Face Detection and Pose Estimation Using Part-Based Models

Markus Heukelom

Master's Thesis

Artificial Intelligence,
Multimodel Intelligent Systems

July 2005

Supervised by
Dr. ir. B.J.A. Kröse
Dr. J.J. Verbeek

Intelligent Autonomous Systems
University of Amsterdam

UNIVERSITEIT VAN AMSTERDAM

IAS
intelligent autonomous systems

This thesis, ”Face Detection and Pose Estimation Using Part-Based Models”, is submitted in partial fulfilment to the requirements for the degree **Master of Science** in Artificial Intelligence at the University of Amsterdam.

Date:

Author’s signature: Supervisor’s signature:

Contents

Chapter 1

Introduction

The topic of this thesis is face detection and face pose estimation. We propose a method to detect the face and, simultaneously, estimate its pose from images acquired with a calibrated camera. The approach is novel in the way it combines techniques from computer vision and recent developments in object detection using planar models. An overview of the system is presented in Fig. 1.2. The method is tested on different data sets, providing a proof of concept of the applicability of the approach.

1.1 Application domain

Considering possible applications of face pose estimation is not the main concern of this thesis, and so we suffice to say that knowing the pose of the face is important in many applications, including human-computer interfaces, video database indexing and face recognition. Philips Research has recently demonstrated [16] a new consumer product named iCat (Fig. 1.1), which provides natural human-like interaction intended to help in daily tasks such as sending messages, accessing daily information, selecting your favorite music, photos and video or even guarding your home. Being able to locate and estimate a person's face can greatly improve the feel of human-like interaction, for example by turning the iCat's head towards the person it is interacting with, and only obeying to voice commands when looked at.

Figure 1.1: Philips' Icat

1.2 Concern of the thesis

Before the pose of a face can be estimated, the face region needs to be detected in an image. This poses a problem, because successful face detectors, such as presented in [17], [20], [21], are not invariant to rotation. As a result we cannot simply apply such a detector to find the face region and subsequently estimate its pose (provided we have a method to do this), because faces under rotation will simply not be detected. As a work-around, we could learn a face detector for a discrete set of poses, which is done by Viola *et al.* [22] as an extension to the face detector developed by the same authors in [21]. Although the speed and detection rate is unprecedented, this approach is unsatisfactory in the sense that it effectively treats different views of the same object as different object classes, which has a somewhat unnatural appeal and, moreover, requires separate training data for each 'orientation' class. Another solution would be to use a highly controlled environment in order to make the detection of the face under varying rotation much easier. For example, in the work by Ji *et al.* [8] a special (near-infrared) illuminator is used so that the pupils are easily detected, which are used to track the face. The pose of the face is subsequently estimated from an ellipse fitted on the face boundary. The disadvantage of this approach is that depending on a controlled environment makes the applicability of the method much more constrained.
We therefore propose a different method, based on recent developments in object recognition by parts. In recognition by parts, an object is composed of *parts* and *shape*, where *parts* are image patches, detected by appropriate detectors, and *shape* describes spatial relation of the parts. The method we propose makes use of the Sift detector developed by Lowe [13], and a planar spatial part model, which is used to both detect and estimate the face pose. Our approach is inspired by recent developments in object class recognition, such as the approaches presented by Fergus *et al.* [3], Helmer *et al.* [7], Weber [23], where the shape is modelled in a way that is invariant to object scale and object location. By extending this to affine invariance, we are able to detect faces under any rotation and scale, which makes it possible to integrate face detection and pose estimation. For the pose estimation we require the camera calibration matrix to be known.

1.3 Outline of the thesis

We start (chapter two) with describing how the estimation of the face pose is carried out when model - image point correspondences are known. The pose consist of a three dimensional rotation matrix and translation vector relating the face coordinate frame to the camera coordinate frame. Because we use a planar model of points, this poses a minor problem as conventional techniques from computer vision require correspondences between the image and a *three dimensional* object model in order to retrieve the objects pose. A planar model of points and their corresponding image points are related by a *homography*, from which we use to estimate an initial pose. The Levenberg-Marquardt iterative optimization routine is employed to improve the initial pose.
Chapter three deals with facial part acquisition, explaining the advantage of the Sift detector for this purpose and our investigations to test its applicability to our problem. The appearance of the Sift feature in preselected areas of the face is modelled using a mixture of Gaussians (MoG). A Bayesian classifier is used to obtain a set of candidate features for each part, when provided with a new image.

Chapter four deals with finding the best candidate from the candidates sets for each part. Each such matching is called a hypothesis. The best hypothesis is found by employing shape constraints implemented by means of a spatial model. Because searching through the full hypothesis space is intractable, heuristic search in the form of A* is applied, which greatly reduces the number of explored hypotheses. In the experiments we put all the pieces together, to show how faces in realistic images are detected and how the face pose is estimated when the camera calibration matrix is known.
Finally, chapter five concludes the thesis and discusses the advantages and disadvantages of the proposed method, together with some directions for future research to improve the performance of the method.

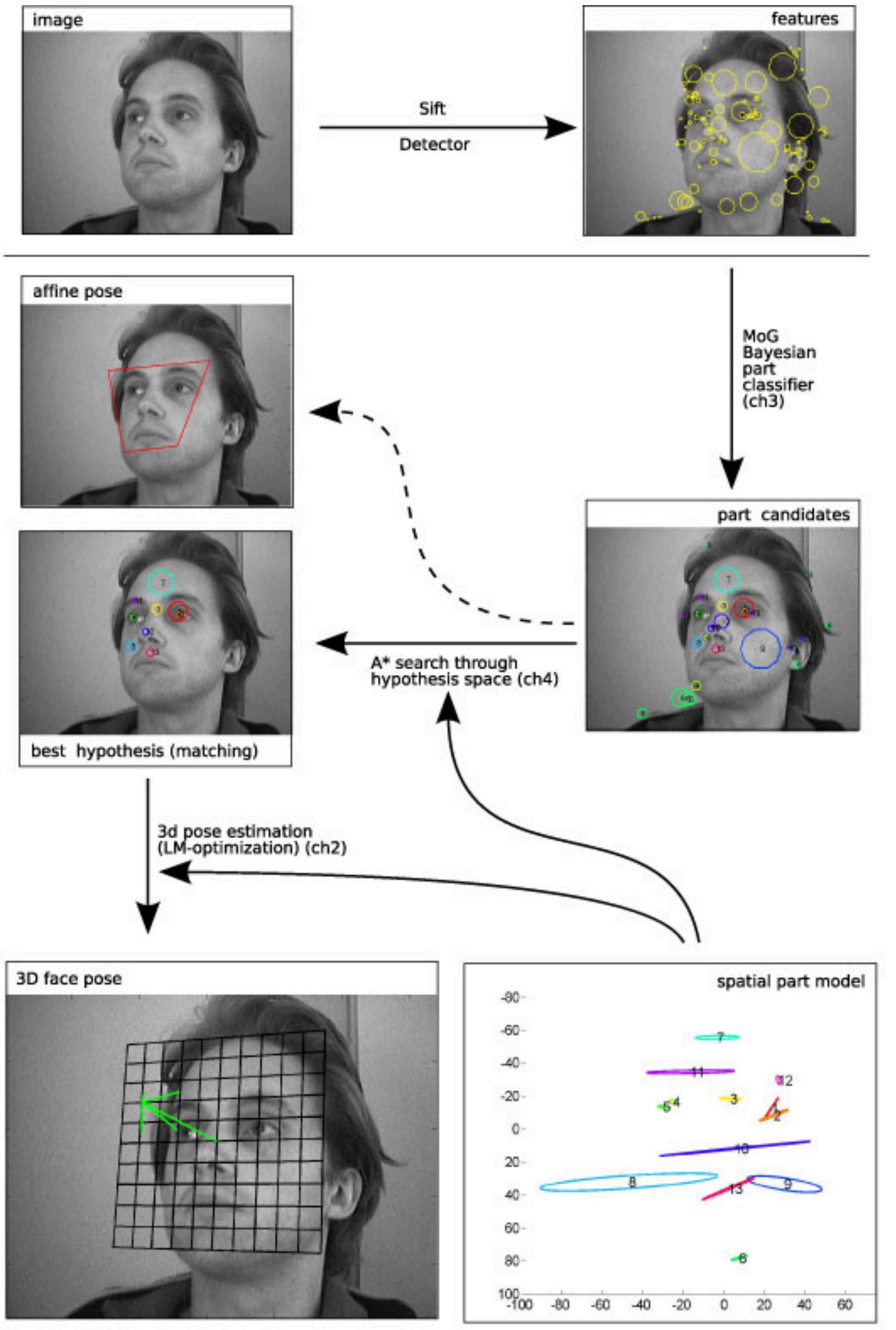


Figure 1.2: Architectural overview of the proposed method.

Chapter 2

3D face pose estimation from a planar model

This chapter treats 3D face pose estimation using a planar model of points describing the relative position of face features. The aim is to estimate a three dimensional rotation matrix R and a translation vector $\mathbf{t}$, indicating the pose of the face with respect to the camera. In short, estimation of the pose consists of two phases. First, an initial estimate is computed from a homography, which describes the mapping between the planar model points and image points. The second phase consists of optimizing the initial estimate of the pose using the Levenberg-Marquardt (LM) nonlinear optimization routine. Estimating the pose of a planar model from a homography is well known, for example see [25]. The approach we used to optimize the initial estimate, is (to the author's knowledge) not reported in literature, however, using LM to solve computer vision related problems is extensively used and often reported, such as in [6], from which many of the ideas in this chapter stem. In this respect the described method is not a novel approach, but merely a report of its application to the solution of the problem we seek to solve. The method is tested on synthetic data to get an idea of the accuracy we can expect, and on some real-world images to provide a qualitative measure on the applicability of the method in real-life.

2.1 Relation between the model plane and its image

We start we some notation. A 2D point is denoted by $\mathbf{x} = [x, y]^\top$. A 3D point is denoted by $\mathbf{X} = [x, y, z]^\top$. We use $\tilde{\mathbf{x}}$ to denote the homogeneous form of $\mathbf{x}$ which has an additional last element $\tilde{w}$: $\tilde{\mathbf{x}} = [\tilde{x}, \tilde{y}, \tilde{w}]^\top$, and $\tilde{\mathbf{X}} = [\tilde{x}, \tilde{y}, \tilde{z}, \tilde{w}]^\top$. The cartesian form is retrieved from its homogeneous form by dividing through $\tilde{w}$ and removing the last element: $\mathbf{x} = [\tilde{x}/\tilde{w}, \tilde{y}/\tilde{w}]^\top$. The homogeneous form is constructed from the cartesian form by setting $\tilde{w} = 1$ as last element: $\tilde{\mathbf{x}} = \lambda[x, y, 1]^\top$; we may chose any scale $\lambda \neq 0$ due to the fact that homogeneous coordinates are defined up to scale.

The *pinhole* camera model describes the relation between a 3D point $\mathbf{X}$ and its projection onto the camera image plane, $\mathbf{x}$:

$$\mathbf{x} = \begin{bmatrix} \tilde{x}/\tilde{w} \\ \tilde{y}/\tilde{w} \end{bmatrix}, \quad \tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = KP\tilde{\mathbf{X}}, \tag{2.1}$$

where matrix $P = [R \quad \mathbf{t}]$ is composed of the rotation and translation, transforming the position of $\mathbf{X}$ in world coordinate system to the camera coordinate system. Because let the world coordinate frame coincide with the camera coordinate frame, P expresses the pose with respect the camera. Matrix K models the pinhole camera and is called the camera (intrinsic) calibration matrix, given by

$$K = \begin{bmatrix} s_x & k & x_0 \\ 0 & s_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

in which (x_0, y_0) are the coordinates of the principal point (which is the intersection of the camera's optical axis with the image plane), s_x and s_y the scale factors in image x and y axes, and k the skew factor. Throughout this chapter, we assume that K is known. We work with a left-handed camera frame, meaning that the camera 'looks' into the negative Z-axis.

As mentioned in the introduction, our aim is to estimate P in Eq. (2.1) using a *planar model*. Without loss of generality, we align the model plane parallel with the xy-plane of the camera coordinate system at $z = 0$, writing $\tilde{\mathbf{X}}_n = [u, v, 0, 1]$ for each model point $\mathbf{x}_n = [u, v]^\top$. Let $\mathbf{r}_i$ denote the ith column of R, a model point $\mathbf{x} = [u, v]^\top$ then relates to an image point $\mathbf{x}' = [x, y]^\top$ by:

$$\tilde{\mathbf{x}}' = \underbrace{K \,[\, \mathbf{r}_1 \; \mathbf{r}_2 \; \mathbf{t} \,]}_{H} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H\tilde{\mathbf{x}}, \tag{2.3}$$

which follows from Eq. (2.1) due to the zero in $\tilde{\mathbf{X}}$ at the third entry. Note that this equation uses the homogeneous form of both the model and image point. The difference with Eq. (2.1) is that $P = [\mathbf{r}_1 \; \mathbf{r}_2 \; \mathbf{t}]$ is now a 3×3 matrix instead of 3×4. The calibration matrix K is also of size 3×3, so the relation between the model plane and the image plane is a 3×3 matrix, which we denote with H. The 'output' of Eq. (2.3) is in homogeneous form, however that we observe cartesian coordinates: $\mathbf{x}'$. Dividing by $\tilde{w} = \mathbf{h}_3^\top \tilde{\mathbf{x}}$, where $\mathbf{h}_i^\top$ denotes the ith row of H, we obtain the cartesian coordinates:

$$\mathbf{x}' = \frac{1}{\mathbf{h}_3^\top \tilde{\mathbf{x}}} \begin{bmatrix} \mathbf{h}_1^\top \tilde{\mathbf{x}} \\ \mathbf{h}_2^\top \tilde{\mathbf{x}} \end{bmatrix}, \quad \text{with } H = \; [\mathbf{h}_1 \; \mathbf{h}_2 \; \mathbf{h}_3 \,]^\top = K[\mathbf{r}_1 \; \mathbf{r}_2 \; \mathbf{t}]. \tag{2.4}$$

Note that although $\mathbf{r}_3$ is removed in this equation it can be found unambiguously from the cross product of $\mathbf{r}_1$ and $\mathbf{r}_2$, due to the required orthogonality between the columns of a rotation matrix.

2.2 Problem statement

Given we have N model - image point correspondences, we want to retrieve P using Eq. (2.4). Because of noise we can, however, not expect that there is a pose for which this relation holds exactly for all correspondence pairs. This noise may stem from our particular acquisition process, such as the error introduced when obtaining the image points by hand, or from inaccuracy of the detector we might employ to automatically obtain these points, as is done in chapter three. The noise can also be a result of inaccuracy of our model. For example, using the pinhole model, we observe error in the position of the image points due to lens distortion as noise, although is a result of the lack of expressing lens distorting using the pinhole model.

We therefore define the true pose to be the one which minimizes the distance between the expected position given by Eq. (2.4) and observed position $\mathbf{y}_n$. We must also ensure that $K^{-1}\mathbf{h}_i$ are valid columns of a rotation matrix, ie. they should have unit norm and their inner product should be zero. The problem for estimating the pose using a planar model is therefore stated by:

$$\operatorname*{argmin}_{R,\mathbf{t}} f(R,\mathbf{t}), \quad f(R,\mathbf{t}) = \sum_n \left\| \mathbf{y}_n - \frac{1}{\mathbf{h}_3^\top \tilde{\mathbf{x}}_n} \begin{bmatrix} \mathbf{h}_1^\top \tilde{\mathbf{x}}_n \\ \mathbf{h}_2^\top \tilde{\mathbf{x}}_n \end{bmatrix} \right\|^2, \tag{2.5}$$

$$\text{such that} \quad i,j \in \{1,2\} : \quad [K^{-1}\mathbf{h}_i]^\top [K^{-1}\mathbf{h}_j] = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, \tag{2.6}$$

where $||.||^2$ denotes the squared Euclidian distance, and $H = [\mathbf{h}_1 \ \mathbf{h}_2 \ \mathbf{h}_3\ ]^\top = K[\mathbf{r}_1 \ \mathbf{r}_2 \ \mathbf{t}]$.

To the best of the authors knowledge, there is no closed form solution to this problem. Therefore, we take a different approach. First, we estimate a general 3×3 matrix H, that is, we drop the constraints on the first two columns of this matrix, Eq. (2.6). A general, nonsingular, 3×3 matrix H is known as a homography, so this reduces the problem to finding a *homography* between the image and model points. From the homography we then retrieve an (initial) estimate of the pose $P = [R \quad \mathbf{t}]$. Because the constraints are dropped in estimating the initial pose, there is reason to believe that the found pose is not the optimal solution for our problem statement. For this reason, the second step consists of optimizing the initial estimated using the Levenberg-Marquardt (LM) optimization scheme.

The rest of this chapter is organized as follows. We start with describing how the homography between the model and image points is computed, and how the initial pose is subsequently retrieved from the homography. In Section 2.4, we give a short review of the LM procedure, and proceed with explaining how to the routine is applied to the optimization of the initial pose. The accuracy of the methods (before and after optimization) is tested experimentally and discussed in Section 2.5. We end with a conclusion in Section 2.6.

2.3 Initial estimate of the pose

As said, the initial pose is obtained by estimating a general 3×3 linear map between the point correspondence. A non-singular, 3×3 matrix operating on homogeneous planar coordinates, is called a homography. The difference between a homography and H is that the first has eight degrees of freedom (dof) and the latter only six. A 3×3 matrix has nine entries, but because homogeneous coordinates are defined up to scale, the homography is also defined up to scale, and so it has nine dof minus one for scale. H has six dof, because it is constructed from K (which is assumed to be known and therefore has zero dof), $\mathbf{r}_1, \mathbf{r}_2$ and $\mathbf{t}$. $\mathbf{r}_1$ and $\mathbf{r}_2$ fully specify R, which is specified by three angles, so together with the three entries of $\mathbf{t}$, H has 6 dof.

2.3.1 Computing the homography H

A homography between planar point correspondences can be computed as follows. Suppose we have N image - model correspondences, $\mathbf{y}_n = [x_n, y_n]^\top$ and $\mathbf{x}_n = [u_n, v_n]^\top$ respectively. We want to satisfy the relation

$$\mathbf{y}_n = \frac{1}{\mathbf{h}_3^\top \tilde{\mathbf{x}}_n} \begin{bmatrix} \mathbf{h}_1^\top \tilde{\mathbf{x}}_n \\ \mathbf{h}_2^\top \tilde{\mathbf{x}}_n \end{bmatrix} \tag{2.7}$$

as good as possible for each point pair. We may rewrite this equation as

$$\begin{bmatrix} \mathbf{h}_1^\top \tilde{\mathbf{x}}_n \\ \mathbf{h}_2^\top \tilde{\mathbf{x}}_n \end{bmatrix} - \begin{bmatrix} x_n \mathbf{h}_3^\top \tilde{\mathbf{x}}_n \\ y_n \mathbf{h}_3^\top \tilde{\mathbf{x}}_n \end{bmatrix} = 0.$$

Writing H as a vector $\mathbf{h} = [\mathbf{h}_1^\top \quad \mathbf{h}_2^\top \quad \mathbf{h}_3^\top]^\top$, allows us to obtain a system of $2N$ linear equations and nine unknowns, $A\mathbf{h} = 0$, with

$$A = \begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1 x_1 & -v_1 x_1 & -x_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1 y_1 & -v_1 y_1 & -y_1 \\ & \vdots & & & \vdots & & & \vdots & \\ u_N & v_N & 1 & 0 & 0 & 0 & -u_N x_N & -v_N x_N & -x_N \\ 0 & 0 & 0 & u_N & v_N & 1 & -u_N y_N & -v_N y_N & -y_N \end{bmatrix}. \tag{2.8}$$

We seek $\mathbf{h}$ which minimizes $||A\mathbf{h}||$. The trivial solution $\mathbf{h} = 0$ to the system $A\mathbf{h} = 0$ is of course of no interest. Because H, and so $\mathbf{h}$ is defined up to scale, we need to minimize $||A\mathbf{h}||$ under the constraint $||\mathbf{h}|| = 1$. The solution to this problem is well known from linear algebra theory [2]: $\mathbf{h}$ is the last column of V corresponding to the smallest singular value, where $A = UDV^\top$ is the singular value decomposition of A.

