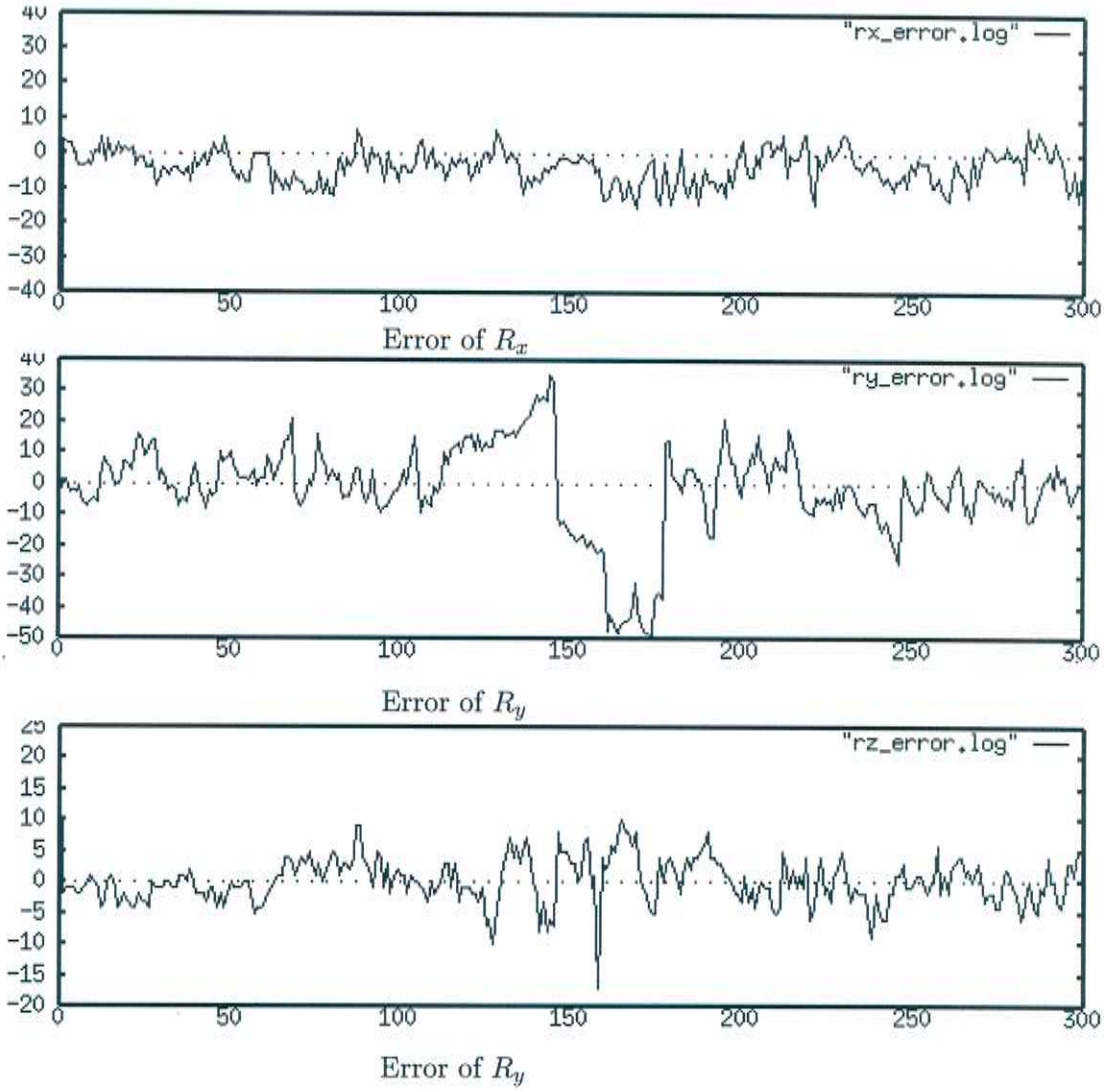
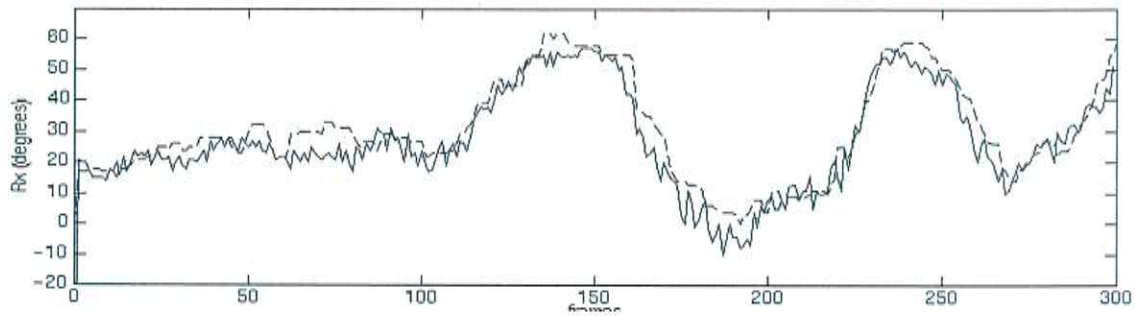
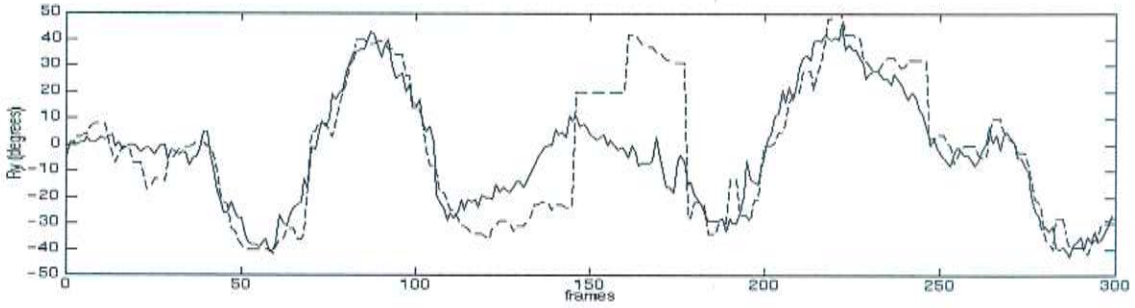


