Fitting a 3-D Deformable Model To 2-D Images - Some Results

Karl Skoglund

Informatics and Mathematical Modelling, Technical University of Denmark

October 30, 2003

Abstract

This document renders some results from trying to fit a 3-D appearance model [3] of the human face to 2-D images using ordinary optimization methods. The objective function is the absolute difference between a 2-D projection of the model and the image. The results show well-formed 2-parameter surfaces near global minima, which should lead to robust minimization in larger parameter spaces.

1 Introduction

A deformable model in three dimensions can be used for (semi) automatic recognition and segmentation of 2-D images. The advantage of using a 3-D model is the larger amount of information it contains. A model of the human face [4], which is used in this article, can be fitted to images of heads in a wide range of poses.

In two dimensions, deformable models can be successfully fitted to objects in underlying images using the *active appearance model* [3]. In this article, a different approach is taken. Inspired by Blanz et al. [1], ordinary optimization algorithms are used.

2 Method

The *appearance* of the deformable face model is controlled by a number of parameters, $n-1$, where n is the number of actual faces the model is built from. These control shape and texture simultaneously. To specify position, orientation and scale, seven $(3+3+1)$ more parameters are added. In an extended implementation, parameters for lighting conditions, camera settings, etc. could be added.

The fitting algorithm attempts to find a set of parameters that results in the lowest possible difference between a 2-D projection of the model and the image. The difference is calculated by treating the projection and image as vectors, $\mathbf{v}_p$ and $\mathbf{v}_i$, and calculating the norm of the difference between them. The p-norm $\|\mathbf{v}\|_p$ is defined as

$$\|\mathbf{v}\|_p \equiv \left(\sum_i |v|^p \right)^{1/p} \tag{1}$$

The norm used here is the so-called L_1-norm, $\|\mathbf{v}_p - \mathbf{v}_i\|_1$ which is simply the sum of absolute differences. The reason for this is that the L_1-norm doesn't penalize differences as heavily as the commonly used L_2-norm.

To find the most suitable ensemble of parameters, an iterative, multidimensional optimization method is used. In this article two different methods are employed, *steepest decent*, which is a slowly converging gradient-based method, and the *simplex* method (also known as the *Nelder-Mead* or *Amoeba* method), which is not gradient-based and has faster convergence. The steepest decent algorithm was used to investigate whether a gradient-based method would have any advantages over the widely used simplex method.

The notation and descriptive ideas in the next two sections are from [2].

2.1 The Steepest-Decent Method

The steepest decent method works by finding the direction of maximum slope and then moving in this direction until the minimum along this line is

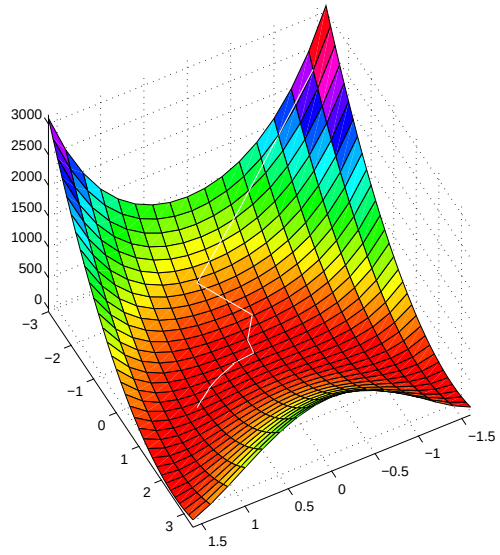


Figure 1: Using steepest decent to find the minimum of the "banana-function".

found. This procedure is iterated until the real minimum is reached. The direction $\mathbf{d}_k$ of steepest decent is found as the negative gradient vector. For non-analytical functions this can be calculated using finite differences. One iteration is as follows

- Find the direction of steepest decent at the current position k, $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$.
- Let $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{d}_k$, where λ_k solves $\min_\lambda g(\lambda) = \min_\lambda f(\mathbf{x}_k + \lambda \mathbf{d}_k)$.