2.3.2 Retrieving the pose from the homography H

Let $H = [\ \mathbf{h}_1 \quad \mathbf{h}_2 \quad \mathbf{h}_3\ ]$ be a full (that is 8 dof) homography. In order to obtain the initial pose, we set

$$\begin{aligned} \mathbf{r}_1 &= \lambda_1 K^{-1} \mathbf{h}_1 \\ \mathbf{r}_2 &= \lambda_2 K^{-1} \mathbf{h}_2 \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} &= \lambda_3 K^{-1} \mathbf{h}_3 \end{aligned} \tag{2.9}$$

with $\lambda_1 = 1/||K^{-1}\mathbf{h}_1||$, $\lambda_2 = 1/||K^{-1}\mathbf{h}_2||$ due to the required unit-length of the columns of matrix R. For λ_3 we take the average between the inverted norms of $\mathbf{r}_1$, $\mathbf{r}_2$: $\lambda_3 = (\lambda_1 + \lambda_2)/2$. Due

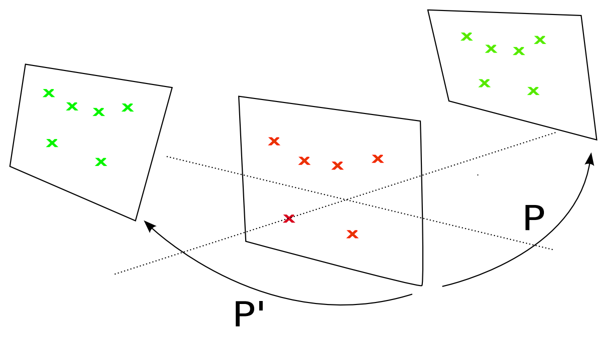


Figure 2.1: Image points (red) can be the result of two model positions (green).

```
Objective  Estimate initial pose $P$

Input  model points $\mathbf{x}_n$, image points $\mathbf{y}_n$, camera calibration matrix $K$.

  1. Form matrix $A$ using $\mathbf{x}_n$, $\mathbf{y}_n$ (Eq. (2.8)).
  2. Form $H$ from the last column of $V$, where $A = UDV^T$ is the svd of $A$. Set $H' = -H$.
  3. Determine $P$ and $P'$ using resp. $H$ and $H'$ (Eq. (2.9)).
  4. if $P_{34} \geq 0$ return $P$, else return $P'$.
```

Figure 2.2: Algorithm to estimate an initial pose P using a planar model.

to the unconstrained form of H and because of noise, the so-computed matrix $Q = [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3]$ does not in general have orthogonality between its columns, which is a required property of a rotation matrix. The solution is to find the best rotation matrix R to approximate the estimated matrix Q. The optimal solution is achieved by setting

$$R = UV^T,$$

where $Q = UDV^T$ is the singular value decomposition of Q. A derivation of this result can be found in [25].
Because H is defined up to scale, $-H$ is also a solution of (2.4). As a result we also have two possible extrinsic matrices P and P'. A geometric interpretation of these two solutions is given in Fig. 2.1. To determine which of these is the correct one, we can simply examine the value of the z-coordinate of the translation vector $\mathbf{t}$. Since the face can only be seen by the camera if it was in front of it, z should be less than zero (remember that we are looking into the negative z-axis). Fig. 2.2 summarizes the algorithm.

2.4 Optimizing the initial pose

As mentioned, the Levenberg-Marquardt (LM) optimization procedure is used to optimize the initial estimate of the face pose. We start with a brief discussion of LM, followed by its application to our optimization problem.

The LM algorithm is a optimization scheme, used to find a set of optimal parameter values, represented by vector $\mathbf{p}$, for a (nonlinear) function $\mathbf{f}(\mathbf{p})$ such that a desired output value $\mathbf{f}_{des}$ is best approximated. Note that in our case $f_{des} = 0$ (we have a scalar valued function). As a means to an end, the optimization is performed by updating an initial set of parameter values iteratively by adding an increment to $\mathbf{p}$ at each step: $\mathbf{p} \leftarrow \mathbf{p} + \triangle_{\mathbf{p}}$ such that the error function $||\boldsymbol{\epsilon}||^2 = ||\mathbf{f}(\mathbf{p}) - \mathbf{f}_{des}||^2$ is decreased. The increment vector $\triangle_{\mathbf{p}}$ is computed using an *update equation*. In order to explain how LM works, we first consider two other update equations: Gauss-Newton and gradient-descent, to which the LM update equation is closely related. A derivation of these update equations, which explains why they lead to a reduction of the error function, is given in [6]. Let J denote a linear mapping represented by the Jacobian matrix $J = \frac{\partial \mathbf{f}}{\partial \mathbf{p}}$ evaluated at $\mathbf{p}$, and $\boldsymbol{\epsilon} = \mathbf{f}(\mathbf{p}) - \mathbf{f}_{des}$ the error of the current set of parameter values. The update equation for Gauss-Newton and gradient descent are given by:

Gauss-Newton :

$$\triangle_{\mathbf{p}} = -(J^\top J)^{-1} J^\top \boldsymbol{\epsilon}. \tag{2.10}$$

The main property of the this update equation is that if the error function $||\boldsymbol{\epsilon}||^2$ is close to being quadratic in $\mathbf{p}$, this method will converge fast to the minimum value. Because it is based on the assumption that $\mathbf{f}$ is locally linear, it can lead to an increase in the error when this assumption is not met.

gradient-descent :

$$\triangle_{\mathbf{p}} = -\lambda J^\top \boldsymbol{\epsilon}. \tag{2.11}$$

The negative gradient $-J^\top \boldsymbol{\epsilon}$ is the direction of the most rapid decrease of the cost function. The strategy of minimizing $\boldsymbol{\epsilon}$ by moving iteratively into the negative gradient direction, is therefore known as gradient descent. The parameter λ controls the length of the step. λ is typically decreased if the error is not reduced, and vice versa. In this way, the method guarantees that the error is never increased. However, gradient descent by itself is not a very good minimization strategy, typically characterized by slow convergence due to zig-zagging.

The Levenberg-Marquardt method is essentially a Gauss-Newton method that transitions smoothly to gradient descent when the going is tough. It uses the property of fast convergence for Gauss-Newton, and if this fails, automatically falls back to gradient-descent. In order to do this, it uses the following update equation:

Levenberg-Marquardt :

$$\triangle_{\mathbf{p}} = -(J^\top J + \lambda I)^{-1} J^\top \epsilon. \tag{2.12}$$

If $\mathbf{p} \leftarrow \mathbf{p} + \triangle_{\mathbf{p}}$ doesn't lead to a reduction in the error, λ is increased by a factor of 10 and the step is repeated, until the error is decreased. As soon as the error decreases, λ is decreased as well by a factor of 10, and the next step done. This process is repeated until the percentile reduction in error is less than some constant ρ.
To understand the reasoning behind this method, consider what happens for different values of λ. When λ is very small, the method is essentially the same as Gauss-Newton. As a result, if the error function $||\boldsymbol{\epsilon}||^2$ is close to being quadratic around $\mathbf{p}$, this method will converge fast

to the minimum value. On the other hand when λ is large, we get $(J^\top J + \lambda I) \approx \lambda I$, and so $\triangle_\mathbf{p} \approx -\lambda^{-1} J^\top \boldsymbol{\epsilon}$, which is approximately the one given by the gradient-descent method. Thus, the LM algorithm moves seamlessly between Gauss-Newton iteration, which causes rapid convergence in the neighborhood of the solution, and a gradient descent approach, which will guarantee a decrease in the cost function when the going is difficult. As λ becomes larger and larger, the length of the increment step $\triangle_\mathbf{p}$ decreases and eventually it will lead to a decrease of the cost function $||\boldsymbol{\epsilon}||^2$.

However, no matter how fast any nonlinear optimization scheme is, is has the clear disadvantage that it does not guarantee to find the globally optimal solution, if it finds a better solution at all. The solution very much depends on the initial estimate, and the LM optimization routine can get stuck easily in a local optimum. To test whether the optimization by the LM algorithm actually leads to a significant improvement in the pose estimate, experiments were conducted on synthetic data to measure this behavior, which are described in Section 2.5.

In order to apply LM to our optimization problem, we use the function as defined in Eq. (2.5) with desired value $f_{des} = 0$, but with a different parameterization to reformulate the constraints. Any 3D rotation matrix R can be constructed using three angles, describing the (ordered) rotation around three different axes. This is called a Euler angles representation of a rotation matrix. In particular, we form R by rotation of α degrees around the x-axis, followed by β degrees around the y-axis and finally by γ degrees around the z-axis:

$$R(\alpha,\beta,\gamma) = \begin{bmatrix} \mathrm{c}\gamma\,\mathrm{c}\beta & -\mathrm{s}\gamma\,\mathrm{c}\alpha + \mathrm{c}\gamma\,\mathrm{s}\beta\,\mathrm{s}\alpha & \mathrm{s}\gamma\,\mathrm{s}\alpha + \mathrm{c}\gamma\,\mathrm{s}\beta\,\mathrm{c}\alpha \\ \mathrm{s}\gamma\,\mathrm{c}\beta & \mathrm{c}\gamma\,\mathrm{c}\alpha + \mathrm{s}\gamma\,\mathrm{s}\beta\,\mathrm{s}\alpha & -\mathrm{c}\gamma\,\mathrm{s}\alpha + \mathrm{s}\gamma\,\mathrm{s}\beta\,\mathrm{c}\alpha \\ -\mathrm{s}\beta & \mathrm{c}\beta\,\mathrm{s}\alpha & \mathrm{c}\beta\,\mathrm{c}\alpha \end{bmatrix}, \qquad (2.13)$$

where c, s stand for cos and sin, respectively. Together with three parameters for the translation vector $\mathbf{t}$, we have six parameters in total: $\mathbf{p} = [\alpha,\beta,\gamma,t_x,t_y,t_z]^\top$. This means that $f : \mathcal{R}^6 \to \mathcal{R}^1$.

From the initial estimate of the pose, however, we get the rotation matrix R only. This means we need to extract the angles from R, which can be done in the following way:

$$\alpha = \mathrm{atan}\frac{R_{32}}{R_{33}} \quad \beta = \mathrm{asin}(-R_{31}) \quad \gamma = \mathrm{atan}\frac{R_{21}}{R_{11}}. \qquad (2.14)$$

The Jacobian of f was formed by numerical differentiation. We used a value of $\rho = 0.1\%$. The initial value for λ was found empirically. Fig. 2.3 summarizes the LM algorithm applied to our problem.

2.5 Experiments

To obtain insight in the performance of the method described in the last section we performed several experiments. First of all, we want to confirm whether the LM optimization really improves the pose estimation. Second, we are interested in the performance with respect to the amount of noise in the image points and with respect to the amount of points in the model. We expect that the pose estimation is significantly better when more model points are used, and significantly worse when the amount of noise is increased. We created synthetic test data to measure this performance, results are in Section 2.5.1. In Section 2.5.2, we picked

```
Objective  Optimize initial pose $P$, using Levenberg-Marquardt optimization

Input  model points $\mathbf{x}_n$, image points $\mathbf{y}_n$, camera calibration matrix $K$, initial pose
       $P_0$, stop condition $\rho$.

  1. Form initial parameter vector $\mathbf{p} = [\alpha, \beta, \gamma, t_x, t_y, t_z]$ using Eq. (2.14).
  2. while $((\boldsymbol{\epsilon} - \boldsymbol{\epsilon}')/\boldsymbol{\epsilon} < 100\rho)$
     (a) compute Jacobian $J$ at $\mathbf{p}$
     (b) set $\boldsymbol{\epsilon} = f(\mathbf{p})$, $\mathbf{p}' = \mathbf{p} - (J^\top J + \lambda I)^{-1} J^\top \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon}' = f(\mathbf{p}')$
     (c) while $(||\boldsymbol{\epsilon}'||^2 > ||\boldsymbol{\epsilon}||^2)$
         i. set $\lambda = 10\lambda$
         ii. set $\mathbf{p}' = \mathbf{p} - (J^\top J + \lambda I)^{-1} J^\top \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon}' = f(\mathbf{p}')$
     (d) set $\mathbf{p} = \mathbf{p}'$, $\lambda = \lambda/10$.
  3. Form $\mathbf{t}$ and $R$ from $\mathbf{p}$ using Eq. (2.13).
```

Figure 2.3: Algorithm to optimize the initial pose P.

two poses and a model with 16 model points, in order to investigate how well the orientation and translation is retrieved under different distances from the camera, and different amounts of added noise. Finally, we experimented with real images, to get a feeling of the performance in a real-life setting. The results of these experiments are described in Section 2.5.3.

2.5.1 Performance under six poses

Aim of this test is to investigate to what extent the LM algorithm improves the initial estimate. At the same time we want to get some insight into what accuracy we can expect with different amounts of noise and different numbers of model points.

We selected six poses, the orientations of which are depicted in Fig. 2.4, together with an example of pose six with nine model points and added Gaussian noise of $\sigma = 5$ pixels in the image coordinates. The green arrows show the viewing direction of the face, which coincides with the normal of the model. The orientation of the faces is not intuitively expressed using Euler angles, therefore we use the direction of the normal of the model plane to indicate the orientation of the model. The directions for each pose, together with their accompanying Euler angles are given in Tab. 2.1. The distance to the camera was fixed at 500 mm. We use these particular poses because we believe they represent some of the more common poses a face is viewed under for a system like the Icat. Note that due to similarity, the results of the mirrored set of these poses will be the same. To obtain a realistic setting, we use the calibration matrix estimated for the real camera, used in Section 2.5.3 to acquire images for real-world experiments. The parameters of this camera are given in Tab. 2.1. We define the error in the estimated orientation by the angle θ between the true normal, and the estimated normal of the model plane. The error is averaged over 25 runs and is plotted in Fig. 2.5 and Fig. 2.6 for the described poses.

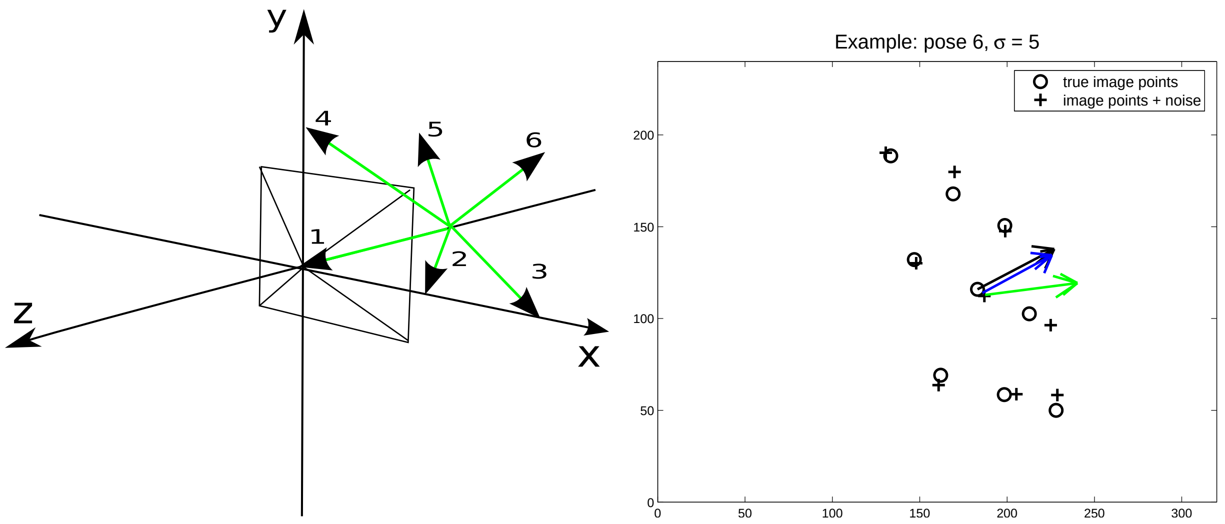


Figure 2.4: Left: the six poses used for the synthetic test. Right: an image example of a synthetic test result, for nine model point, noise level 5. The black arrow indicates the true normal, green the initial estimate, and blue the estimated normal after optimization.

pose	direction	α	β	γ	camera parameters	
1	$[0\ 0\ 1]^\top$	0°	0°	0°	s_x	516
2	$[1\ 0\ 1]^\top$	0°	45°	0°	s_y	515
3	$[2\ 0\ 1]^\top$	0°	63°	0°	x_0	183
4	$[0\ 1\ 1]^\top$	−45°	0°	0°	y_0	116
5	$[1\ 1\ 1]^\top$	−45°	35.3°	−15°	k	0.42
6	$[2\ 1\ 1]^\top$	−45°	54.7°	−24°		

Table 2.1: The Euler angles accompanying the six different poses (left) and the camera parameters (right).

Results and discussion

From the results we conclude that the LM algorithm significantly improves the result established by the initial pose estimate, in particular when the amount of noise in the image points is not that large.

The only exception is pose one, where the model plane is parallel with the image plane, ie. the pose contains no orientation. A striking feature of this result is that at the error is actually slightly increased by the LM algorithm at 81 model points, although not that much. At first sight, this seems an odd result, because the algorithm can only decrease an error function, not increase. To see why the error is increased, note that LM minimizes the Euclidian distance between the expected image points and the measured image points, while the error depicted in the results is the angle between the true normal and the estimated normal. As a result, LM can minimize its error function, while the actual pose estimate is worsen in terms of angles between the normals. Because of the small effect and incidental occurrence, we conclude that the slightly worse result in pose one for the LM algorithm is just a contingency. Note that pose estimation of pose one is significantly worse than the others. We were unable to pinpoint an exact cause of this result.

Another (remarkable feature of the) result is that after about 30 points, the increase in accuracy of the estimated orientation becomes less significant. This is particularly evident in pose four, five and six at noise level $\sigma = 5$. For noise level $\sigma = 10$ there is not much to conclude about the increase in accuracy with increasing number of points; the graphs show improvement, but the line is to bumpy to make any claims at a finer level. At noise level $\sigma = 1$ there is no significant improvement in the estimated orientation, which is due to the fact that their is not much to improve.

2.5.2 Performance under increasing depth

The second synthetic test was performed to get some insight on the accuracy of the estimated pose (both translation and rotation) in relation to the distance to the camera, for a model with 16 points. We use this relative small amount of points, because we expect that not many consistent facial points can be found. The results for pose **one** and pose **six** are shown in Fig. 2.7. The results for the other poses are not shown, because they are approximately the same as pose six, and so do not provide any extra insight.

Results and discussion

There are some strange hick-ups in the estimation of the translation of pose six. There is no good explanation for those hick-ups, but the accompanying estimated orientations are not good either. An explanation could be that of the 25 runs there were a relatively high number of bad initial estimates, which the LM algorithm could not improve much. However, because the estimated orientation is really bad for these depths as well, it is not very interesting to find out what exactly causes these hick-ups, so we did not further investigate it. Note that the estimation of the pose of the face up to about 20 degrees is still usable in practice (it gives a good indication of the viewing direction), so this particular configuration is usable for distance up to 80 centimeters from the camera with a fair amount of noise $\sigma = 5$.

Although the results provide useful information on reasonable boundaries for the application of the proposed methods, a second line of thought on the results does realize that increasing the depth is effectively the same a increasing the noise. To see this note that

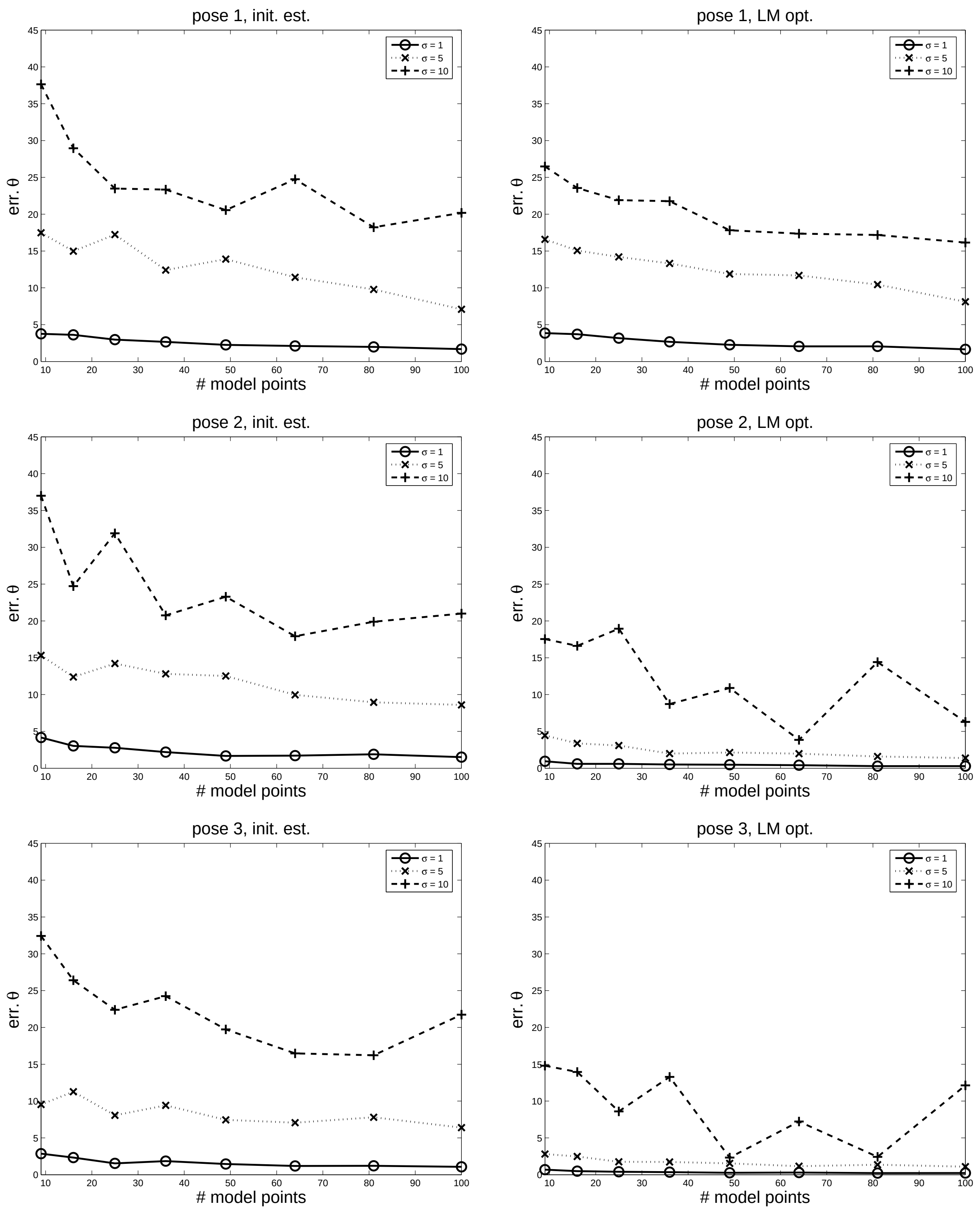


Figure 2.5: Error in the estimated orientation for pose one, two and three, for different numbers of model points and three levels of noise. Left graphs show the error after the initial estimate, right the estimate after optimization using the LM algorithm.