Figure 7.2: Rotation errors

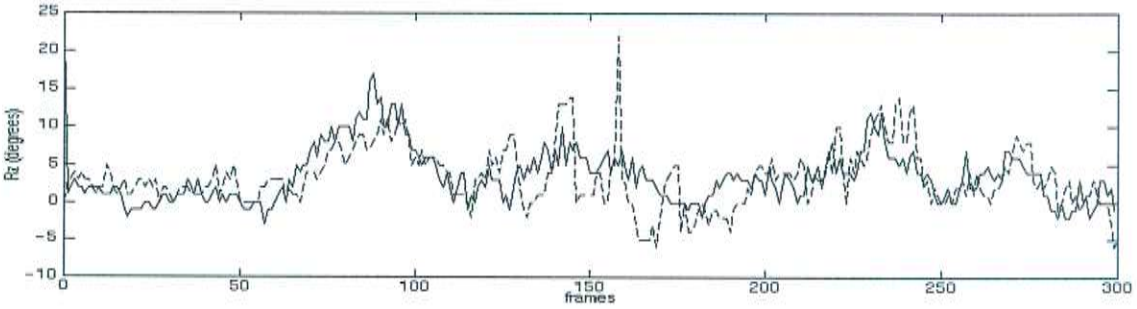
ability of the system to recover from tracking failure.



Rotation around x-axis



Rotation around y-axis



Rotation around z-axis

Table 7.10: These figures show the estimated rotation angles computed with the hand-labelled features (solid line) and with automatically tracked features (dashed line) for image sequence "sequence 2".

Chapter 8

A Multimodal Interface to control a Panorama Image Viewer

In order to show the applicability of our gaze tracking system, we developed a multimodal interface to view panorama images. The image viewer application usually allows the user to scroll through 360 degree panorama images by using the mouse and to zoom in and out using the keyboard. We used the gaze tracker to control scrolling through the panorama images, and voice-commands to control the zoom. The interface receives parameters describing the rotation of the 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. Both the interface and the image viewer are running on a PC, and communication is done via sockets.