Often it is not possible or necessary to solve the one-dimensional line search in every iteration. Instead, a single step length can be approximated using for example *Armijo's rule*. This rule defines a function $T(\lambda) = g(0) + \epsilon\lambda g'(0)$, $0 < \epsilon < 1$. The step length λ is then chosen so that

$$\begin{aligned} g(\lambda) &\leq T(\lambda) \\ g(\alpha\lambda) &\geq T(\alpha\lambda), \quad \alpha > 1 \end{aligned} \tag{2}$$

Common choices for the parameters ϵ and α are $\epsilon = 0.5$ and $\alpha = 2$.

2.2 The Nelder-Mead (Simplex) Method

The method used for most experiments here is the Nelder-Mead, or Simplex, method. A simplex in $\mathbf{R}^n$ is a set of $n+1$ corner points $\mathbf{x}_1, \ldots, \mathbf{x}_{n+1}$ such that the vectors $\mathbf{x}_i - \mathbf{x}_1$ are linearly independent. In practice this forms a non-degenerate hyper-tetrahedron. In one dimension this is a line, in two dimensions a triangle and in three dimensions a tetrahedron. The method works by changing the position of the corner point with the largest function value in each iteration. This makes the simplex move and deform until it encompasses the minimum with some chosen accuracy.

Define $\mathbf{x}_m$ and $\mathbf{x}_M$ as the corner points with the smallest and largest function values respectively. In an iteration, $\mathbf{x}_M$ is moved along the line going from $\mathbf{x}_M$ through the center of mass of all other points. All but one point in a simplex form a hyper-plane, so the center of mass is calculated as

$$\mathbf{x}_g = \frac{1}{n} \sum_{\mathbf{x}_i \neq \mathbf{x}_M} \mathbf{x}_i \tag{3}$$

The new corner point is chosen along the line according to

$$\mathbf{x}_r = \mathbf{x}_g + \alpha(\mathbf{x}_g - \mathbf{x}_M), \quad \alpha > 0 \tag{4}$$

Three cases occur:

1. $f(\mathbf{x}_m) < f(\mathbf{x}_r) < f(\mathbf{x}_M)$, choose $\mathbf{x}_{\text{new}} = \mathbf{x}_r$.
2. $f(\mathbf{x}_r) \leq f(\mathbf{x}_m)$, the minimum is probably even further away, so the simplex is *expanded*. Let

   $$\mathbf{x}_e = \mathbf{x}_g + \beta(\mathbf{x}_r - \mathbf{x}_g), \quad \beta > 1 \tag{5}$$

   Tow sub-cases occur:

   (a) $f(\mathbf{x}_e) < f(\mathbf{x}_r)$, set $\mathbf{x}_{\text{new}} = \mathbf{x}_e$.

   (b) $f(\mathbf{x}_e) \geq f(\mathbf{x}_r)$, set $\mathbf{x}_{\text{new}} = \mathbf{x}_r$.
3. $f(\mathbf{x}_r) \geq f(\mathbf{x}_M)$, the minimum seems to be inside the area defined by the current simplex and $\mathbf{x}_r$, so the simplex is *contracted*. Let

   $$\mathbf{x}_c = \mathbf{x}_g + \gamma(\mathbf{x}_r - \mathbf{x}_g), \quad 0 < \gamma < 1 \tag{6}$$

   Again, two sub-cases occur:

   (a) $f(\mathbf{x}_c) < f(\mathbf{x}_M)$, set $\mathbf{x}_{\text{new}} = \mathbf{x}_c$.

   (b) $f(\mathbf{x}_c) \geq f(\mathbf{x}_M)$, the simplex is *shrinked*, set $\mathbf{x}_i = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_m)$ for all i.

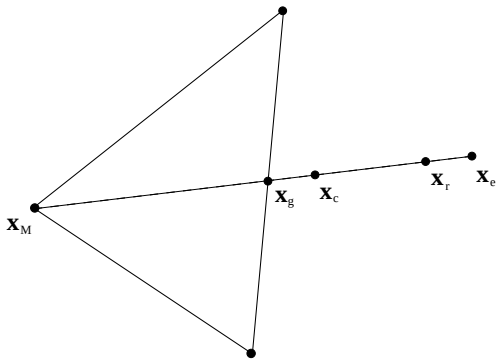


Figure 2: A two-dimensional simplex annotated with the possible new positions for $\mathbf{x}_M$.