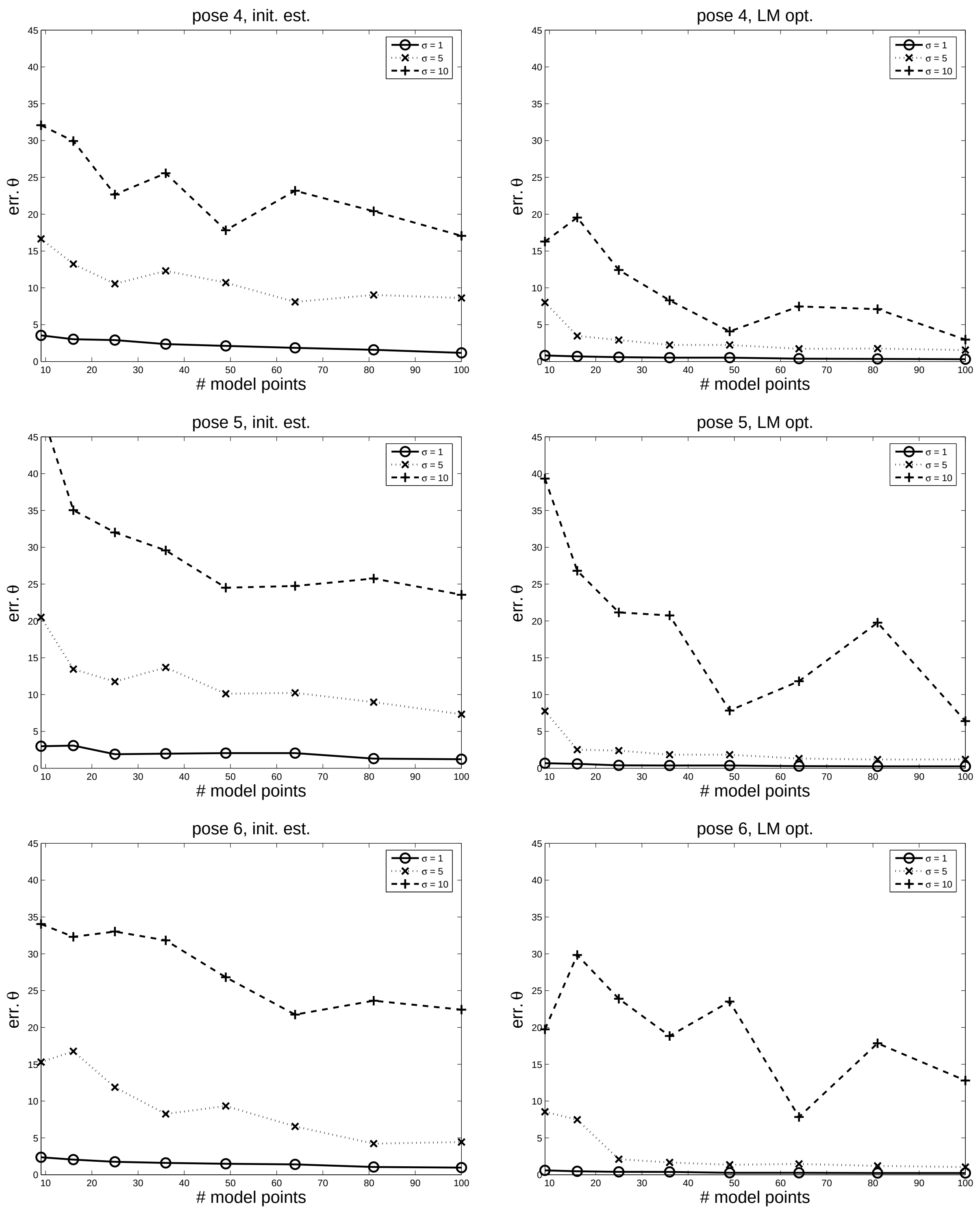


Figure 2.6: Error in the estimated orientation for pose four, five and six, for different numbers of model points and three levels of noise. Left graphs show the error after the initial estimate, right the estimate after optimization using the LM algorithm

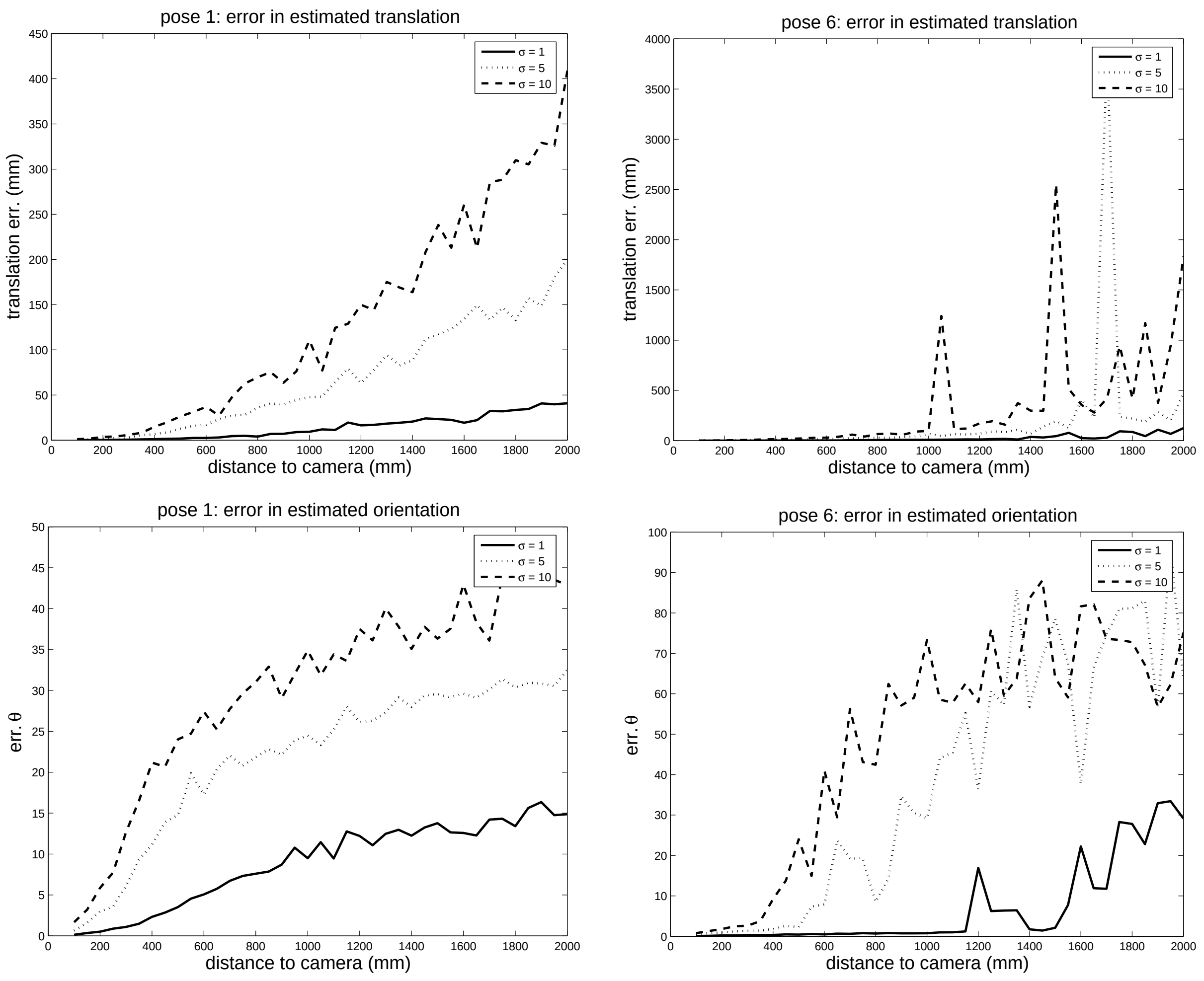


Figure 2.7: Error in the estimate pose after optimization using the LM algorithm for a model with 16 points under increasing depth, and different levels of noise. Top graphs show the error in the estimated translation for pose one and pose six. Bottom graphs show the error in estimated orientation for the same poses. Note that the scale of the y-axis differs.

the overall scale of the image model is inversely proportional to distance from the camera, due to the perspective projection of the pinhole camera. On a smaller scale, the same level of noise distorts the perspective information to a relative larger degree. So, increasing the distance from the camera is effectually the same as increasing the relative noise in the image coordinates.

2.5.3 Qualitative test: real images

To get some feeling of how the method performs on real-world images, an final experiment was conducted with a simple model of face points and images of a face under different poses.

The model was created by measuring the relative position of six facial points directly on the face. The resulting model is depicted in the top image of Fig. 2.8. Corresponding points of the model in each image where marked by hand for four images of the author under different poses, shown below the model. The camera was calibrated using the method presented in [25], which also provides a reference to an implementation of the method by the same author. The parameters of the camera intrinsic matrix were already given in Tab. 2.1. Fig. 2.8 shows estimated poses of the face in each image.

Results and discussion

Although only six model points are used, the result is fairly good. As expected from the results of the synthetic tests, the estimated orientation of the frontal face image is less good than the estimation of the pose of the faces under more rotation. In this case, however, there is also an obvious cause: the corners of the mouth (point one and two) are slightly higher than normal due the smile. This flattens observed vertical scale of the image points somewhat, and biases the estimation towards inclination in the pose.

2.6 Conclusion

We have shown how the pose, consisting of the position and orientation of the face is estimated as using a planar model of points. In order to do so, an initial estimate of the pose is computed from the homography relating the model and image points, which is accordingly optimized using the Levenberg-Marquardt optimization routine. The results from the experiments show that moderately good result can be achieved even with a small number of model points. Up to 30 points the accuracy of the estimated pose is improved if the noise in the image points is not to high; adding more points does not improve the result much. We showed that frontal poses are less accurate estimated than that of a face under out of plane rotation. Tests on real-world images show that face pose estimation is possible using a small amount of model points. For the used camera, we therefore expect to be able to estimate the face pose up to about 800 mm from the camera, although the accuracy this depends on the particular pose of the face, and, of course, on the amount of noise in the image points.

Now that we have shown how to estimate the pose of the face using a planar model, we need to acquire the facial parts representing the model parts and a method to detect them. This is dealt with in the next chapter, where part acquisition and detection is done using the Sift detector. Chapter four uses the result from this chapter together with the result from the next chapter to automatically detect faces in images and to estimated the face pose.

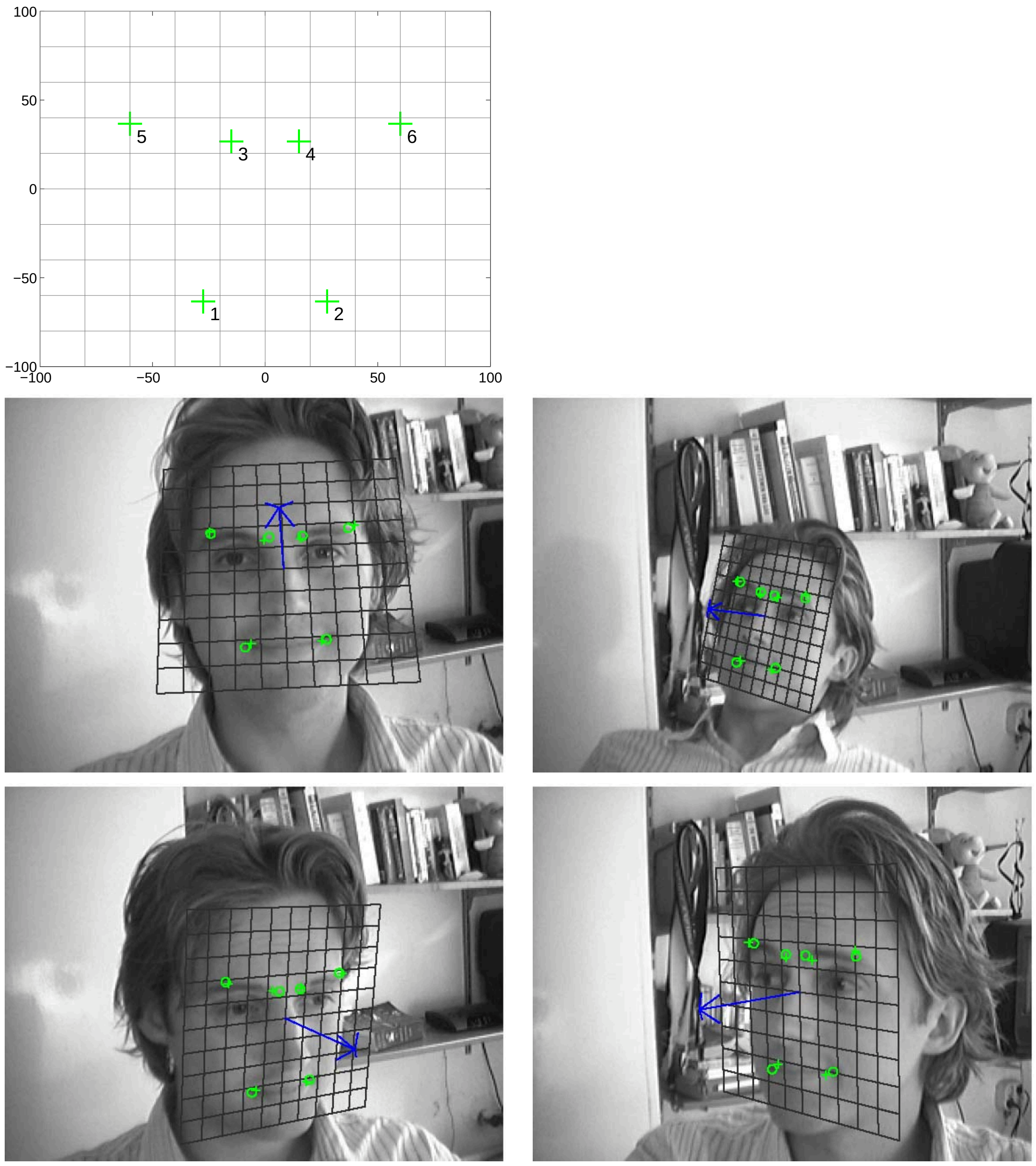


Figure 2.8: Top image: the model plane with six points. Rest: estimated poses.

Chapter 3

Part acquisition

The question we investigate in this chapter is how to detect facial points. In the previous chapter the points were marked manually; here we're in pursuit of their automatic detection. From here on we will refer to the facial points as parts. No matter how their detections takes place, it will be rare that no false positives will be returned. Each part detector therefore returns a set of candidates for that part, out of which we have to select the right one. The next chapter describes how a geometric constraint in the form of a spatial part model can be employed for this purpose. Curious readers (or hasty) can peek ahead to the next chapter. This chapter mainly describes the our investigations to select the facial parts and how to detect them.

3.1 Problem statement

To detect parts in images, we need to decide which type of detector is most appropriate for our problem. Generally speaking, detectors fall into one of two categories: special purpose detectors, learned to detect specific intensity patterns, and more general purpose detectors which are used to find salient (interesting) features in an image. An example of the first type is template matching, where the similarity between a part template and an image region is measured. Typically, the image region is of the same size as the template. A simple and often used similarity measure is normalized cross correlation [15]. Examples of this type of matching applied to the detection facial points is found in [1], [14], where templates are created for the eyes, nostrils and nose from the respective regions in training images. As remarked in [1], this type of detector results in many false positives, and often true positives are missed as well. Another drawback of this approach is that the selection of parts is based solely on our intuition as experts of face detection. This might bias the selection process towards parts which may be easily detected by humans, but sub optimal in combination with in the computational models used in computers. A third drawback of template matching is its sensitivity to object scale and rotation, and lighting conditions.

Instead of template matching, or other forms of special purpose detectors, we may use saliency or interesting point detectors. Many computer vision tasks rely on this type of detectors, and a lot of research is committed to the development of such detectors which are invariant to scale and rotation. These type of detectors do not find regions for which they are specially trained, but instead respond to regions which are in some respect interesting (salient). The term 'interesting' refers here to the property of the region to be (relatively) easily detectable by the same detector under different scale and/or orientation. The most famous of these interesting point detectors are the Harris corner detector [5], the entropy based detector by Kadir and Brady [9] and Lowe's Sift detector [12]. The first detects point at a single scale, the latter two detect point at different scales.

The rest of this chapter is devoted to the use of Sift features to acquire and detect facial parts. The reason we have chosen Sift features for this purpose, is that the Sift detector is, by construction, invariant to scale, in-plane rotation and to a lesser degree out-of-plane rotation. This is a useful property, because it allows to use the same part model to detect the facial feature under different scale and rotation. Because our aim is to estimate the face's pose, we must be able to detect the facial parts under different scale and rotation and using the Sift features this is greatly simplified. We also experimented with the entropy detector by Kadir and Brady, but it turned out to be very slow, compared to the Sift detector and yield much worse results. Finally, the Harris detector yielded too many point detections on the face (especially in the eye-region). As a result, selecting good facial features, consistent across different images, was intractable to do by hand.

In order to understand how and to what extent the Sift-detector can be used for face part acquisition and modelling, there are two main questions we need to answer. First, how many consistent Sift-features are found on a human face? With consistency we refer here to the presence of a feature in the same region of a face across different images of the same person and across different persons. A high presence rate is important to ensure we have enough detected parts per image to detect our object. Sift feature consistency for faces is investigated in Section 3.3. Secondly, how can we find the particular face parts among all features returned by the detector? What performance can be expected from this part detection? We consider two detection methods: the matching method proposed in [13], discussed in Section 3.4 and detection based on modelling the part descriptors using a mixture of Gaussians, discussed in Section 3.5. We start with brief discussion on the workings of the Sift detector in Section 3.2.

3.2 Sift features

For an extensive treatise on Sift features dealing with all its intricacies and some applications we refer [13]. The material covered here provides a technical outline of the Sift procedure, sufficient to understand its application in this thesis.

The Sift (Scale Invariant Feature Transform) feature detector, developed by Lowe [13], finds interest regions, called features or keypoints, located at peaks in the difference of Gaussian function convolved with the image in scale space. In other words, they are located in regions and at scales where there are large intensity gradients in all directions when compared to neighboring scales and location. Fig. 3.1 provides a graphical interpretation of the procedure. Aside of determining which regions of the image are interesting, there is also the question of how to represent this region. The approach developed by Lowe is to first blur and resample the region at the appropriate scale and then transform it into a region of gradients. From here, the region is divided into $m \times m$ patches. Each patch is then described as a k-bin histogram of image gradients, see Fig. 3.2. The advantage of such a representation is that it is invariant to global brightness changes, rotation and scale, and somewhat invariant to affine distortions.

In our experiments we use the Sift detector implemented by Z. Zivkovic at the University of Amsterdam, which is based on original code provide by Lowe [11]. This implementation

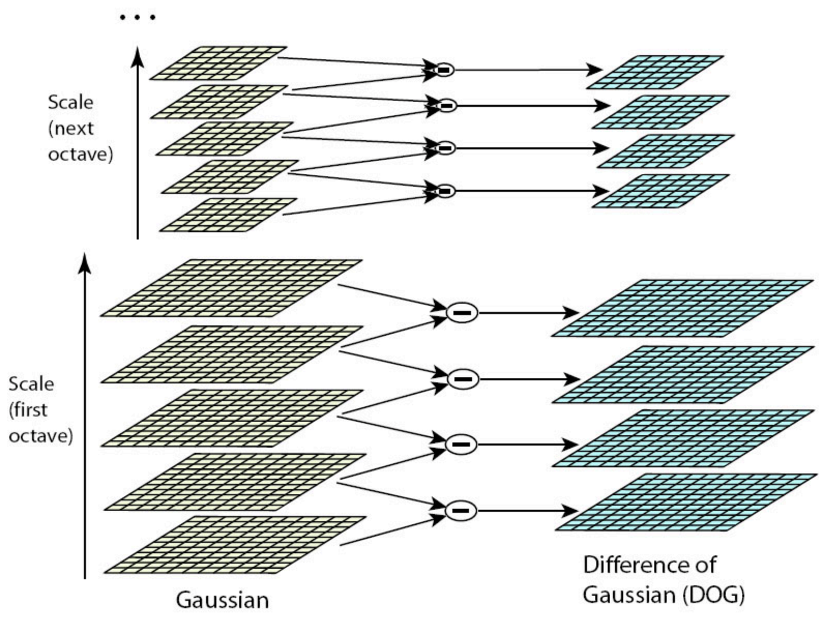


Figure 3.1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated. Image taken from [13].