With such an interface, a user can fully control the panorama image viewer without using his/her hands. He can scroll through the panorama images by looking to the left and right or up and down, and he can control the zoom by speaking the commands "zoom in" or "zoom out".

Chapter 9

Conclusions and Further Directions

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.

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 computes the pose by finding correspondences between points in a model of a head and points in the camera image. The system automatically finds and tracks the facial feature points in the image and is able to recover from tracking failure. The usefulness of the gaze tracking system has been shown by using it to control scrolling in a panorama image viewer.

To obtain a more precise gaze estimation it should be useful to estimate the eye-gaze direction on top of the head pose, whenever high-resolution images of the eye region are available, i.e, when the user is close to the camera. To extract the eye regions for further examination, the positions of the pupils obtained by our feature tracker can be used.

In order to beeing able to adjust the image resolution of the facial region, active camera zoom control will be helpful. This will however introduce

some camera calibration difficulties that have to be dealt with.

Active control of the panning and tilting mechanism of the camera should be useful in order to beeing able to track a person's head for example when the person is walking around, or when the camera has zoomed in on the facial region and therefore the field of view of the camera is rather small.

One disadvantage of the current system is, that using just one general model of the head could lead to inaccurate pose results. One solution might be to adapt or calibrate the head model for new users. This could be done by taking a frontal and a lateral image of the user's head, extracting the facial feature points, and initializing the 3D model accordingly.

Bibliography

[1] D. F. Dementhon. *De la Vision Artificielle à la Réalité Synthétique: Systeméme d'interaction avec un ordinateur utilisant l'analyse d'images vidéo.* PhD thesis, Université Joseph Fourier - Grenoble I, Laboratoire TIMC/IMAG, October 1993.

[2] 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.

[3] Oberkampf, D. Dementhon, D.F. and Davis, L.S. 1993. Iterative Pose Estimation using Coplanar Feature Points. *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 626-627, New York 1993, full version: Center for Automation Research Technical Report CAR-TR-677, University of Maryland.

[4] O. D. Faugeras. *Three Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, Boston, 1993.

[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. B. Haralick, H. Joo, C-N. Lee, X. Zhuang, V.G. Vaidya, and M. B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6): 1426-1445, 1989.

[7] 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.

[8] R. Horaud, S. Christy, and F. Dornaika. Object Pose: The Link between Weak Perspective, Para Perspective, and Full Perspective. Rapport de recherche No. 2356, INRIA, Grenoble, 1994.

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

[10] T. Oshima, H. Yamamoto, H. Tamura. Gaze-directed adaptive rendering for interacting with virtual space.

[11] T. Q. Phong, R. Horaud, A. Yassine, and D. T. Pham. Object pose from 2-D to 3-D point and line correspondences. *International Journal of Computer Vision*, 13, 1994.

[12] Roberts, L.G. Machine Perception of Three-Dimensional Solids. *Optical and Electrooptical Information Processing*, J. Tippet et al., eds, 1965, MIT Press.

[13] Sutherland, I.E. Three-Dimensional Input by Tablet. *Proceedings of the IEEE*, 1974, Vol. 62, pp. 453-461.

[14] Tsai, R.Y. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. *IEEE J. Robotics and Automation*, 1987, Vol. 3, pp.323-344.

[15] 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.

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

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

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

[19] D. Gennery. Visual tracking of known three-dimensional objects. *International Journal of Computer Vision*, 7(3): 243-270, 1992.

[20] C. Harris. Tracking with rigid models. In A. Blake and A. Yuille, editors, *Active Vision*, pages 59-73. MIT Press, Cambridge MA, 1992.

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

[22] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of computer Vision*, 8(2):113-122, 1992.

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

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

[25] L. Stringa. Eyes Detection for Face Recognition. *Applied Artificial Intelligence*, 7:365-382, 1993.

[26] A. Azarbayejani, T. Starner, B. Horowitz, A. Pentland. Visually controlled Graphics. IEEE PAMI 15(6) , June 1993.

[27] 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.