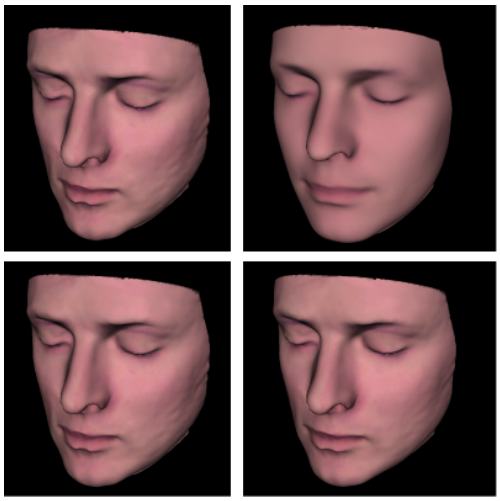


Figure 3: Fitting to an image using all 30 parameters. The left column shows the image and the right shows the model. The top row represents before and the bottom row after optimization.

3 Results

In appendix A, a series of plots are presented, each showing one or two parameters and the corresponding function values. Surfaces constructed from appearance parameters are smooth and unimodal in the whole range (± 3 standard deviations). The plots showing position and rotation changes also have these properties, but show slightly more complex surfaces.

Figure 3 shows a face fitted to an image using all 30 parameters. The model was manually initialized with respect to location, orientation and scale, but no initialization of the appearance parameters was carried out. Despite this, the algorithm was able to correctly fit the model to the image. Of course, the image used in this example poses a relatively easy problem. It is an image of one of the faces from which the model is built. In a real application, one would like to fit the model to an ordinary digital image of a person, including background. This, however, requires a more general model, and some extra work.

4 Conclusions and Summary

This paper has presented some results from fitting a deformable 3-D model to images. Plotting pairs of parameters against the absolute difference between the model and the image shows well-formed surfaces around the minima. This indicates that it should be possible to fit the model to an image using more, possibly all, parameters. A few successful experiments with this have been carried out (see figure 3). However, to make the method useful in a real application, the model needs to be more general, more parameters must be added along with greater control of the initialization of these, and a way to deal with background information must be devised.

References

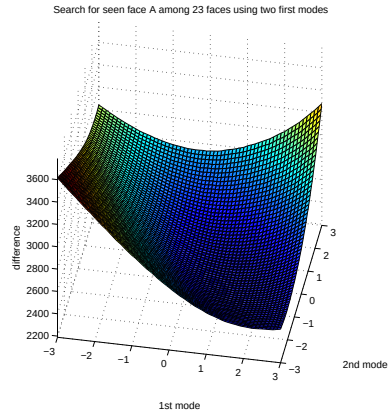
[1] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 187–194, Los Angeles, 1999. Addison Wesley Longman.

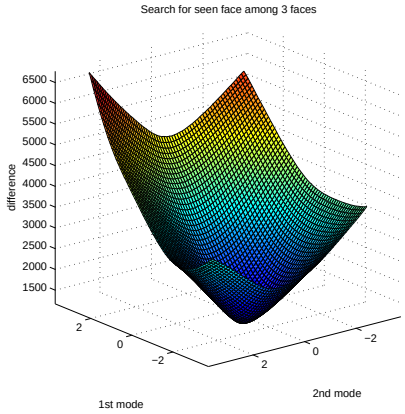
[2] Lars-Christer Boijers. *Lectures On Optimization.* KFS AB, 2001.

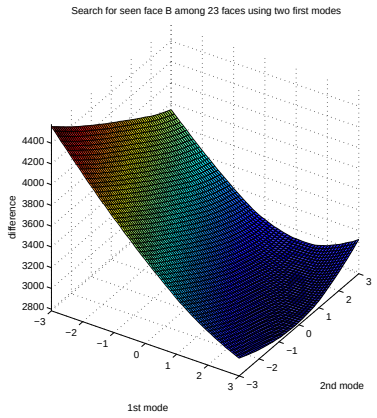
[3] T.F Cootes and C.J. Taylor. *Statistical Models of Appearance for Computer Vision.* University of Manchester, 2001.