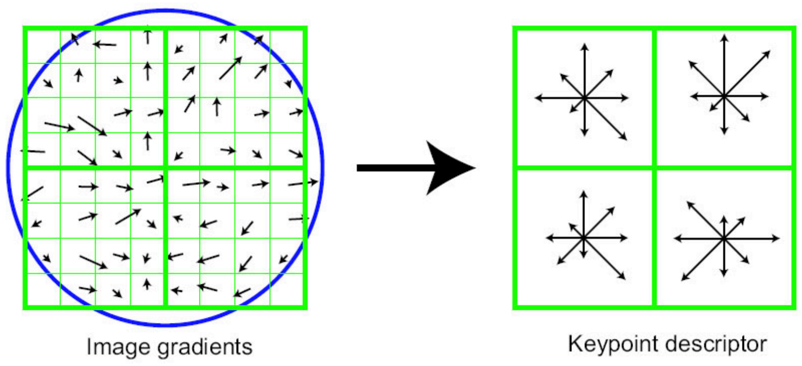


Figure 3.2: A feature (keypoint) descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the feature location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over $4 \times 4 = 16$ subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. The figure on the right shows a 2x2 array of gradient histograms using 8 bins, resulting in a 32 dimensional descriptor. The experiments in this thesis use a $4 \times 4 = 16$ array represented by histograms of 8 bins, resulting in 128 dimensional descriptors. Image taken from [13].

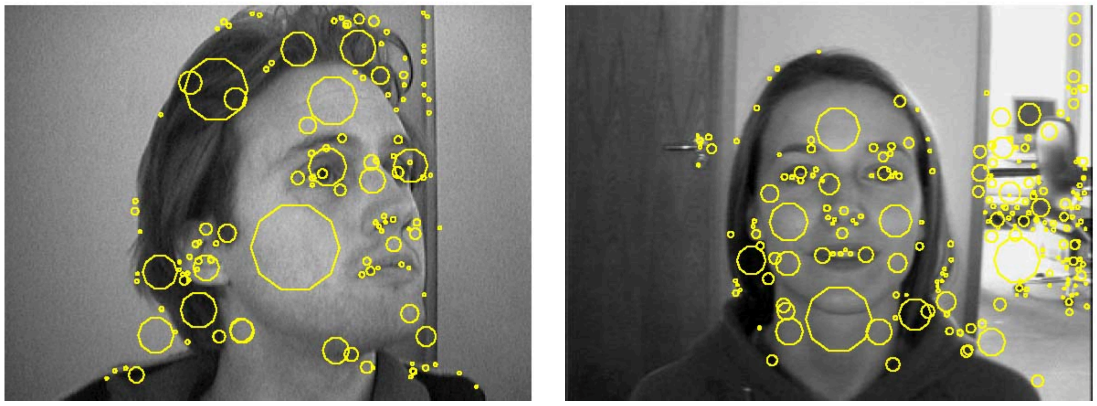


Figure 3.3: Sift responses for a sample image of the pose date set (left) and the bioid data set (right).

is restricted to $k = 8$ bins and $4 \times 4 = 16$ patches, resulting in an 128-dimensional image descriptor. Fig. 3.3 shows the Sift response for a sample image for the pose and bioid data set described in the next section.

3.3 Sift feature consistency on faces

Before committing ourselves to Sift features, we investigated to what extent the features detected by the Sift-detector return consistent regions among faces. We recognize two categories: (i) consistency between different persons, and (ii) consistency between different poses for the same subject. The first category is a measure of the general applicability of using Sift features to detect faces. The second category gives an idea of the applicability of the Sift detector for face pose estimation. To measure the consistency of both categories, we have hand labelled facial features in three different data sets, which are described below. Consistency is then simply measured as the percentage of images in which the feature was present.

3.3.1 Experimental setup

For each image, we have hand-labelled one feature corresponding to a certain region of the face, or none if there were no such feature. Examples of face regions used are the pupil, corner of the mouth, etc. We tested on three data sets, indicated with 'yale', 'bioid', and 'pose', all of which contain only intensity images.

pose The pose data set contains 60 images of size 320 $\times$ 240, each showing the face of one subject (the author) under different poses. The set is build from 3 takes of 20 images each, where the face pose in each image ranges from -45 to 45 degrees 'spinal' rotation. Additionally in take 1, 2, 3 the face has inclination of respectively -30, 0 and 30 degrees. The number of Sift feature is about 150 per image.

yale This is a subset of the Yale Face Database B [4]. For each of the 10 subjects in this database pictures are taken under different lighting conditions and 8 different poses. Poses 1, 2, 3, 4, and 5 are about 12 degrees 'spinal' rotation with respect to pose 0, while poses 6, 7, and 8 were about 24 degrees. For the conducted experiment we used the images under frontal light only, and cropped the image such that it contained only the face area. Each image (80 in total) is of size 301×241 with on average 150 Sift features.

bioid The BioId Face DB (`www.humanscan.de`), consists of 1521 images with a resolution of 384×286 pixels, from which we took every fifth image. Each image in this set shows the frontal view of a face of one out of 23 different test persons. The pictures are taken in an loosely controlled office setting. In total this set contains 304 images. The number of Sift-feature responses range from 100 to 600 per image.

Using the bioid data set we measure inter-person consistency of facial parts. The pose data set is used to measure the consistency under different poses. Finally, the yale data set contains different persons as well as different poses, so this is a measure of the both categories at once.

3.3.2 Results and Discussion

The results of the consistency experiments are found in Tab. 3.3.2. The pose data set shows good results for different poses (same person), but it should be noted that the imaging conditions for this set where highly controlled. The imaging conditions under which the images of the bioid data set were acquired are more realistic and a significant drop in consistency for this data set is observed. This decrease is also a result of more difference in face appearance caused by the fact that there are different persons in the images.

Together, we believe the experiments indicate that the Sift detector provides enough consistency on faces to be applicable to facial part detection, and therewith to face detection and to face pose estimation. In the next section we proceed with method for detecting the face parts among all features found by the detector in an image.

The second question we want to answer in this chapter is how we can find the right parts among all features returned by the Sift detector. We investigated two methods to accomplish this. The first involves the descriptor matching method as presented together with the Sift detector in [13], described in the next section. It turned out that this method yields unsatisfactory results on faces, and this section can be skipped without loss of continuity. Section 3.5 presents the second approach, where the descriptor is modelled in a probabilistic sense using a mixture of Gaussians, in which each mixture component represents a part.

3.4 Detecting parts: descriptor matching

In [13] Sift features are used to find objects in a large database of images. An object is represented by an example image of the object segmented from the background. Features are matched based on the Euclidian distance between feature descriptors from the image and the database. To reduce the number of false detections, the matching between descriptors takes actually place by measuring the ratio of Euclidian distances of the closest neighbor and second closest neighbor to the descriptor. The feature is only matched to the closest neighbor, if the second closest neighbor is far enough away. The rationale behind this method is that for false matches, there will likely be a number of other false matches within similar distances due to the high dimensionality of the descriptor space. The method rejects all matches in which the distance ratio is greater than 0.8. This number was found empirically, by comparing the probability density functions over the distance ratio for correct matches against incorrect

part no.	pose (%)	yale (%)	bioid (%)	face region
1	82	19	67	right eye
2	97	91	70	right pupil
3	100	56	51	top nose bridge
4	93	43	54	left eye
5	100	96	48	left pupil
6	48	82	80	chin
7	98	41	77	forehead
8	68	90	56	left cheek
9	80	93	57	right cheek
10	72	85	83	tip of the nose
11	60	38	40	right of left eye brow
12	65	68	49	left right eye brow
13	85	58	50	center upper lip
avg	80.6	66.2	60.2	

Table 3.1: Feature consistency for the three data sets, measured as the percentage of images with a feature in the appropriate face region.

matches, see Fig. 3.4. The matching method returns either an index to the matched feature in the scene, or a zero if the match is not available

3.4.1 Experiments

Initially we were interested in using the method described above to match parts. The reason for this is that if this method provides reasonable results, it would be a very powerful method to find facial parts in a new image, by using a single (for example frontal) image of a face as model. To test the matching method, three experiments were conducted. First we repeated the example Lowe gives in [12], (Verification test). A second experiment tests the matching method on a planar object with different in- and out-of-plane rotation (Planar test). The last experiments tests the matching method on faces from the Yale Face Database B (Yale Face test).

Verification test

To verify the implementation and to see if we get the same results as Lowe, we repeated the example Lowe gives in [12]. In this example three objects, segmented from the background and in parallel with the image plane, are matched against a scene containing a constellation of the same objects. In the scene, the objects occlude each other and have undergone out-of-plane rotation. In Tab. 3.2 the matching for the three objects in a scene are depicted. The top row shows the sift responses for each object. The second row shows the scene. If an object feature matches to a scene feature, the scene feature is depicted with the same color as the object feature. Note that if there were false matches, these would be seen as scene

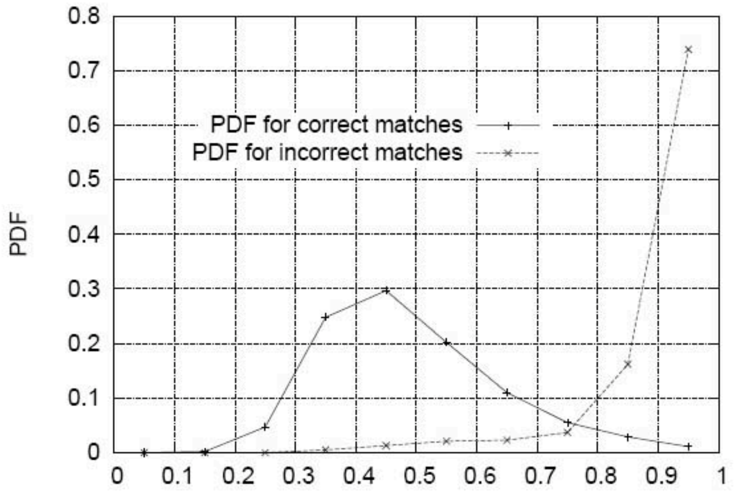


Figure 3.4: Probabilities of the ratio distances (closest/next closest), for correct and incorrect matches. Figure taken from [13].

features in an object color, located anywhere but on the corresponding object. In order to obtain ground truth on the performance of the matching method, we would have needed all [object feature, scene feature]-correspondence pairs, which were not available. However, by visual examination of the results we can still gain useful insight. The precision for this setting is (probably) moderately high. It is not at all perfect: for example the lowest red scene feature is a false positive, as is the highest green one. Closer inspection reveals even more false positives. (Note that there are also some large features, present in both the model and the scene that are not matched, while one would expect so, because of their high resemblance. An example is the largest feature in the white area above 'basmati'.)

Planar test

A second test was done on a planar object, under different in- and out-of-plane rotation. This experiment was merely conducted to test wether the matching method yields approximately the same result as Lowes example, but on our own images. This was done to test whether Lowes example was just a (good) result for a particular situation, or applies in general. The result is in favor of the latter situation, and is found in Appendix B, Tab. B.1. The second reason this experiment was conducted, was to investigate the behavior of the matching method with increasing out-of-plane rotation. As expected, the method performance degrades rapidly with increasing out-of-plane rotation. This is not surprising, because the Sift detector itself is not invariant to perspective distortion.

Yale Face test

A third test was conducted to investigate the behavior of the Sift detector and matching method on faces. We were interested to see which regions of the face are returned by the Sift detector and how the matching method behaves in this case. The **yale** data set described above, was used for this purpose. For each of the ten subjects in this database, pictures

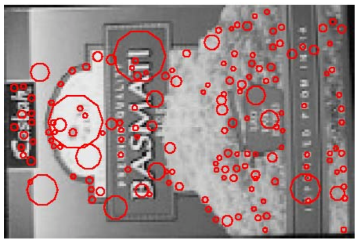
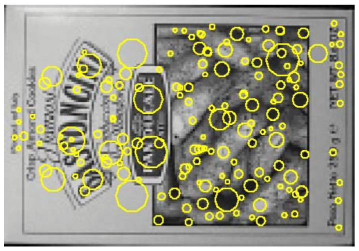
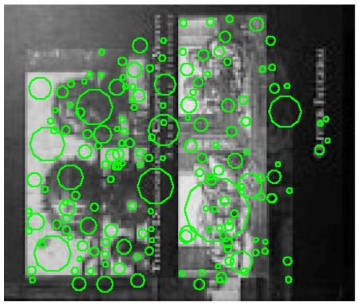
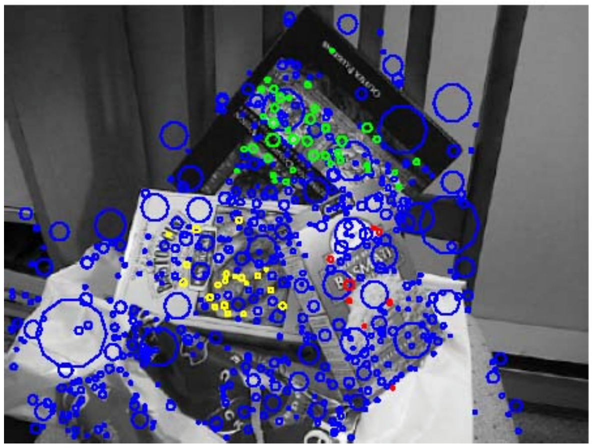
object 1	object 2	object 3
		
	scene	
		

Table 3.2: Sift feature matches for images and scene. Images from [12].

are taken under different lighting conditions and eight different poses. For the conducted experiment we used the images under frontal light only. Appendix B, Tab. B.2 shows results for three persons under poses 1 and 2, in which the faces have undergone about 12 degrees 'spinal' rotation relative to the frontal pose, pose 0, and poses 6 and 8, which are under 24 degrees spinal rotation. Note that in many images the pupil areas are consistently returned as sift features, as are the cheeks and mouth corners. The matching method does find some feature correspondences, however, this happens only when the images are very much alike. Pose 8 hardly shows any matches. Clearly, the matching method of [13] is not robust enough for our application.

3.4.2 Results and discussion

We confirmed that the matching methods as presented in [13] give useful results on planar objects. However, on faces, the method does not yield usable results. We have seen that some feature pairs, although clearly corresponding to the same part (for example the left eye), were not matched. Another drawback of the method is that is does either provide a best match, or nothing if the method believes there is no reliable match. We would prefer a method returning candidate matches accompanied with confidence measure. The reason for this is that we can select, for example, the best two candidates for each part in a new image and let a spatial model, described in the next chapter, decide which of two is the right one. The face is very symmetric in nature, so it may not be uncommon that a 'left eye'-part matches first to a 'right-eye'-feature and secondly to the correct 'left-eye'-feature, with only a slight decrease in probability. It is the responsibility of the spatial model to determine the true best candidate among all others by means of geometric constraints on the parts relative canonical locations for the object class. This is treated in the next chapter.

3.5 Detecting parts: modelling the descriptor

In the previous section we concluded that the fast matching method proposed in [13] does not perform well enough to find facial parts. In this section we are therefore concerned with modelling the descriptor of the features in statistical sense, by estimating a mixture of Gaussians (MoG) of the features descriptors, to obtain a (hopefully) better detection rate, and more control over its behavior. The components of this mixture represents the parts and the background. Because the descriptor is invariant under image rotation and scale by construction, each mixture components only needs to capture the difference in appearance between different detections and not the difference in appearance due to different scale or orientation of observation. Using the MoG, the probability of observing a descriptor $\mathbf{a}_f$ is given by

$$p(\mathbf{a}_f) = \sum_{p=0}^{P} p(\mathbf{a}_f|\mathcal{P} = p)\, \pi_p, \tag{3.1}$$

where $\pi_p \geq 0$, $\sum_p \pi_p = 1$ are the component priors or mixture weights, and $p(\mathbf{a}_f|\mathcal{P} = p) = G(\mathbf{a}_f; \Sigma_p, \boldsymbol{\mu}_p)$ the Gaussians modelling the appearance. The form of a multivariate Gaussian probability density function (pdf) is given in Appendix A. In this mixture model, component 0 represent the background, and all other component represent a part. By selecting the features for which the posterior is maximal for a certain part, we obtain a set of candidates for each part.

Because each descriptor is a vector in a 128 dimensional space, estimating the full covariance matrix cannot be done robustly because we have far too little training data to estimate over 8000 parameters per component. We must therefore restrict the covariance matrix to a form with less parameters. We consider three types of restrictions: an isotropic covariance matrix, independent variables assumption, and a probabilistic form of principal component analysis: probabilistic pca (). We believe it makes sense to use different types of restriction for the part components and the background component. The reason for this is that we want the background component to express a more general probability of observing the descriptor, while the part components should be more specific for the probability of the individual parts. The isotropic covariance matrix is the most restricted form: it does not capture much of the structure of the data. In contrast, probabilistic pca captures a subspace of the full data space covering most of the variance in the training data (in our case we require ppca to explain at least 95 % of the variance). The independent variables is somewhat in between these two forms. The next paragraphs describe the specific forms is some more detail. The maximum likelihood estimators for each form applied to our situation and the the estimation of component priors are found in Appendix A.

Isotropic An isotropic covariance matrix Σ has only one parameter σ:

$$\Sigma = \sigma I. \tag{3.2}$$

This is the most restricted form of the covariance matrix, because it has only one parameter. It does not capture any structure of the data, besides global scale. We included this form of restriction because it effectually degrades the probability measure based on the Mahanalanobis distance to a probability measure based on the Euclidian distance. Using the Euclidian distance between to samples is the most basic form of matching.

Independent variables In this case the variation of the individual variables is captured:

$$\Sigma = \text{diag}(\boldsymbol{\sigma}), \tag{3.3}$$

where $\boldsymbol{\sigma}$ is a vector containing the standard deviations of the individual variables, and 'diag' refers to the procedure of putting the elements of a vector on the diagonal of a matrix. Instead of just one, this form has 128 parameters. This form can be interpreted in the same way as the isotropic form, but with the difference that it weights each variable in the Euclidian distance measure.

Probabilistic PCA Probabilistic principal component analysis (ppca), as proposed in [19] restricts the covariance matrix to the form:

$$\Sigma = \sigma I + WW^\top, \tag{3.4}$$

where W is of size $128 \times k$. This form assumes that the samples for each component lay on a k-dimensional subspace ($k << 128$), representing the structural difference in appearance of the part, and deviate from this subspace according to an isotropic noise model $\sigma^2 I$. This structural difference in appearance of the part is described by the so-called latent variables. The assumption behind this model is that each appearance $\mathbf{a}$ is generated by

$$\mathbf{a} = W\mathbf{x} + \boldsymbol{\mu} + \boldsymbol{\epsilon}, \tag{3.5}$$

where the latent variables in $\mathbf{x}$ are assumed to be independently distributed with unit variance and with prior $p(\mathbf{x}) = G(\mathbf{x}; 0, I)$. The noise $\boldsymbol{\epsilon}$ is assumed to be generated by an isotropic gaussian distribution. We may view the latent variables $\mathbf{x}$ as describing the degrees of freedom in appearance, and the noise as a result of the data acquisition process. The number of parameters here is the size of W plus one for the isotropic noise. In [19] the maximum likelihood estimators for this model are derived, which are given here in Appendix A.3. The advantage of this form for the covariance matrix is that we retain most of the dependency information (in our case 95 %) between the individual variables of the descriptors, while significantly reducing the number of parameters. Tab. 3.3 gives the number of principal components used to capture 95 % of the variance, for the data sets used in the experiments. The difference in the number of components is explained by the fact that bioid data set contains images of different persons, while the pose data set contains images of a single subject. As a result, there is more variety in the descriptors of the bioid data set parts than in the descriptos of the pose data set parts.

part	bg	1	2	3	4	5	6	7	8	9	10	11	12	13
ppca components														
pose	65	8	9	6	11	9	8	6	10	10	11	5	8	10
bioid	65	44	43	36	41	42	44	44	49	47	49	26	34	44

Table 3.3: The number of principal components explaining at least 95 % of the variance for the bioid and pose data set.

3.5.1 Detecting the parts

Suppose we have P parts in our model. Let $p(\mathbf{a}_f|\mathcal{P}=p),\ p=\{1,..,P\}$ denote the probability that the descriptor $\mathbf{a}_f$ of feature f is generated by the component representing part p and $p(\mathbf{a}_f|\mathcal{P}=0)$ the probability of the descriptor being generated by the background component. The probability of the descriptor being generated by component p is computed using Bayes' rule:

$$p(\mathcal{P}=p|\mathbf{a}_f)=\frac{p(\mathbf{a}_f|\mathcal{P}=p)\pi_p}{p(\mathbf{a}_f)}. \tag{3.6}$$

Each feature is now classified by selecting the part which has maximum a posteriori probability $p(\mathcal{P}=p|\mathbf{a}_f)$. We do not need to compute $p(\mathbf{a}_f)$ when we just want to select the best part among the alternatives. It is just a normalization factor to ensure that the $p(\mathcal{P}=p|\mathbf{a}_f),\ p\in\{0,..,P\}$ sum to unity. However, we do need normalization if we want to compare the posterior probabilities between different features. We might do this in order to select a subset of the features classified as a certain part, to ensure that the hypothesis space does not grow too large when we have many candidates per part. There is practical issue when computing the posterior probabilities due to the high dimensionality of the data, which is dealt with in Appendix A.5.

3.5.2 Experiments

We tested the detector described in the last section on the pose and bioid data set, using nine configurations of the mixture model. The nine configuration include all possible combinations of the three forms for the covariance matrix, employed to the background component and to the part components. Each configuration was tested on both the bioid and pose data set using cross validation, leaving one image out at a time to test on and estimating the parameters using the rest.

3.5.3 Results and discussion

Performance of the classifiers is expressed through the use of precision and recall:

$$\text{precision}=\frac{\text{true positives}}{\text{true+false positives}},\quad \text{recall}=\frac{\text{true positives}}{\text{instances of the object class}}. \tag{3.7}$$

Results for the different configurations are shown in Tab. 3.4 - Tab. 3.7. The last row of each table, indicating the average precision and recall of the part detection, is most interesting, but note the sometimes huge difference among the individual parts as well the striking zero-valued entries.

bg component →	iso	iso	iso	indep	**indep**	indep	ppca	ppca	ppca
fg component →	iso	indep	ppca	iso	**indep**	ppca	iso	indep	ppca
↓ part									
bg	96	100	100	0	83	100	0	0	0
1	65	0	0	73	88	86	73	92	100
2	74	0	0	78	93	86	78	95	97
3	83	0	0	95	97	93	95	97	100
4	71	0	0	91	88	82	91	93	100
5	70	0	0	82	88	82	82	97	98
6	52	0	0	79	90	21	79	90	93
7	81	0	0	83	97	90	83	97	98
8	76	0	0	88	93	66	88	93	98
9	69	0	0	98	96	71	98	98	100
10	72	0	0	93	88	65	93	95	93
11	53	0	0	89	86	69	89	86	97
12	49	0	0	97	92	54	97	92	97
13	45	0	0	96	96	75	96	98	100
avg on parts	66	0	0	88	92	72	87	94	98

Table 3.4: **Recall** (%) for part detection on the **pose** data set.

Regarding the pose data set

The independent variables assumption yields the best result for the pose data set, when used for both the background and the part components (see Tab. 3.4 - Tab. 3.5). Although using ppca for the part components results in a far better precision (97%!), we observe a significant drop in recall. Another significant result is that using the isotropic covariance matrix for the background component, not a single correct part detection is returned when either the independent or ppca form is used for the part components. This is a unexpected and we were unable to find a satisfying explanation for this behavior. We conclude that our best option is to use is the indep/indep combination with the pose data set. With a recall of 92% and an average presence rate of 80.6% (see Sift feature consistency on the pose data set, Tab. 3.3.2), we can expect $0.92 \times 0.806 \times 13 \approx 9.6$ true positives among the part candidate sets.

Regarding the bioid data set

On the bioid data set (Tab. 3.6 - Tab. 3.7) the ppca performs better. Using ppca for both the background as well as for the part components is the best choice here, as it yields the highest recall. However, the precision is dramatically low at 8%. Using the double isotropic configuration, a slightly better precision is achieved, leading to a remarkably high recall at 41% as

bg component →	iso	iso	iso	indep	**indep**	indep	ppca	ppca	ppca
fg component →	iso	indep	ppca	iso	**indep**	ppca	iso	indep	ppca
↓ part									
bg	98	92	92	0	99	98	0	0	0
1	80	0	0	59	81	98	59	26	70
2	96	0	0	88	81	98	88	29	6
3	56	0	0	58	75	100	58	75	11
4	40	0	0	3	43	100	3	5	9
5	86	0	0	33	68	100	33	20	4
6	83	0	0	2	6	100	2	4	36
7	100	0	0	100	100	98	100	100	3
8	33	0	0	5	35	93	5	13	52
9	61	0	0	22	46	97	22	11	16
10	22	0	0	2	13	97	2	1	12
11	100	0	0	24	21	100	24	13	18
12	83	0	0	5	15	91	5	7	6
13	77	0	0	14	74	95	14	8	7
avg on parts	71	0	0	32	51	97	32	24	19

Table 3.5: **Precision** (%) for part detection on the **pose** data set.

well. However, because of the average presence rate at 60.6% (see Sift feature consistency on the bioid data set, Tab. 3.3.2), a recall of 41% results on average in $0.606 \times 0.41 \times 13 \approx 3.2$ true positives among the part candidates. As will be shown later, three matched parts or less leads to a pose which perfectly fits the model part locations onto the image part locations, paralyzing the ability of the spatial model used in the next chapter to find the optimal matching. Our only option therefore is to use the ppca form for both the background as well as part covariance matrices, which results in $0.606 \times 0.71 \times 13 \approx 5.5$ true positives among the part candidate sets. Note that the recall of the background class is in this case 0, meaning that every detected feature is designated as one of the parts in the model.

The low precision on using the ppca/ppca combination on bioid data is problematic: an average precision of 8% on average indicates that we can expect about $1/0.08 = 12.5$ candidates per part per image (true+false positives = 1/precision, see Eq. (3.7)). In the next chapter, a choice from the set of candidates for each part forms a hypothesis, assigning a particular candidate to each part resulting in a matching between the model and the image. The issue we face is that the number of hypotheses is exponential in the number of candidates. Although we use heuristics and pruning techniques to search through the hypothesis space, the computation time needed to find the optimal matching is 20 minutes at maximum and 145 seconds on average, due to the number of candidates for each part. A better classifier, with at least higher precision on the same recall, would therefore be very useful. However, we leave this issue for future research, noting that the use of color information could play a significant role in construction such classifiers. The bioid data set contains intensities images only, so we would need to construct a new training set for this purpose.

Finally, we note that we reduced the number of features for the bioid data set before clas-

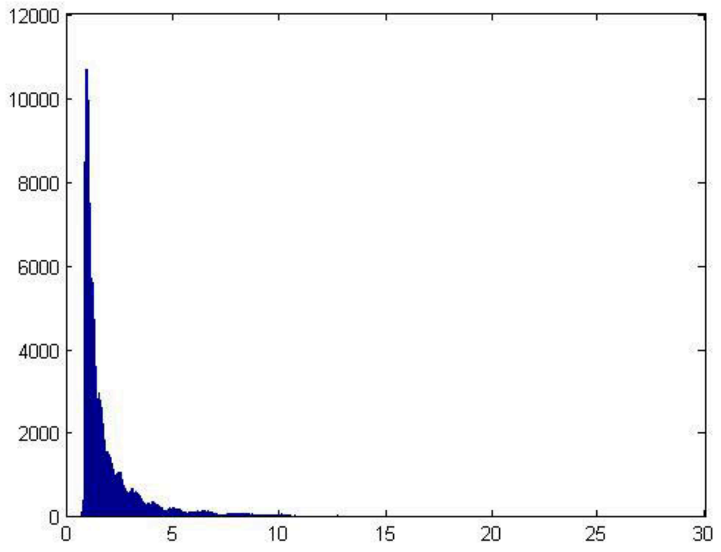
bg component →	iso	iso	iso	indep	indep	indep	ppca	ppca	**ppca**
fg component →	iso	indep	ppca	iso	indep	ppca	iso	indep	**ppca**
↓ part									
bg	80	100	100	0	45	100	0	0	0
1	48	0	0	48	66	14	48	66	89
2	45	0	0	47	74	35	47	74	89
3	60	0	0	64	76	25	64	76	69
4	45	0	0	47	53	2	47	56	63
5	42	0	0	46	55	39	46	55	84
6	31	0	0	45	29	0	45	37	45
7	61	0	0	61	65	31	61	67	90
8	33	0	0	37	37	3	37	41	89
9	47	0	0	48	45	0	48	52	52
10	29	0	0	29	51	5	29	51	93
11	35	0	0	62	59	14	62	59	41
12	25	0	0	54	79	25	54	79	69
13	27	0	0	45	56	11	45	58	53
avg on parts	41	0	0	49	57	16	48	59	71

Table 3.6: **Recall** (%) for part detection from the **bioid** data set.

sification, by pruning every feature with radius smaller than three. From the Sift feature scale distribution, depicted for the bioid data set in Fig. 3.5, it is clear that this lead to a great reduction (in practice 30% up to 80%) of the used features and therewith the number of candidates (remember that not one feature was classified as background using the ppca/ppca combination).

3.6 Conclusion

We investigated whether the Sift detector, introduced by [13] is applicable for the acquisition of facial parts, to be used in a planar model. We showed that the detector yields a usable consistency on preselected facial features. Further, we explained that the, by construction, insensitivity to scale and rotations, gives the Sift detector a head start as a basis to detect faces under different scales and rotation using only one model.
We tested whether the matching method proposed in [13] is also applicable to match facial features between images to find the corresponding facial features, by using one image as a model. The experiment we conducted shows that this matching method does not perform well enough when applied to facial features. Moreover, the method lacks control over the matching process. We also explored a Bayesian detector based on modelling the feature appearances as a mixture of Gaussians. The mixture model uses one component to represent each part, and one component for the background. Because of the high dimensionality of the descriptor vectors, and the relative small amount of training data, we are obliged to use a restricted form of the covariance matrix. We recognized three forms of restrictions: the isotropic form, the independent variables, and probabilistic principal component analysis



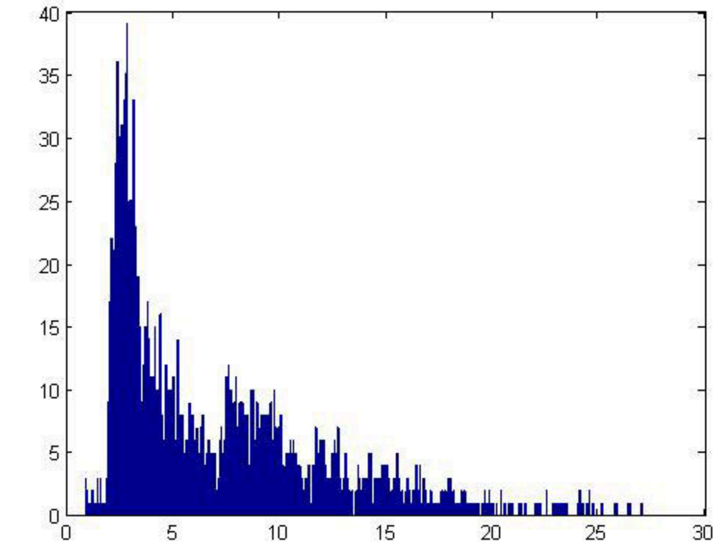


Figure 3.5: Sift feature scale distribution for all features in the bioid data set (top) and for hand-marked facial features (bottom).

bg component →	iso	iso	iso	indep	indep	indep	ppca	ppca	**ppca**
fg component →	iso	indep	ppca	iso	indep	ppca	iso	indep	**ppca**
↓ part									
bg	99	98	98	0	100	98	0	0	0
1	5	0	0	2	3	100	2	2	2
2	41	0	0	36	8	81	36	7	2
3	5	0	0	4	2	70	4	2	15
4	3	0	0	1	1	25	1	1	5
5	8	0	0	5	3	86	5	2	1
6	1	0	0	0	2	0	0	1	17
7	8	0	0	8	5	90	8	4	1
8	2	0	0	1	2	67	1	1	1
9	2	0	0	1	2	0	1	1	2
10	6	0	0	5	2	100	5	1	1
11	50	0	0	2	1	100	2	1	44
12	32	0	0	5	1	80	5	1	3
13	6	0	0	2	1	100	2	1	4
avg on parts	13	0	0	6	3	69	6	2	8

Table 3.7: **Precision** (%) for part detection on the **bioid** data set.

(ppca) as introduced by [19]. We tested different combination of the restriction applied to the background component and the part components. From the results we concluded that using a least restricted form, ppca, which preserves most of the structure in the data, does not necessarily lead to better detection performance. Indeed, on the bioid data set using both the isotropic restriction for both the background component and the part components leads to a far better performance than expected, although this configuration degrades the Bayesian detector to detection based on the Euclidian distance of the descriptor in the descriptor space. However, the recall of iso/iso combination is still too low to be usable, such that our only option is to use the ppca/ppca combination for the bioid data set. For the pose data set we will use the indep/indep combination because it yield the best recall, precision combination. Now that we have shown how candidates for the facial parts are found, we need a method to find the best combination among the set of candidates for each part, leading to the detection of the face. This is achieved by means of applying spatial constraints through the use of a planar model, which is the topic of the next chapter.

Chapter 4

Face detection & pose estimation

In the previous chapter we described the process of selecting and detecting parts of a face. The output of that process is a set of candidate features for each part. In this chapter we employ a planar spatial model to find the optimal selection of features for each part from their respective candidate sets. Such a selection is expressed by a vector called the *hypothesis*. In the experiments, we use the MoG Bayesian classifier from the previous chapter to construct the candidates sets, but the material covered here does not depend on the particular type of part detectors.

The face can be present in the image under different poses, however the actual pose is unknown. By defining the spatial model with respect to a reference pose, we are able to detect the pose and to model the location of the facial features independent of the pose of the face in the image. The pose is defined as a linear map relating the part location means in the reference pose to the observed features in the image. In chapter two this linear map was a perspectivity in order to extract the 3D pose from the image. For reasons explained later, when searching for the optimal hypothesis we weaken the pose to an affine map. This chapter is based on work by Burl *et al.* presented in [1] and by Pham, presented in [14]. The use of appearance and scale in addition to location to detect objects is discussed in Section 4.1.1.

4.1 The spatial model

We start with some notation. An image is represented by a set $\mathcal{F}$ of detected features. Each feature is represented by its location: $\mathcal{F} = \{\mathbf{x}_1, ..., \mathbf{x}_{|\mathcal{F}|}\}$. The set of candidates for part p is denoted by $\mathcal{C}_p$. Each candidate $c \in \mathcal{C}_p,\ c \neq 0$ is an index to a feature location $\mathbf{x}_c \in \mathcal{F}$. For example, suppose we have 43 detected features $\mathcal{F} = \{\mathbf{x}_1, ..., \mathbf{x}_{43}\}$, then $\mathcal{C}_2 = \{0, 11, 5, 2\}$ denotes that part two has three candidate locations. The zero is used to represent the situation where the part is not observed (for example, due to occlusion), and is member of each candidate set. A hypothesis $\mathbf{h}$ is an assignment of a candidate to each part from its respective candidate set. By construction, each entry in this vector corresponds to a part in the model and the value at each entry is an index to a feature detected in the image. This means that if $h(p) > 0$, then $h(p)$ is an index to a feature with location $\mathbf{x}_{h(p)}$. The candidate sets $\mathcal{C}_p$ dictate which values are allowed at $h(p)$. We use $\mathcal{C}_{bg}$ to denote the set of background features. Background features are features not assigned to any part by a hypothesis. So, $\mathcal{C}_{bg}$ contains all features in $\mathcal{F}$, except those referenced in the current $\mathbf{h}$. Note that while $\mathcal{F}$ is static, $\mathcal{C}_{bg}$ differs for each hypothesis.

The spatial model describes the spatial distribution of the parts in a probabilistic way: $p(\mathbf{h}, L, \mathcal{F})$ denotes the probability of observing hypothesis $\mathbf{h}$, pose L and feature locations $\mathcal{F}$. We assume the following dependence structure between $\mathbf{h}$, L and $\mathcal{F}$:

$$p(\mathbf{h}, L, \mathcal{F}) = p(\mathbf{h})\, p(L)\, p(\mathcal{F}|\mathbf{h}, L), \tag{4.1}$$

$$\text{with} \quad p(\mathcal{F}|\mathbf{h}, L) = \prod_{p:\ h(p)>0} p(\mathbf{x}_{h(p)}|\, L) \prod_{f \in \mathcal{C}_{bg}} p(\mathbf{x}_f | f \in \mathcal{C}_{bg}). \tag{4.2}$$

Note that we assume the location of the parts and background features to be independently distributed. However, due to the symmetric structure of the face, the position of the parts is in fact often correlated. For example, when smiling, the left and right corner of the mouth move simultaneously up or down. However, we do not believe that modelling this interdependence would drastically improve the model's performance. The model is further constructed using the following assumptions, which are discussed in more detail below.

- The pose is defined by an affine linear map L:

$$L = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \end{bmatrix}, \quad \text{with reference pose} \quad \text{Ł}_{ref} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

- The location of a part with respect to the *reference pose* is independently distributed according to a Gaussian pdf:

$$p(\mathbf{x}_{h(p)}|\, L_{ref}) \sim G(\mathbf{x}_{h(p)}; \boldsymbol{\mu}_p, \Sigma_p),$$

- The location of a part *observed in the image* is distributed according to the same Gaussian pdf, but with the means transformed[1] by pose L:

$$p(\mathbf{x}_{h(p)}|\, L) \sim G(\mathbf{x}_{h(p)}; L\tilde{\boldsymbol{\mu}}_p, \Sigma_p),$$

- The locations of features not assigned to any part are background features and are independently and uniformly distributed:

$$p(\mathbf{x}_f|\, f \in \mathcal{C}_{bg}) \sim (\text{image area})^{-1},$$

- Each part has a prior probability w_p of being present,

- The probability of each part being present or not is independent of the presence of the other parts, so that the probability of hypothesis $\mathbf{h}$ is given by[2]

$$p(\mathbf{h}) \sim k \prod_{h(p)>0} w_p \prod_{h(p)=0} 1 - w_p, \tag{4.3}$$

[1] $\tilde{\boldsymbol{\mu}}_p = [x,\ y,\ 1]^\top$, ie. $\boldsymbol{\mu}_p$ augmented with a final 1, which is needed to apply the pose.

[2] here, k is a normalization constant. It is actually never needed, but may be computed as follows: if we sum over all hypotheses, each part will contribute $(|\mathcal{C}_p| - 1)w_p + (1 - w_p)$ to this sum. For a model with one part, the normalization constant would be $k_p = [\,(|\mathcal{C}_p| - 1)w_p + (1 - w_p)\,]^{-1}$. For a model with P parts k is therefore the product of all k_p: $\quad k = \prod_p k_p = \prod_p [\,(|\mathcal{C}_p| - 1)w_p + (1 - w_p)\,]^{-1}$.

- The pose is uniformly distributed: $p(L) = c$.

The reason we use an affine transformation to model the pose when detecting the face is that the 3D pose estimate takes on average about a second to compute. Because we have to evaluate lots of hypotheses, even a second is prohibitively long. For this reason we approximate the 3D pose with an affine pose, which also fits better into our probabilistic spatial model. Note that we do not transform the covariance matrix. As a consequence the covariance matrix describes the noise in the candidate locations encountered in the image, and not in the reference positions, which would be preferable. The reason why we do not transform the covariance matrix is due to the estimation of the pose, which is done by computing L for which $p(C|\mathbf{h}, L)$ is at a maximum. A closed form solution for to computing this maximum is analytically intractable if the covariance matrix is transformed by L as well. Transforming the means only allows the computation of the pose with maximum likelihood in closed form.

Putting everything together, we obtain

$$p(\mathbf{h}, L, \mathcal{F}) \propto O^{-r} \prod_{p:\ h(p)=0}^{P} (1 - w_p) \prod_{p:\ h(p)>0}^{P} w_p\, G(\mathbf{x}_{h(p)};\, L\tilde{\boldsymbol{\mu}}_p, \Sigma_p) \tag{4.4}$$

where O is the image area and $r = |\mathcal{C}_{bg}|$ is the number of background features. Eq. (4.4) is proportional to Eq. (4.1), because we left out the scalars $p(L) = c$ and k. We may use this proportional form of $p(\mathbf{h}, L, C)$, because we only need to find the maximum of this function.

Taking a closer look at this equation, we see that if the probability of the optimal position of a part is smaller than the probability of the part not being present, that is

$$\frac{w_p}{2\pi|\Sigma_p|^{\frac{1}{2}}} < \frac{(1 - w_p)}{O}, \tag{4.5}$$

the part will never get an feature assigned. In this case it is always more likely that the part is not present, see Fig. 4.1. It does not make much sense to keep such part in the model (it is just a constant), so we should expunge every part in the model adhering to this condition. We did not encounter this situation in the trained models.

4.1.1 Modelling the appearance

Besides relative feature location, we may also want to take other observed aspects of the features into account to increase the model's flexibility. For example, although we used appearance (in chapter 3) to create the candidate sets for each part, we may still want to take the appearance into account when evaluating a hypothesis likelihood. A very convincing appearance of the correct feature for a certain part can in this way make up for worse relative location and thereby still being favored over other (incorrect) candidates with a better relative position. We experimented with this by using

$$p(\mathbf{h}, L, \mathcal{F}) = p(\mathbf{h})\, p(L)\, p(A|\mathbf{h})\, p(\mathcal{F}|\mathbf{h}, L), \tag{4.6}$$

instead of Eq. (4.1). In this equation $p(A|\mathbf{h})$ denotes the probability of observing the feature descriptors given a hypothesis, which was modelled by a mixture of Gaussian as described in chapter three. Besides feature appearance, feature scale can also be taken into account in this way. This is, for example, done in [3], [7] and [24]. It turned out, however, that doing this

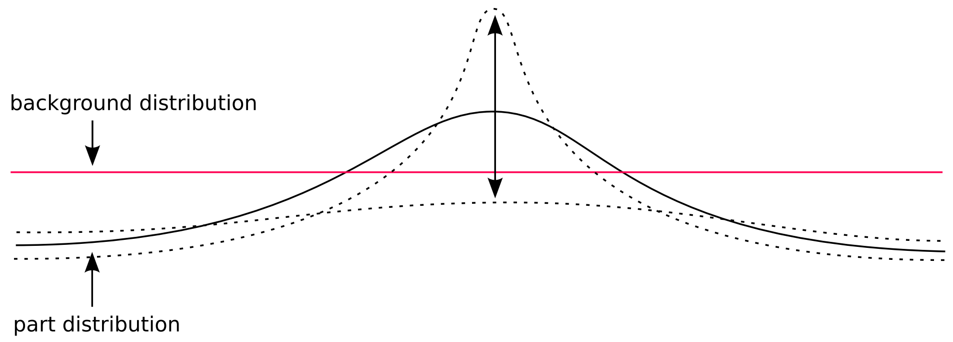


Figure 4.1: If the probability of the location of a part, multiplied with its prior probability of being present (part distribution), is *always* smaller than the likelihood of a background feature, multiplied by the probability of the part *not* being present (background distribution), the part should be expunged from the model.

leads to a problem: due to the general high dimensionality of appearance vectors, the scale of the resulting appearance probabilities can easily destroy the effect of the spatial model. The reason for this is that for high dimensional data the probabilities are typically orders of magnitude smaller than probabilities of low dimensional data. In [3] this issue is tacitly circumvented by computing the principal components of the set of all part appearances and using the first 10-15 components to project newly observed appearances to a lower dimensional space. However, it is not clear how this number should be chosen such that modeling appearance and location together yields better results than first selecting candidates based on their appearance and using location to find the best matching. For this reason, we do not take the appearance into account when applying the spatial model. The appearance is only used to construct the candidate sets.