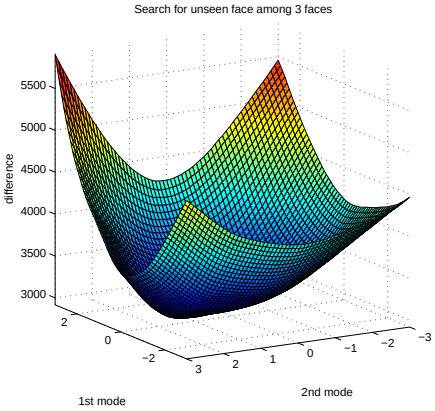
[4] K. Skoglund. Three-dimensional face modelling and analysis. Master's thesis, IMM, Technical University of Denmark, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, August 2003.

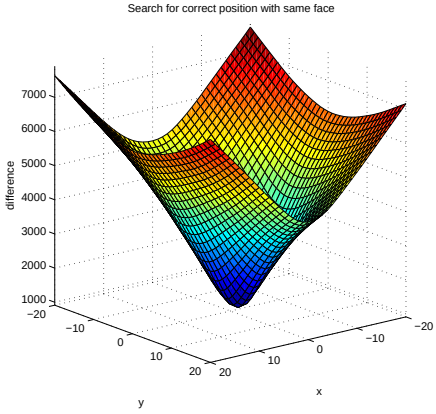
A Objective Function Plots

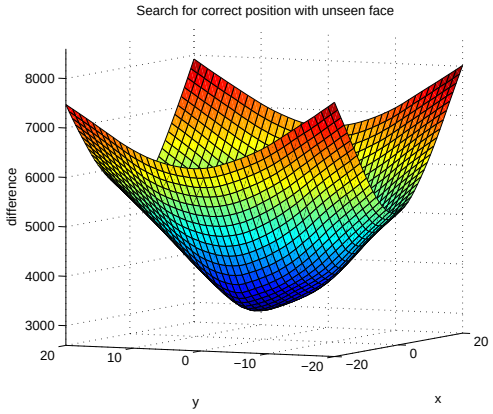


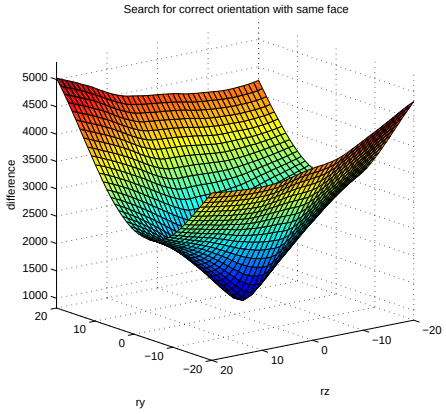


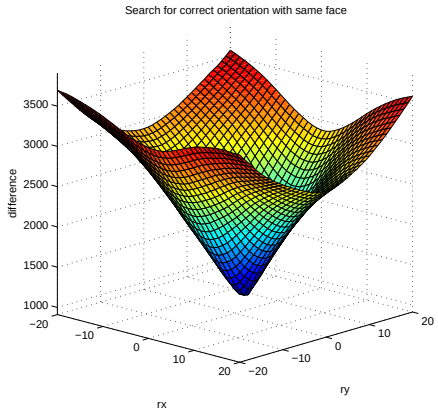


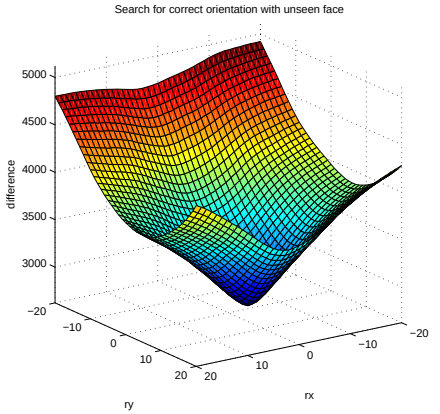


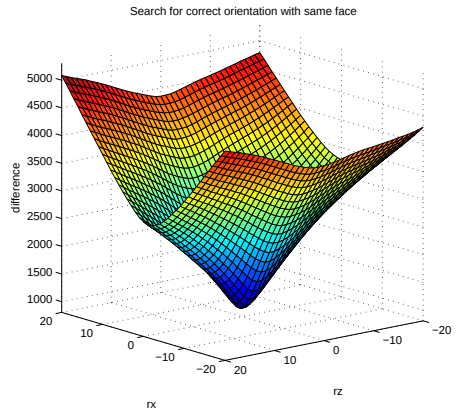


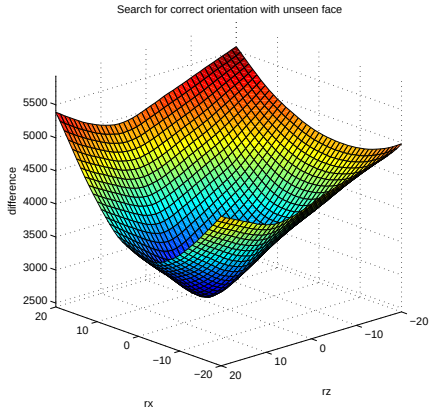

Search for correct position with unseen face
difference
8000
7000
6000
5000
4000
3000
20
10
0
−10
−20
y
−20
0
20
x

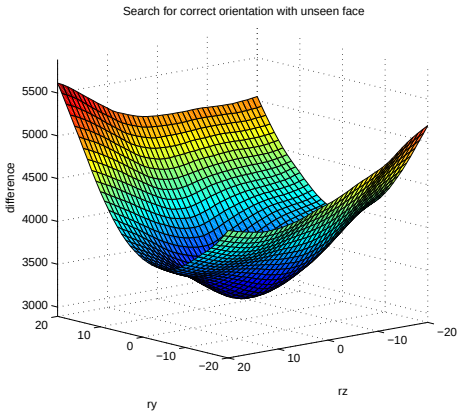

Search for correct orientation with same face
difference
5000
4500
4000
3500
3000
2500
2000
1500
1000
20
10
0
−10
−20
ry
20
10
0
−10
−20
rz

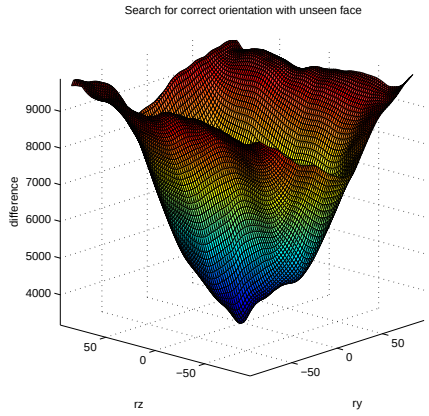

Search for correct orientation with same face
difference
3500
3000
2500
2000
1500
1000
−20
−10
0
10
20
rx
−20
−10
0
10
20
ry