4.2 Estimating the pose

Given an image, we are interested in the (hypothesis, pose) combination with highest probability given the observations:

$$\operatorname*{argmax}_{\mathbf{h},L} [\ p(\mathbf{h},L|\mathcal{F})\ ] \ = \ \operatorname*{argmax}_{\mathbf{h},L} [\ p(\mathbf{h},L,\mathcal{F})\ ]\,, \tag{4.7}$$

where the right hand side follows from the fact $p(\mathcal{F})$ is constant for all $\mathbf{h} \in \mathcal{H}, L \in \mathcal{L}$. Filling in equation Eq. (4.1), we arrive at

$$\operatorname*{argmax}_{\mathbf{h},L} [\ p(\mathbf{h},L|\mathcal{F})\ ] \ = \operatorname*{argmax}_{\mathbf{h}} \left[\ p(\mathbf{h}) \ \ \operatorname*{argmax}_{L} [\ p(\mathcal{F}|\mathbf{h},L)\ ] \right]. \tag{4.8}$$

For a given hypothesis, the pose is therefore computed using

$$\operatorname*{argmax}_{L}\, p(\mathcal{F}|\mathbf{h},L) = \operatorname*{argmax}_{L} \prod_{p:\ h(p)>0} G(\mathbf{x}_{h(p)};\, L\tilde{\boldsymbol{\mu}}_p, \Sigma_p), \tag{4.9}$$

which follows from Eq. (4.4). The maximum argument of Eq. (4.9) is found by differentiating the logarithm of the product of the Gaussians functions with respect to L and equating the result to zero. Details on this computation are found in Appendix C.

4.3 Finding the best hypothesis using A*

To detect the face we need to find $\mathbf{h}$ for which Eq. (4.1) is at a maximum, which follows from Eq. (4.8) (computing the optimal pose for every explored hypothesis). The number of values $h(p)$ can take is $n_p = |\mathcal{C}_p|$. This means that the size of the hypothesis space is exponential in the number of parts:

$$|\mathcal{H}| = \prod_p n_p \geq [\min_p \; n_p]^P. \tag{4.10}$$

As a result the size of the hypothesis space rapidly becomes of intractable size. A 13-part model, with 2 candidates per part, already results in a hypothesis size of nearly $3^{13} \approx 1,6 \times 10^6$ (each part has three possible values: two candidates plus the zero). As shown the previous chapter, we can expect about 12 candidates on average on images from the bioid data set! Clearly, this renders exhaustive search through the hypothesis space intractable in almost all situations. Using heuristic search we are able to significantly reduce the search space which needs to be explored, which is described in the rest of this section. Although heuristic search does not remove the exponential complexity of the search problem in general, it turns out to work very well in practice.

Following ideas presented in [14], we use the A* algorithm [18] to employ heuristics in searching for the best hypothesis. The A* algorithm finds the shortest path from a `start` node to a `goal` node in a directed graph. This is achieved by building a set of paths incrementally, at each stage expanding the path for which the actual cost plus an estimated cost to reach the goal node is smallest. This latter cost is referred to as a *heuristic*. In our case, a path is specified by a (partial) hypothesis $\mathbf{h}$. To let A* build hypotheses incrementally, we recognize an additional state for $h(p)$: $h(p) = -1$. We now have three types of $h(p)$:

- $h(p) = -1$: the part is not yet assigned to a value from candidate set $\mathcal{C}_p$,
- $h(p) = 0$: the part is determined to be not present,
- $h(p) > 0$: a feature is assigned to the part,

The hypothesis space is represented as a layered and acyclic directed graph, where each layer represents a part, as is shown in Fig. 4.2. Further, we let

$$z(\mathbf{h}) = \operatorname*{argmin}_p \; [\, h(p) = -1 \,] \tag{4.11}$$

denote the entry used to fork a hypothesis into a set of new hypotheses, when it is selected to be expanded. This means that hypotheses are build incrementally, starting with part one.

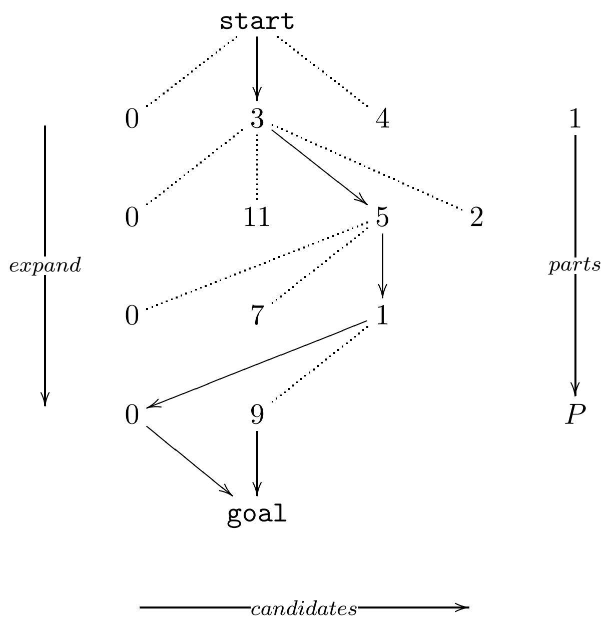


Figure 4.2: Search through the hypothesis space for a model with four parts, represented by directed graph. The solid arrows show the path corresponding to hypothesis $h = [3, 5, 1, 0]$. The dotted lines indicate the hypotheses that were explored.

For the selection of the hypothesis to-be expanded, two cost functions are used: $g()$ is the actual cost of the assigned part of a hypothesis: entries 1 to $[z(\mathbf{h}) - 1]$. The heuristic function $e()$ is a cost estimate for the hypothesis entries not yet assigned to a value, $z(\mathbf{h})$ to P. At each step, A* expands the hypothesis with smallest value for $f() = g() + e()$ by replacing entry $z(\mathbf{h})$ with each value in $C_{z(\mathbf{h})}$. The $|C_{z(\mathbf{h})}|$ new hypotheses resulting from this expansion are added to the set of current hypotheses, from which the expanded path itself is removed, and the process is repeated. As soon as a completely assigned hypothesis is selected for expansion, the algorithm terminates. Fig. 4.3 depicts the first four steps of the A* algorithm for the situation in Fig. 4.2.

We use the negative logarithm of $p(\mathbf{h}, L, \mathcal{F})$ for the actual cost function $g()$:

$$g(\mathbf{h}) = -\log O^{-b} \prod_{p:\ h(p)=0} (1 - w_p) \prod_{p:\ h(p)>0} w_p\, G(\mathbf{x}_{h(p)};\, L\tilde{\boldsymbol{\mu}}_p, \Sigma_p) \tag{4.12}$$

where $b = |\mathcal{C}_{bg}| - \text{sum}[h(p) = -1]$ is the number background features, lessened with the number of parts to-be explored. The heuristic cost function $e()$ is given by

$$e(\mathbf{h}) = -\log \prod_{p:\ h(p)=-1} \frac{w_p}{2\pi|\Sigma_p|^{\frac{1}{2}}}, \tag{4.13}$$

The particular choice for the heuristic function is explained shortly. Although the logarithm allows rewriting the product terms to sums, we retain this form here for notational conciseness.

A* is complete, optimal, and optimally efficient, provided that $e()$ never overestimates (ie. is optimistic) the actual cost to reach `goal` [18]. Such an $e()$ is called an *admissible* heuristic.

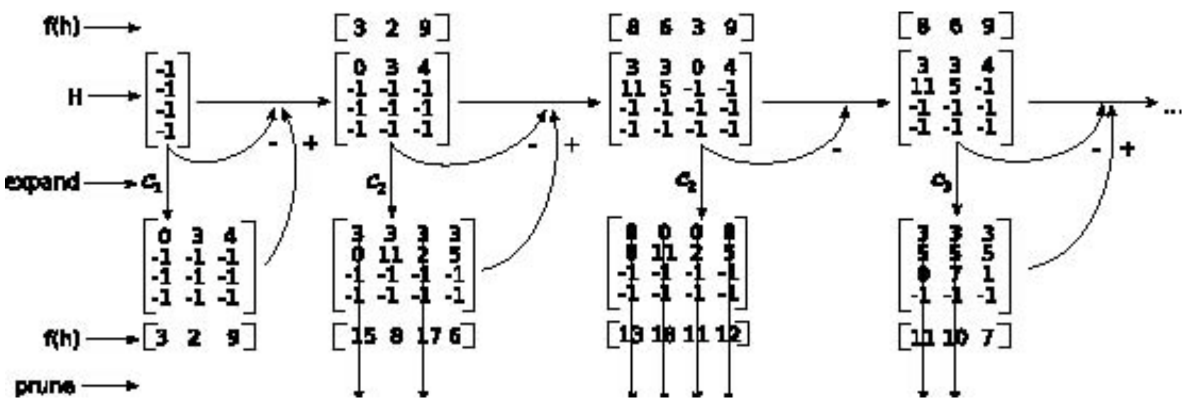


Figure 4.3: First four steps taken by the A* algorithm for an example situation. In this case it is assumed that $f(\mathbf{h}_0) = 10$.

To see why $e(\mathbf{h})$ in Eq. (4.13) is admissible, note that the form follows by letting the location of each part-to-be-explored exactly coincide with the corresponding transformed part mean, using the pose of the assigned parts. Because a Gaussian function is maximum at the mean, the negative logarithm of a Gaussian is minimal at the mean. As a result this is the best possible location for a part, and so $e()$ can never overestimate the actual cost.

A* is complete in the sense that if there exist a solution, it will always find it. Optimal refers to the property that it finds the highest-quality solution. This is the property we really enjoy because any path (hypothesis) leads to a solution in our case. Finally, A* is optimally efficient, which means no better algorithm with same the knowledge performs better.

Note that the cost function $g(\mathbf{h})$ is only defined for the assigned part of the hypothesis *as a whole*, ie. it *cannot* be build from the addition of the cost of the individual steps, because adding a part changes the estimated pose L_m (see Section 4.2), and thereby it can increase the 'path length' of previous steps. Although A* is often applied to topographic problems (at least in examples), where the actual path cost often *is* the addition of its individual steps, the algorithm does not requires this. For A* to work it needs only to be able to evaluate the cost of the actual path.

Hypotheses are pruned from the current set of hypotheses when the cost $f(\mathbf{h})$ is greater than the background hypothesis $\mathbf{h}_0 = \mathbf{0}$ which consists of only zeros. Of course such hypotheses with cost higher than that of the background hypothesis will never be selected for expansion. However, because there is potentially a large amount of such hypotheses, they use a significant amount of memory and administration and are therefore better removed. Fig. 4.4 summarizes the algorithm.

4.4 Estimating the model parameters

Using the labelled data from the pose and bioid data set, used in the last chapter, we obtain two sets of the spatial model parameters. The procedure is outlined in Fig. 4.5. The part priors are obtained from Tab. 3.3.2. The spatial models obtained for the data sets are depicted in Fig. 4.6. These models are used in the experiments described below. All training data was used to construct the spatial models. Although using the same data for training as well as

```
Objective Find hypothesis $\mathbf{h}_{opt}$ with maximum a posteriori probability (MAP),
    using heuristic search method A*.

Input model $m = \{\Sigma_1, \boldsymbol{\mu}_1, w_1, \ldots, \Sigma_p, \boldsymbol{\mu}_p, w_p\}$, candidates $\{\mathcal{C}_1 \cup \mathcal{C}_2 \ldots \cup \mathcal{C}_P\}$, image
    area $O$

  1. initialize hypothesis space $\mathcal{H} = \{-\mathbf{1}\}$
  2. select $\mathbf{h}_i \in \mathcal{H}$ for which $f(\mathbf{h}_i) = g(\mathbf{h}_i) + e(\mathbf{h}_i)$ is minimal
  3. if $\mathbf{h}_i$ is not completely assigned set $q = z(\mathbf{h}_i)$
     (a) for each $c \in \mathcal{C}_q$
           i. set $\mathbf{h} = \mathbf{h}_i,\ h(q) = c$
          ii. if $f(\mathbf{h}) < f(\mathbf{h}_0)$ $\mathcal{H} = \mathcal{H} \cup \mathbf{h}$ else prune $\mathbf{h}$
     (b) $\mathcal{H} = \mathcal{H}/\mathbf{h}_i$, goto step 2
  4. else return $\mathbf{h}_{opt} = \mathbf{h}_i$
```

Figure 4.4: Heuristic hypothesis space search algorithm.

testing does not give a good indication of the performance of the method in general, we did not split up the data because the aim of the experiments is to provide a proof of concept of the presented algorithm. Note that spatial model of the pose data set has very tight parameters, for example part ten allows almost not vertical variation in the image part location. The reason that the spatial model of the pose data set is much more tight than the spatial model of the bioid data set, is that it is estimated using image of a single person (the author), while the bioid data set contains many different subjects.

4.5 Experiments

4.5.1 Detecting faces

To investigate the capability of the method to detect faces of different subjects in a realistic setting, the method was tested on the bioid data set, described in the previous chapter. No calibration matrix is available for this data set, so the 3D pose of the faces can not be estimated in this experiment. Because the used part classifier (the ppca/ppca MoG bayesian classifier, see chapter three) has a very low precision, many candidates for each part are returned.

4.5.2 Results and discussion

In Fig. 4.7 the candidates returned for two example images are depicted. As a result of the number of candidates for each part, the search algorithm had to deal with very large hypotheses spaces, containing on average 6.7×10^7 hypotheses. As a result of the heuristic search, not more than 0.5 % of the hypothesis space needed to be explored. The time to conduct this search took on average 145 seconds, with peaks up to 20 minutes[3]. The cumulative number

[3]The experiments were conducted on a laptop computer with a 1800 MHz Intel pentium 4 processor and 512mb ram.

Objective Estimate spatial model parameters $\theta = \{\boldsymbol{\mu}_1, \Sigma_1, ..., \boldsymbol{\mu}_p, \Sigma_p\}$

Input Labelled data ($\mathbf{h}$ is known for each image)

1. **Model means**. Choose master image, translate part positions to zero mean. Set $\forall_{p:\, h(p)>0}\, \Sigma_p = \lambda I$, for any $\lambda \neq 0$. This is the *initial model*. If the pose is to be estimated, scale the model means to appropriate size, such that the distances between the parts in the model correspond to the distances between the facial features measured directly on the face.
2. Estimate the pose of each image in the data set using the parts present in both the new image and the initial model (Section 4.2).
3. Use the pose of each image to map the part location to reference pose.
4. Compute part means of the model by averaging the transformed location for each part.
5. **Image covariance matrices**. Use part means to estimate the pose of each image.
6. Transform the means using the obtained pose for each image.
7. Compute the image covariance matrices the location of the transformed part means and observed part locations in each image.

Figure 4.5: Procedure to estimate model parameters.

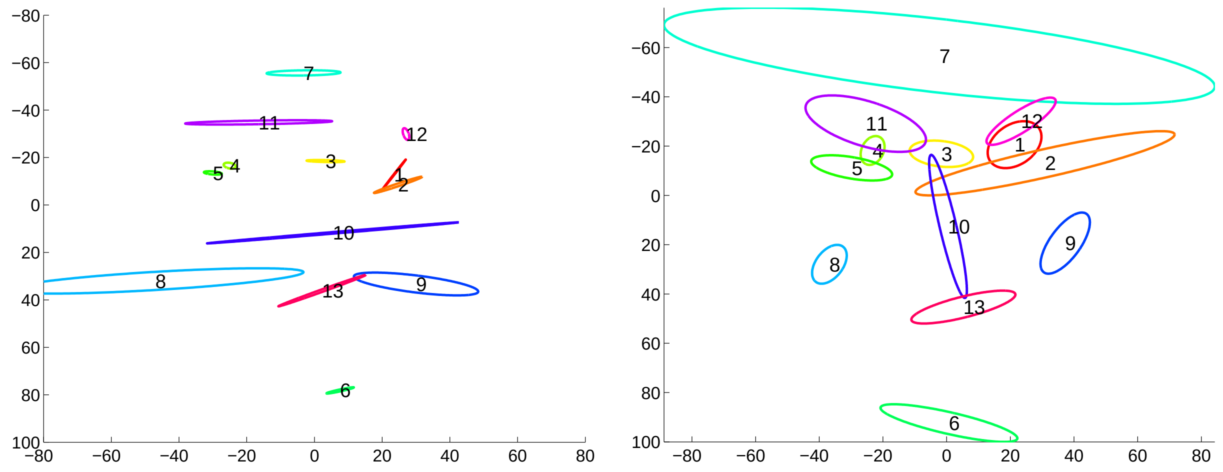


Figure 4.6: Spatial model for the pose data set (left) and the bioid data set (right). Consult Tab. 3.3.2 on page 26 for a legend of the parts.

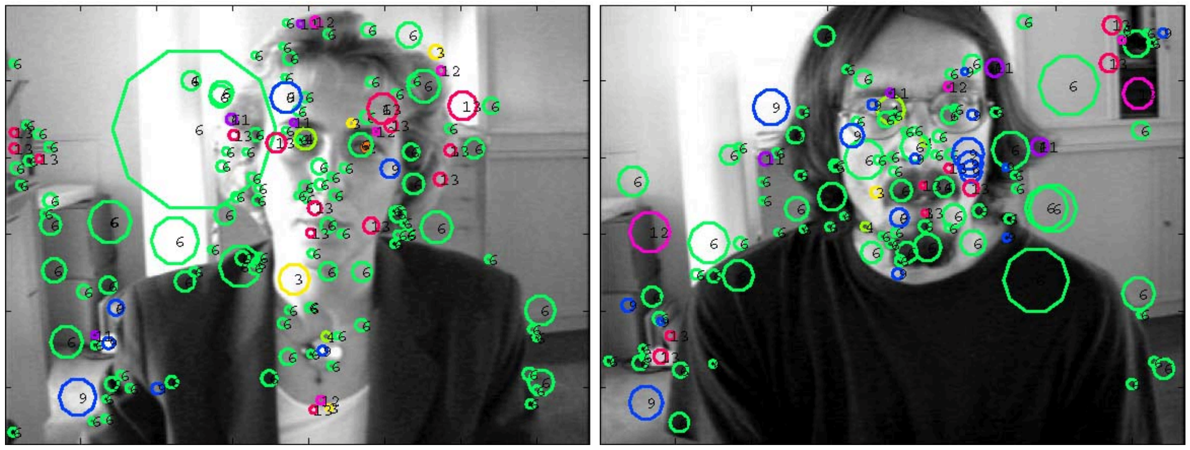


Figure 4.7: Detected candidate features for each part for two images from the bioid data set.

Category	Percentage
Correct detections	30%
Detections 'close' to correct	19%
False detections	50%

Table 4.1: Percentage of the face detections on 304 images from the bioid data set, falling in each of the three detection categories.

of correctly returned hypothesis entries is shown in Fig. 4.8. Note that not once the fully correct hypothesis was returned.

Because Fig. 4.8 says little about the quality of the detections, each detection was categorized into one of three classes by visual inspection: correct detections, detections 'close' to being correct, and false detections. The middle category is used because it is sometimes hard to draw the line between correct and false detections. Percentages of the detection falling into each category are given in Tab. 4.1. Examples of each category are given in Fig. 4.9 - Fig. 4.11.

From this experiment we conclude that the method is capable of face detection in realistic settings and can even cope with face artifacts such as glasses, beards and mustaches. However, the computational time needed to perform a detection is prohibitively long. This a direct consequence of the low precision of the part classifier. A better part classifier, taking for example color information into account as well, should be able to have a much higher precision, leading to faster detections.