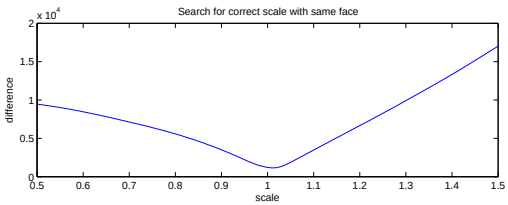

Search for correct orientation with unseen face
difference
5000
4500
4000
3500
3000
−20
−10
0
10
20
ry
20
10
0
−10
−20
rx

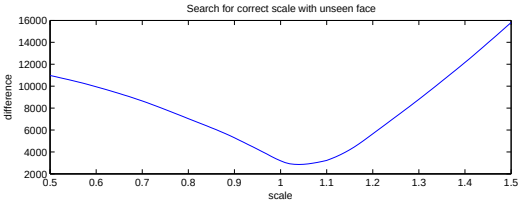

Search for correct orientation with same face
difference
5000
4500
4000
3500
3000
2500
2000
1500
1000
20
10
0
−10
−20
rx
20
10
0
−10
−20
rz


Search for correct orientation with unseen face
difference
5500
5000
4500
4000
3500
3000
2500
20
10
0
−10
−20
rx
20
10
0
−10
−20
rz


Search for correct orientation with unseen face
difference
5500
5000
4500
4000
3500
3000
20
10
0
−10
−20
ry
20
10
0
−10
−20
rz


Search for correct orientation with unseen face
difference
9000
8000
7000
6000
5000
4000
50
0
−50
rz
50
0
−50
ry


Search for correct scale with same face
x 10^4
difference
2
1.5
1
0.5
0
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5
scale


Search for correct scale with unseen face
difference
16000
14000
12000
10000
8000
6000
4000
2000
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5
scale