4.5.3 Detecting the face and estimating the face pose

This experiment puts all the pieces together on the pose data set. The true pose of the faces in the pose data set were estimated by hand to obtain ground truth of the performance of the method. The poses where obtained by letting the person in the images (the author) look at a grid of spots on the wall with a fixed distance to the camera. The pose of the face in each image was accordingly fined-tuned through visual inspection. This fine tuning was needed

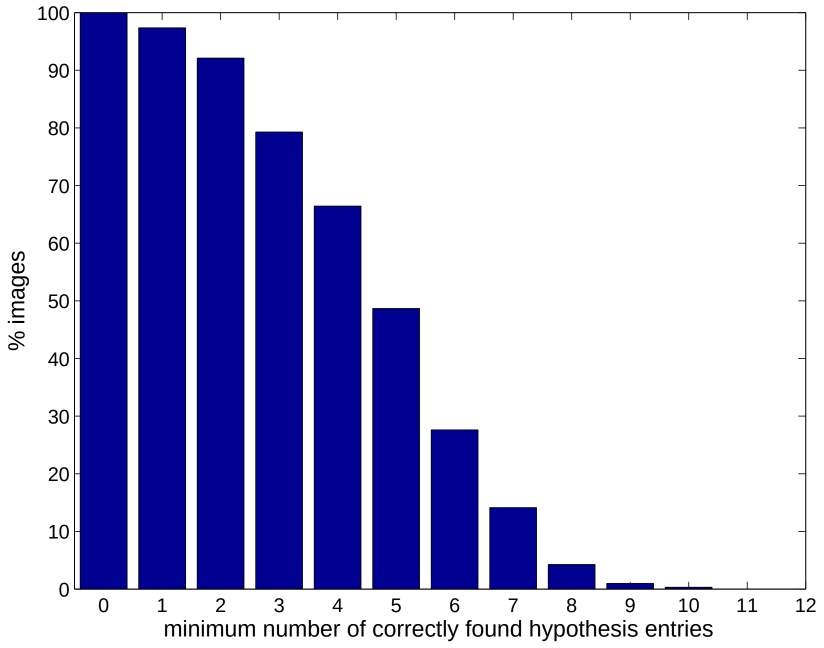


Figure 4.8: Percentage of images from the bioid data set for which a minimum number of hypothesis entries are correctly found. Noted that the true hypothesis entries are zero about 40% of the time (see Tab. 3.3.2, page 26).

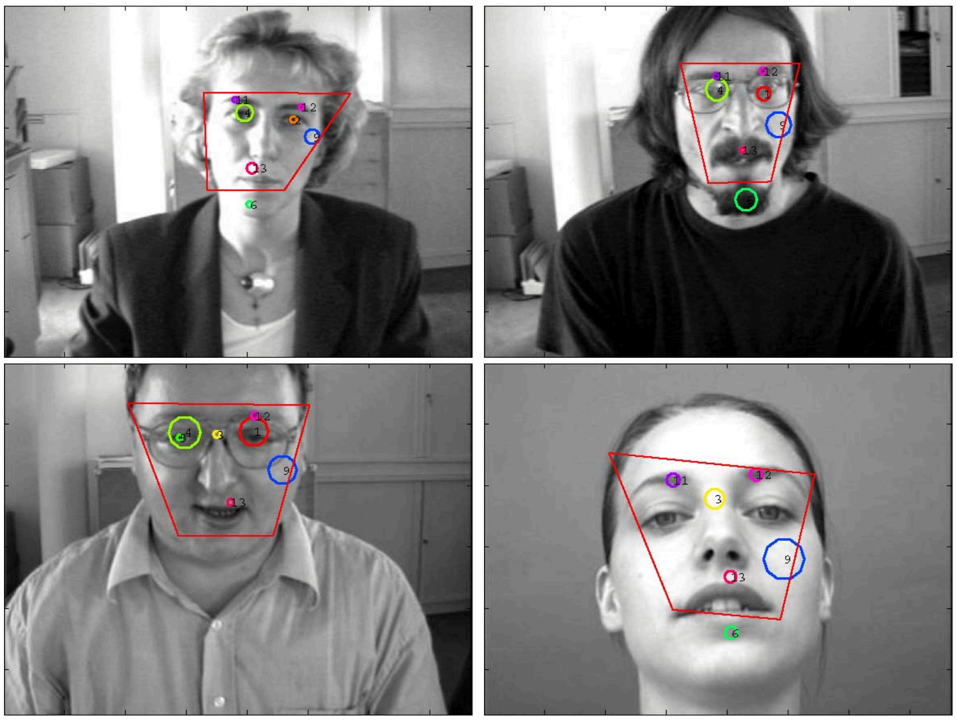


Figure 4.9: Examples of correctly detected faces for the bioid data set.

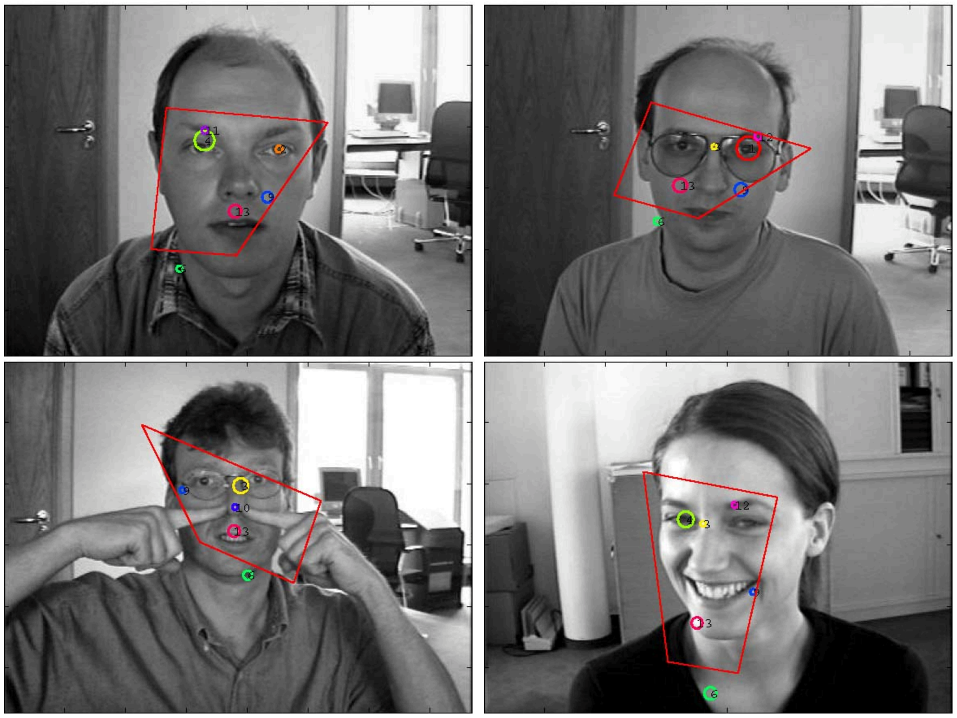


Figure 4.10: Examples of face detections somewhat 'close' to being correct for the bioid data set.

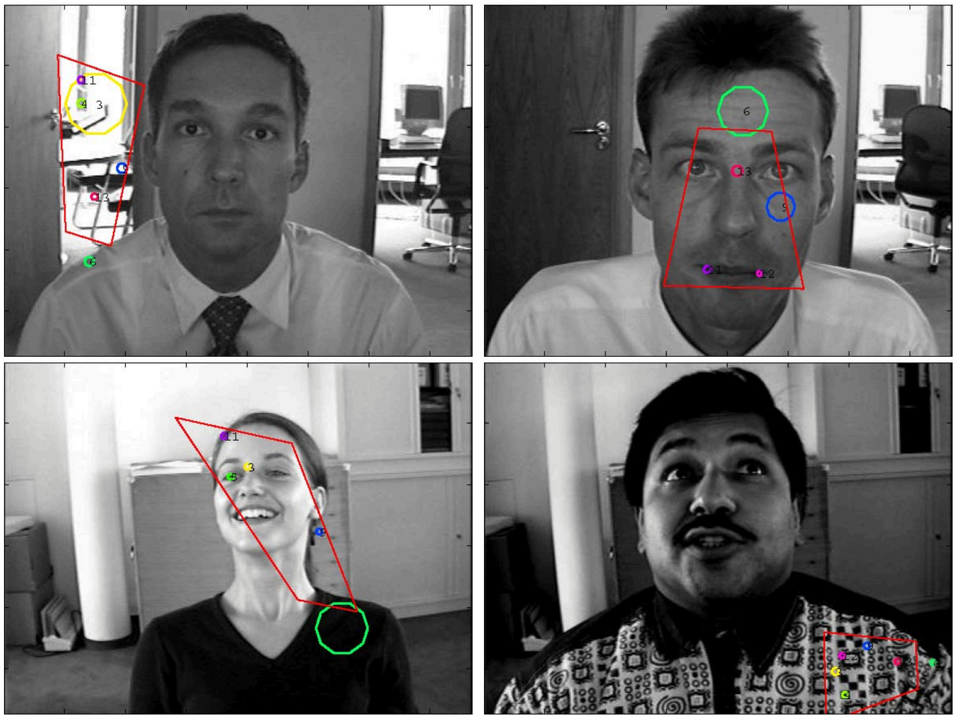


Figure 4.11: Examples of false face detections for the bioid data set.

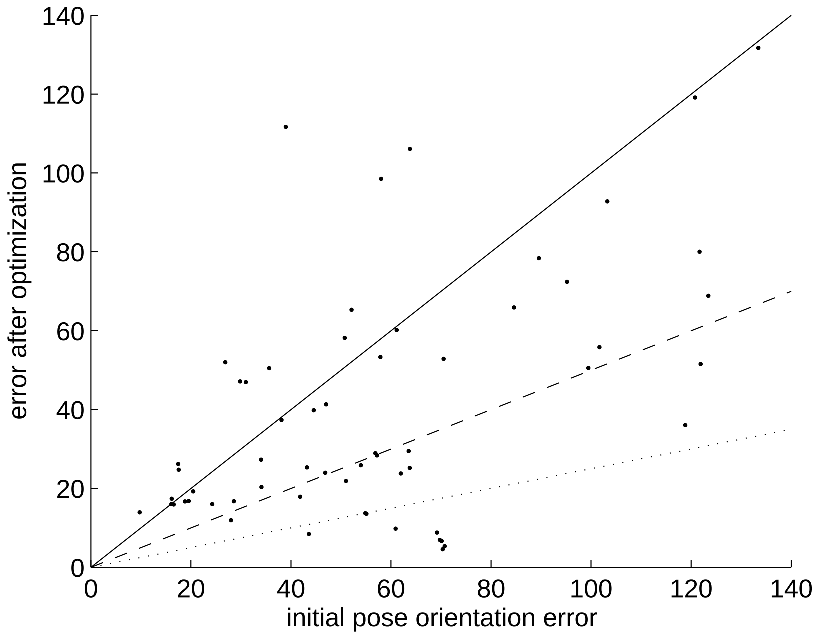


Figure 4.12: Scatter plot of the error in the estimated face poses as the angle between the true plane normal in the image and the estimated plane normal before optimization (x-axis) and after optimization (y-axis), for the pose data set. Dots below the solid line indicate pose estimates that were improved by the optimization routine. The error of the estimated poses below the dashed line are reduced more than two times; poses below the dotted line have a more than four times lower error. Note that in three cases the 'optimized' pose estimate is actual much worse than the initial pose, and in ten cases the optimization produced slightly worsened estimates.

because it turned out to be hard to use only the head to look at the spots; most of the time a combination of head rotation and eye rotation were used.

4.5.4 Results and discussion

The process of estimating a face pose given an image and camera parameters, as described throughout this thesis is visualized for three images in Tab. 4.2. As can be seen, the number of candidates is much less for this data set, and as a result the time needed to detect the faces was at maximum a few seconds. The time the LM optimization routine needs to converge is under a second for the correctly estimated poses. Fig. 4.12 shows the error in the estimated face poses as the angle between the true plane normal in the image and the estimated plane normal before optimization (x-axis) and after optimization (y-axis), for all 60 images in the pose data set. As is seen, the optimization serves its purpose, although sometimes the pose estimated is worsened. The estimated pose for some images are depicted in Fig. 4.13 - Fig. 4.15.

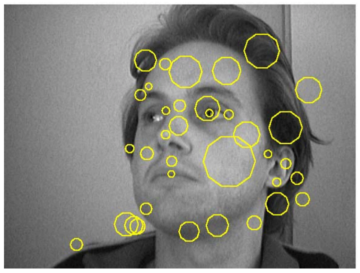
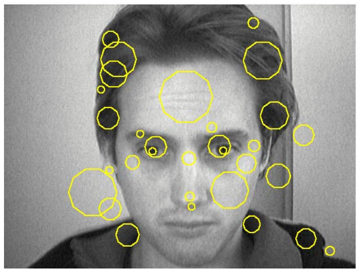
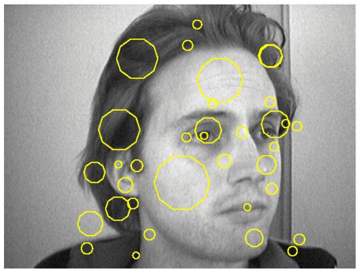
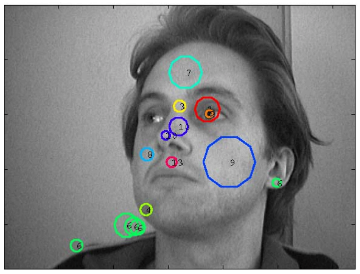
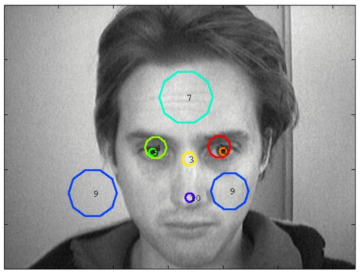
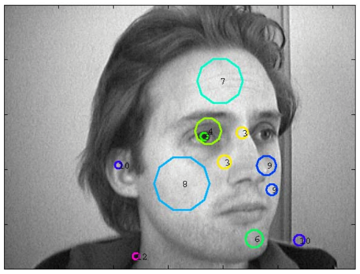
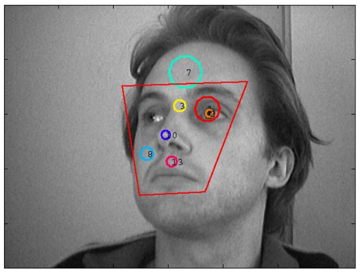
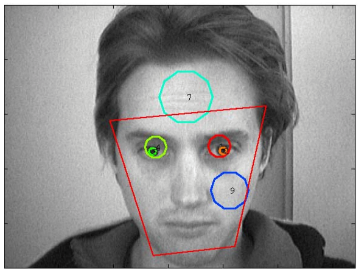
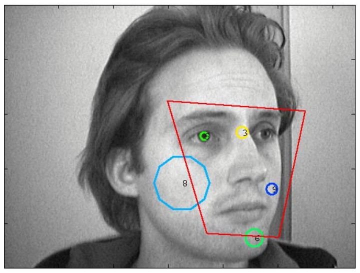
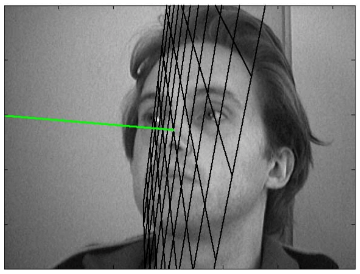
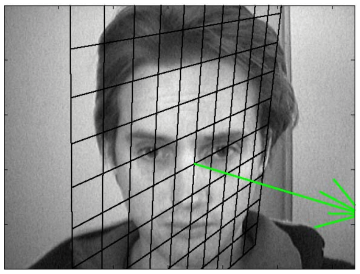
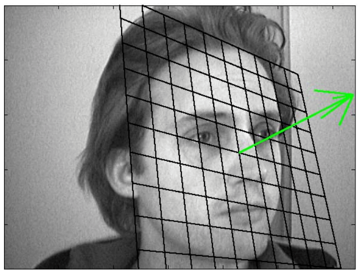
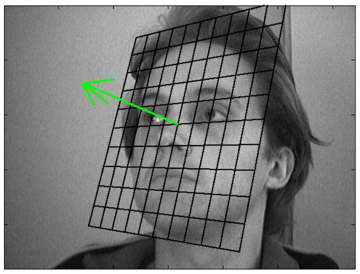
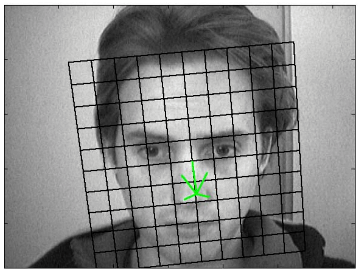
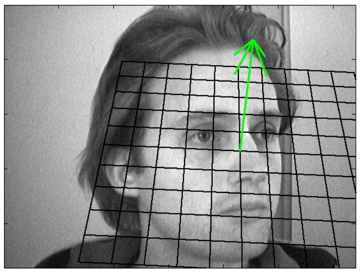
image 20	image 52	image 23
↓	Detect ↓ features	↓
		
↓	Classify ↓ features	↓
		
↓	Find best ↓ matching	↓
		
↓	Estimate ↓ initial pose	↓
		
↓	Optimize ↓ initial pose	↓
		
↓	↓	↓
pose	pose	pose

Table 4.2: Steps taken to estimate the face pose, visualized for images 20, 52 and 23.

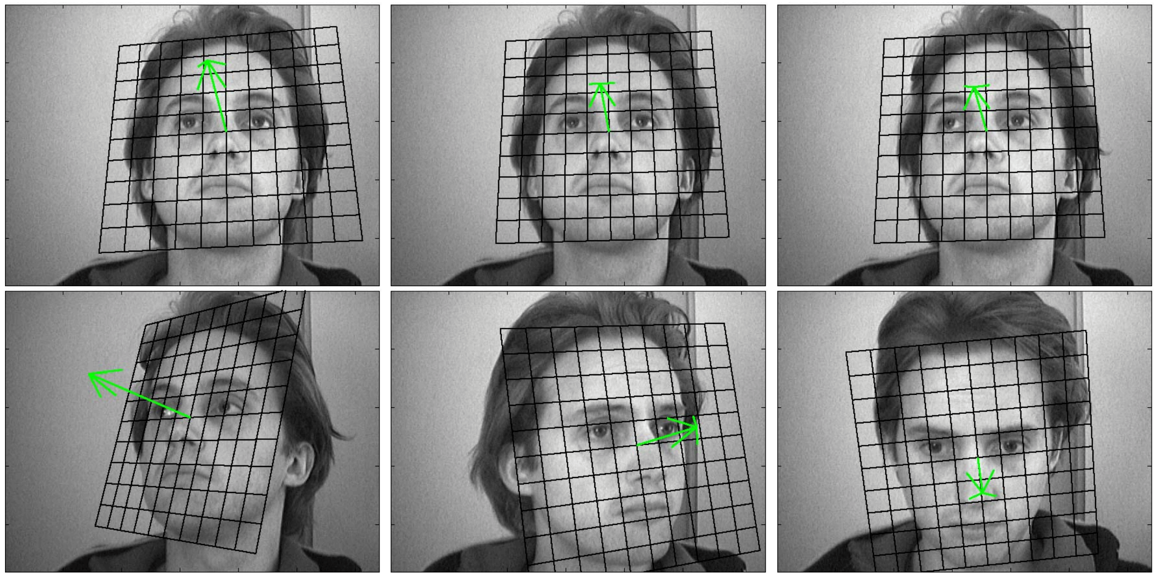


Figure 4.13: Correctly estimated poses (pose data set). Images 11, 13, 15 (top) and 20, 24, 52 (bottom).

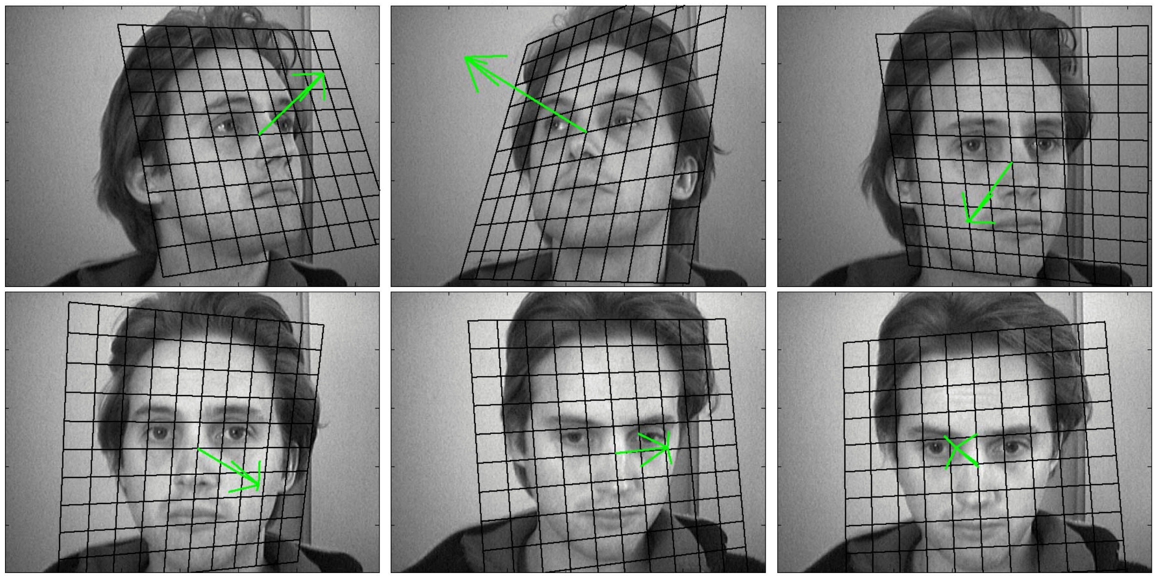


Figure 4.14: Pose estimates getting close (pose data set). Image 2, 17, 26 (top) and 33, 47, 51 (bottom).

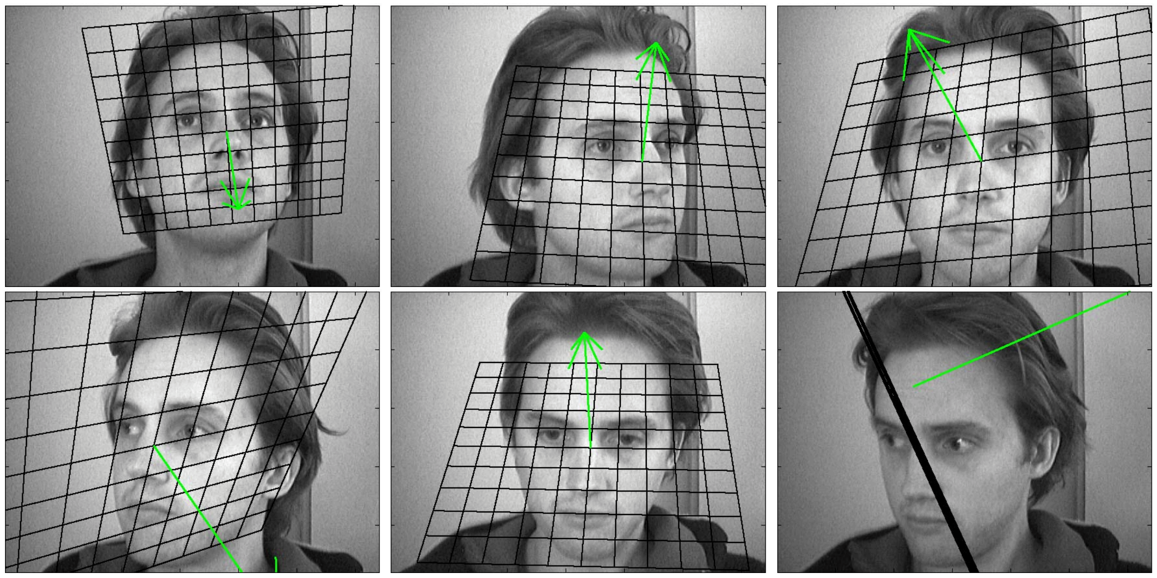


Figure 4.15: Wrong pose estimates (pose data set). Image 10, 23, 32 (top) and 39, 50, 60 (bottom). Note that, except for image 60, the center of the face is still correctly found.

4.6 Conclusion

We showed how a spatial planar model of the part locations is used to find the best match between model parts and features in the image. The method was tested on the bioid data set to investigate its capabilities of detecting faces in a realistic setting. The results are promising, but the computational time needed to detect a face is prohibitively long. The main cause of this deficit is the low precision of the part classifier.
Estimation of the pose was tested on the pose data set. The results provide a proof of concept on the use of a planar model to detect the face and estimate the face pose in an integrated manner.

Chapter 5

Conclusion

We have investigated the use of a part-based planar model of Sift features to detect faces and estimate the face pose in an integrated manner. Evidence was provided on the applicability of this approach. However, better part classifiers are needed to allow face detections to be fast and robust enough for the system to usable in practice. Summarized, the contribution of this work to the area of face detection and face pose estimation consists of:

- Affirming the applicability of the Sift detector to facial part detection,
- Detecting facial parts using a mixture of Gaussians Bayesian classifier,
- Detection of faces invariant to affine transformations of the part locations, using a spatial planar model and heuristic search through the hypothesis space,
- Computing the 3D face pose from planar point correspondences, found as a result of the face detection process.

Considering the work presented in this thesis, two questions remain. First, the presented routine to estimate the 3D pose of the face is unsatisfactory because it is partially based on a optimization routine. As a result even a considerable amount of computational time does not necessarily provide a good estimate of the pose, due to local optima or bad initial estimates. A closed form solution to the face pose estimation problem Eq. (2.5) is therefore highly desirable. The second question involves the integration of the appearance model with the spatial model. As pointed out in Section 4.1.1 it remains unclear how the probabilistic appearance and location model of the parts should be integrated to yield be better results than first selecting candidates based on the appearance and using location find the optimal constellation of candidates for given a spatial model.

5.1 Future research

The use of color information instead of just intensity values could play an important role in obtaining a facial part detector with higher precision. The rationale behind this idea is that the color of the skin in images is often only present on faces (and other human body parts), and therefore easily distinguished from non-face structures. Future research could also include the automatic construction of the planar model, such as is done in [23], [3], [7], which we omitted because of lack of time. Advantages of the automatic construction of the planar model are the use of potentially more parts and the automatic insertion and deletion of parts when presented with new images. Thirdly, because a typical application of face pose estimation is found in human computer interaction, the change observed in the face pose over time will be continuous and local in nature. This suggests the use of a predictor-corrector schema such as the celebrated Kalman filter [10] to improve the pose estimate and perhaps guide the hypothesis search such that faster detection is accomplished.

5.2 Acknowledgements

I am most grateful to the people at IAS for the support and encouragement I received while conducting my research project and while writing this thesis. I particular, I would like to thank J.J. (Sjaak) Verbeek for his patience, insights and the many pleasant talks we had about my research topic. Thanks Sjaak! Finally, I would like to show my gratitude to all my fellow students for cheering me up and for reminding me to eat once in a while. Thanks everyone!

Appendix A

Estimating the descriptor MoG parameters

A multivariate Gaussian probability density function is given by:

$$G(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)}. \tag{A.1}$$

The rest of this section provides the maximum likelihood estimators for the prior, mean and the different forms of restriction on the covariance matrix covariance matrix. We start with some notation.

The set of all detected features in an image is denoted with $\mathcal{F}$. The appearance of feature $f \in \mathcal{F}$ is described by its descriptor, denoted $\mathbf{a}_f \in \mathcal{R}^{128}$ returned by the Sift-detector (together with the feature location $\mathbf{x}_f$, and scale $\mathbf{s}_f$ which are not used here). For each part $p \in \{1, ..., P\}$ we let A_p denote the set of all descriptors representing that part in *all* images in the training set. The set of all features in *all* images in the training set is denoted by A. Note that, since each training image contains at most one feature per part, but many background features, A_0 will be several orders of magnitude greater than A_p.

The component priors π_p are estimated by

$$\pi_p = \frac{|A_p|}{|A|}. \tag{A.2}$$

Note that the priors for the parts will be small compared to the prior of the background, because of the abundance of background features in each image.
The means $\boldsymbol{\mu}_p$ of the components are estimated by

$$\boldsymbol{\mu}_p = \frac{1}{|A_p|} \sum_{f \in A_p} \mathbf{a}_f. \tag{A.3}$$

A.1 Isotropic covariance matrix

The maximum likelihood estimator for the isotropic covariance matrix is given by:

$$\Sigma_p = sI, \quad \text{with } s = \frac{1}{128}\sum_{i=1}^{128}\sigma_i^p, \tag{A.4}$$

where σ_i^p is the variance of the ith variable of the observation in A_p:

$$\sigma_i^p = \frac{1}{|A_p|}\sum_{f\in A_p}\left[\mathbf{a}_f(i) - \mu_p(i)\right]^2. \tag{A.5}$$

A.2 Independent covariance matrix

The maximum likelihood estimator for the covariance matrix on the assumption that the variables are independently distributed is given by:

$$\Sigma_p = \operatorname{diag}(\sigma_1^p, .., \sigma_{128}^p), \tag{A.6}$$

where σ_i^p is the variance of the ith variable of the observation in A_p, Eq. (A.5). 'diag' refers to the construction of a square matrix by putting the its argument along the diagonal of the matrix and leaving the other entries zero.

A.3 Probabilistic PCA covariance matrix

Let S denote the covariance matrix of the A_p. Tipping *et al.* [19] show that the maximum likelihood estimator for W is then given by

$$W = U_k(\Lambda_k - \sigma^2 I)^{\frac{1}{2}}R, \tag{A.7}$$

where the k column vectors in the $d \times k$ matrix U_k are the principal eigenvectors of S (ie. the eigenvectors with largest eigenvalues), with corresponding eigenvalues in the $k \times k$ diagonal matrix Λ_k, and R is an arbitrary $k \times k$ rotation matrix. Note that we do not need R, because it collapses to I when computing Σ in Eq. (3.4). The maximum likelihood estimator for σ is given by

$$\sigma_p = \frac{1}{d-q}\sum_{j=q+1}^{d}\lambda_j. \tag{A.8}$$

The evaluation of Eq. (A.1) requires the inverse of Σ. Because of the high dimensionality of the data this is an expensive operation. Because $\Sigma = \sigma I + WW^\top$, the inverse can also be computed using the identity

$$(\sigma I + \Lambda\Lambda^\top)^{-1} = \sigma^{-1}I - \sigma^{-2}\Lambda(\underbrace{I + \sigma^{-1}\Lambda^\top\Lambda}_{k\ \times\ k})^{-1}\Lambda^\top, \tag{A.9}$$

which leads to a reduction of the computational cost, because only the inverse of a $k \ \times\ k$ ($k \leq 128$) matrix is computed instead of a matrix of size $128\ \times\ 128$.

What value of k should we choose in Eq. (A.7)? The total variance of the data, the trace of S, is equal to the sum of the eigenvalues, $\mathrm{tr}[S] = \sum_k \lambda_k$. In the experiments we set k at the smallest value such that

$$\sum_k \lambda_k \geq .95\ \mathrm{tr}[S]. \tag{A.10}$$

In words: we require the subspace to explain at least 95% of the data variance.

A.4 Computing an eigen-decomposition from little data

Eq. (A.7) requires the eigen-decomposition of the data covariance matrix S. Suppose A (size $128 \times n$) contains the zero mean descriptors for some part. Then $S \propto AA^\top$. The eigenvalue decomposition $SU = U\Lambda$, may now equally well be calculated using the covariance matrix $S' = A^\top A$ of the transposed data. Let U' and Λ' denote the eigenvectors and eigenvalues of S', respectively. Now

$$S'U' = A^\top AU' = U'\Lambda' \quad \text{left mult. with } A \ \Rightarrow \underbrace{AA^\top}_{S}\underbrace{AU'}_{U} = \underbrace{AU'}_{U}\underbrace{\Lambda'}_{\Lambda}, \tag{A.11}$$

and so we may use $U = AU'$. This leads to a reduction in the computational cost if $|A_p| < 128$.

A.5 Computing the posterior probabilities with high dimensional data

An issue when computing the posterior probability

$$p(\mathcal{P} = p\,|\mathbf{a}_f) = \frac{p(\mathbf{a}_f,\, \mathcal{P} = p)}{\sum_p p(\mathbf{a}_f,\, \mathcal{P} = p)}, \tag{A.12}$$

in Section 3.5.1 for high dimensional data-vectors ($\mathbf{a}_f \in \mathcal{R}^{128}$), is that the numerator $p(\mathbf{a}_f, \mathcal{P} = p)$ will be very small due to the high dimensionality of the data. It is not uncommon that the magnitude of these probabilities will drop under machine precision. This results in an error or zero-valued probabilities. If necessary, this issue can be circumvented by evaluating the numerator for each part according to:

$$p(\mathbf{a}_f, \mathcal{P} = p) \propto \exp\left[\log p(\mathbf{a}_f,\, \mathcal{P} = p) - \underset{p}{\mathrm{argmax}}[\,\log p(\mathbf{a}_f,\, \mathcal{P} = p)\,]\right], \tag{A.13}$$

which is equal $p(\mathbf{a}_f, \mathcal{P} = p)$ up to scalar multiplication defined by the argmax term. The worst thing that can happen now, is that $p(\mathbf{a}_f, \mathcal{P} = p)$ will be 1 for one part and 0 for the others.

Appendix B

Sift matching results

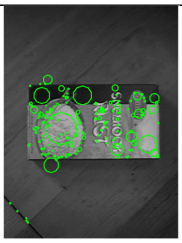
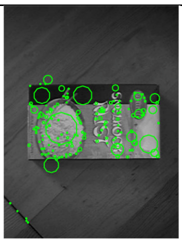
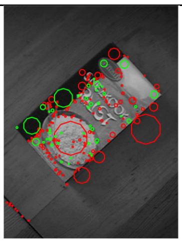
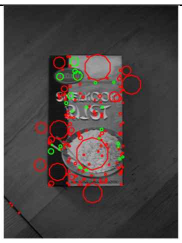
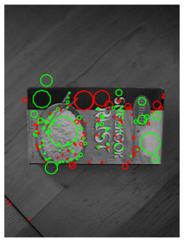
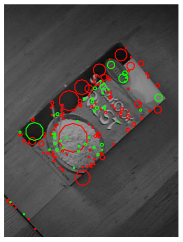
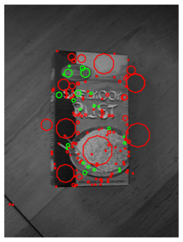
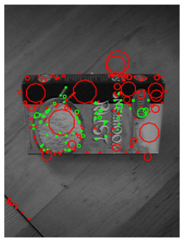
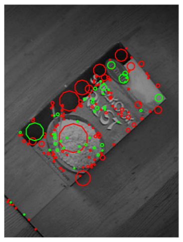
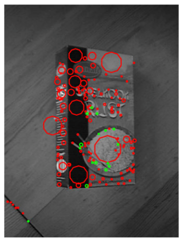
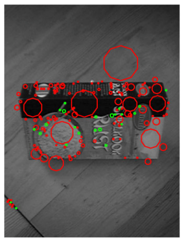
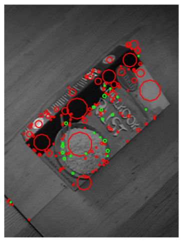
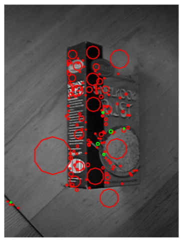
model	$\alpha = 0°$	$\alpha = 45°$	$\alpha = 90°$	
				$\gamma = 0°$
				$\gamma = 15°$
				$\gamma = 30°$
				$\gamma = 45°$

Table B.1: Sift feature responses for a planar object. Features that match to a feature in the model are in green.

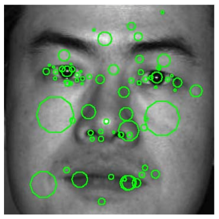
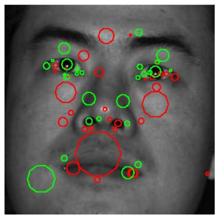
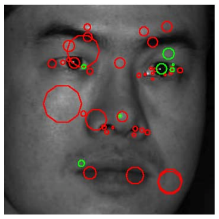
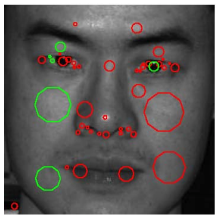
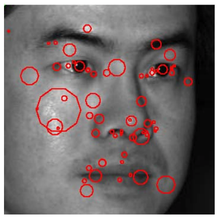
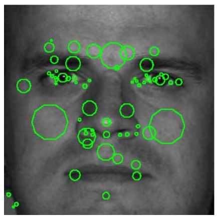
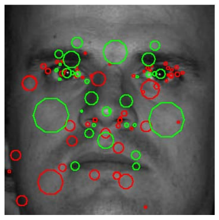
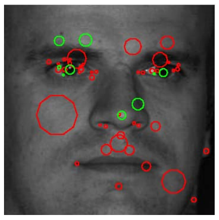
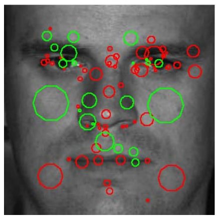
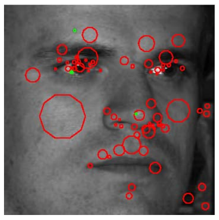
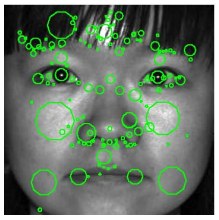
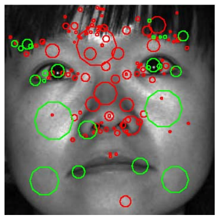
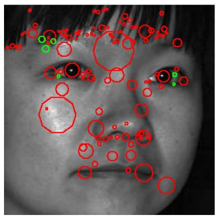
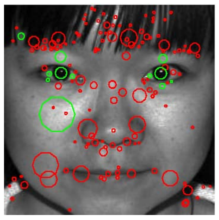
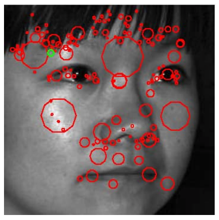
pose 0	pose 1	pose 2	pose 6	pose 8
				
				
				

Table B.2: Sift feature responses for a faces in different poses from the Yale data set. Features that match to a feature in the model are in green.

Appendix C

Estimating the affine pose

As explained in Section 4.2, to obtain the pose for a given hypothesis, we compute

$$\begin{aligned} L_m &= \underset{L}{\operatorname{argmax}} \prod_{p\colon h(p)>0} G(\mathbf{x}_{h(p)};\, L\tilde{\boldsymbol{\mu}}_p, \Sigma_p) \\ &= \underset{L}{\operatorname{argmax}} \sum_{p\colon h(p)>0} \log G(\mathbf{x}_{h(p)};\, L\tilde{\boldsymbol{\mu}}_p, \Sigma_p) \\ &= \underset{L}{\operatorname{argmax}} \; -\tfrac{1}{2} \sum_{p\colon h(p)>0} (\mathbf{x}_p - L\tilde{\boldsymbol{\mu}}_p)^\top \Sigma_p^{-1} (\mathbf{x}_p - L\tilde{\boldsymbol{\mu}}_p). \end{aligned} \tag{C.1}$$

Maximizing a negative quantity is the same as minimizing the positive form, which we denote here with Z:

$$Z = \frac{1}{2} \sum_p (\mathbf{x}_p - L\tilde{\boldsymbol{\mu}}_p)^\top \Sigma_p^{-1} (\mathbf{x}_p - L\tilde{\boldsymbol{\mu}}_p), \tag{C.2}$$

where p is shorthand for p with $h(p) > 0$. In order to find the maximum argument, we differentiate Z with respect to L and equate the result to zero. The derivative of Z with respect to L is given by

$$\frac{\partial Z}{\partial L} = \sum_p \left[C^p L D^p - B^p \right], \tag{C.3}$$

$$\text{where} \quad C^p = \Sigma_p^{-1}, \quad D^p = \tilde{\boldsymbol{\mu}}_p \tilde{\boldsymbol{\mu}}_p^\top, \quad \text{and} \quad B^p = \Sigma_p^{-1} \mathbf{x}_p \tilde{\boldsymbol{\mu}}_p^\top. \tag{C.4}$$

Equating to zero results in

$$\sum_p C^p L D^p = B \qquad \text{with } B = \sum_p B^p, \tag{C.5}$$

which we have to solve for L. Eq. (C.5) can be rewritten as

$$\forall_{ij} \quad B_{ij} = \sum_{k,l} L_{kl} \sum_p C^p_{ik} D^p_{lj}, \tag{C.6}$$

which can again be rewritten as:

$$\underbrace{\begin{bmatrix} e_{11}^{11} & \dots & e_{13}^{11} & \dots & e_{23}^{11} \\ \vdots & & \vdots & & \vdots \\ e_{11}^{13} & \dots & e_{13}^{13} & \dots & e_{23}^{13} \\ \vdots & & \vdots & & \vdots \\ e_{11}^{23} & \dots & e_{13}^{23} & \dots & e_{23}^{23} \end{bmatrix}}_{E} \underbrace{\begin{bmatrix} a_{11} \\ a_{12} \\ t_x \\ a_{21} \\ a_{22} \\ t_y \end{bmatrix}}_{\mathbf{l}} = \underbrace{\begin{bmatrix} B_{11} \\ B_{12} \\ B_{13} \\ B_{21} \\ B_{22} \\ B_{23} \end{bmatrix}}_{\mathbf{b}} \quad \text{where } e_{kl}^{ij} = \sum_p C_{ik}^p D_{lj}^p \tag{C.7}$$

There no reason (in general) to assume that E will be rank-deficient, so the solution to this set of equations is $\mathbf{l} = E^{-1}\mathbf{b}$. In case E is singular we may use the pseudo-inverse, however in our experiments this was never necessary.

Bibliography

[1] M. Burl and P. Perona. Recognition of planar object classes. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 223–230, 1996.

[2] J.W. Demmel. *Numerical linear algebra*. SIAM, first edition, 1997.

[3] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, volume 2, page 264, June 2003.

[4] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

[5] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of The Fourth Alvey Vision Conference, Manchester*, pages 147–151, 1988.

[6] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[7] S. Helmer. Object recognition with many local features. Master's thesis, University of British Columbia (Department of Computer Science), October 2004.

[8] Q. Ji and R. Hu. 3d face pose estimation and tracking from a monocular camera. *Proceedings of the International Conference on Pattern Recognition*, pages 400–403, 2004.

[9] T. Kadir and M. Brady. Scale, saliency and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

[10] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[11] D.G. Lowe. Demo software: Sift keypoint detector.

[12] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1150–1157, 1999.

[13] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[14] T.V. Pham. *Learning Spatial Relations for Object Recognition*. PhD thesis, University of Amsterdam, 2005.

[15] W. K. Pratt. *Digital image processing*. Wiley, 1991.

[16] Philips Research Press Release. Philips presents intuitive icat concept for the home, 2004.

[17] H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[18] S. Russell and P. Norvig. *Artificial Intelligence: a modern approach*. Prentice Hall, first edition, 1995.

[19] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[20] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3:71–86, 1991.

[21] P. Viola and M. Jones. Robust real-time object detection. *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2002.

[22] P. Viola and M. Jones. Fast multi-view face detection. *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2003.

[23] M. Weber. *Unsupervised Learning of Models for Object Recognition*. PhD thesis, California Institute of Technology, 2000.

[24] M. Weber, W. Einhuser, M. Welling, and P. Perona. Viewpoint-invariant learning and detection of human heads. In *FG '00: Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, page 20. IEEE Computer Society, 2000.

[25] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.