

# Using line features for 3D face registration

Fabian Brix

18.06.2013

## **Abstract**

In this bachelor thesis we present a 3D face registration pipeline by adapting Gaussian Process Regression and using labelled point and line correspondences as prior information for the registration process. Gaussian Processes are ideal for modelling smooth deformations of an arbitrary shape surface. Gaussian Process Regression can then be implemented to sample deformation fields that map a template on to a target shape using a set of corresponding facial landmarks. In our approach we extend the registration of face scans by incorporating additional information from line features. The line features define correspondence in highly characteristic areas of a pair of faces, i.e. the eyes and the ears. The deformation fields are optimized - using the prior information as constraints - to obtain a plausible mapping of the template on to the target. We demonstrate that line features have a profound impact on the accuracy and quality of the registration results. In order to enhance the expressiveness of the obtained registrations, we plan to present a near-optimal optimization process for Gaussian Process Regression applied to 3D faces in the near future.

# Contents

<b>Contents</b>	<b>3</b>
<b>1</b>	<b>5</b>
1.1 Prior Work . . . . .	5
1.2 Overview . . . . .	6
<b>2</b>	<b>7</b>
2.1 3D Morphable Model . . . . .	7
2.2 Achieving Correspondence through Registration . . . . .	8
2.3 Available Data . . . . .	9
<b>3</b>	<b>11</b>
3.1 Stochastic Processes . . . . .	11
3.2 Gaussian Processes . . . . .	11
3.3 Gaussian Process Regression . . . . .	13
3.4 Application to 3D Face Meshs . . . . .	15
3.5 Fitting & Optimization . . . . .	17
<b>4</b>	<b>20</b>
4.1 Line Features . . . . .	20
4.2 Sampling 3D Points from 2D Line Features . . . . .	21
4.3 Rigid Mesh Alignment . . . . .	26
4.4 Prior Model . . . . .	28
4.5 Posterior Model . . . . .	29
4.6 Fitting . . . . .	29
4.7 Robust Loss Functions . . . . .	30
<b>5</b>	<b>32</b>
5.1 Sampling . . . . .	32

5.2	The effect of Line Features . . . . .	32
5.3	Registration quality . . . . .	33

# 1

## Introduction

3D Face Models have a wide variety of applications in face image analysis. They are used in face recognition from 2D images, 3D face reconstruction from 2D images and the extraction of face attributes such as facial expressions. Given a set of scanned faces, registering these scans is the first step in the process of deriving a 3D model of morphable faces. Registration is necessary to establish the required dense correspondence between the input scans. It is the process deforming a template so that it reproduces a face scan. In order to achieve this a set of known correspondences, for example point correspondences, are used to define a transformation. In the end, replacing the scans with the respectively deformed template ensures that all faces have the same topology. In effect, the respective parts of different faces can be combined to form new faces. Defining a reproducible and accurate registration algorithm is therefore essential to obtaining a high-quality 3D model. In this thesis we present a registration algorithm which tackles the problem of 3D face registration by modeling the transformation using Gaussian Process Regression (GPR) and extending the prior information used in the transformation to parametric curves - referred to as line features - which define key regions of the face.

### 1.1 Prior Work

Gaussian Processes for Machine Learning were comprehensively introduced in [1] as an approach for defining a prior distribution over a space of functions. A Gaussian Process (GP) is a stochastic process, which in turn can be seen as the generalization of probability distributions to functions. Computations prove to be straightforward because a GP is modelled by a multivariate normal distribution.[2] (Using Landmarks as a Deformation Prior for Hybrid Image Registration) GPs are used

to describe transformation priors in Image Registration. In this context GP allows for defining a space of possible transformations of an image and further allows for incorporating point correspondences as additional information for inference of admissible deformations. The distinction in regard to other registration methods lies therein that the corresponding points are not hard-wired in the transformation, but instead modify the prior to include only the more feasible transformations. In [3] the process of formulating the fitting and optimization of the admissible transformations is described for image registration. GPR has further been used to infer shapes in statistical shape models [4]. In short, the reason for applying GPR is its reproducability in modelling a deformation prior for arbitrary shapes.

The registration methods used so far [5] in 3D face registration are specifically designed for this context are therefore not very well reproducible. Furthermore, these registration methods have to be directly constrained on point correspondences in the optimization process, whereas with GPR only admissible deformations have to be optimized. The advantage of our approach of applying GPR is that we have derived a general framework for 3D face registration.

## 1.2 Overview

In face registration a common parametrization has to be established for all face scans. Using a GP in 3D face registration models a space of possible prior deformations of a facial scan for the purpose of registering it with another scan. The given a priori point correspondences - called landmarks - are used as additional information constraining the space of possible deformations. We additionally use line features to match semantically prominent regions - which have to be brought into dense correspondence - for all scans. With the help of line features we want to infer deformations that result in a more accurate mapping of these regions and therefore better registration results. Using GPR in the registration process results in a highly customizable algorithm, that is dependent on the covariance of the surface points on the a priori given information. The following chapters cover the basics of building 3D morphable models, GPR theory and the implementation of a face registration pipeline with the above-mentioned algorithm. The registration results shown in the last chapter give a first expression of what this method is capable of and how the incorporation of line features affects the registration outcome. The question of how to incorporate the line features in the registration process will be answered in the course of this thesis.

# 2

## 3D Model Building

This chapter describes how to build a generative textured 3D face model from an example set of 3D face scans. A morphable model is derived from the set of scans by transforming their shape and texture into a vector space representation. The term generative implies that - as the face space is spanned by the set of examples - new faces can be generated by calculating linear combinations of these examples.

### 2.1 3D Morphable Model

The 3D Morphable Model (3DMM) published by Blanz and Vetter in 1999 (bib) is a multidimensional function for modelling textured faces derived from a large set of 3D face scans. A vector space of face shapes can be constructed from the available data set where each face is represented by a shape-vector  $S \in \mathbb{R}^{3n}$  that consists of a component-wise listing of its  $n$  vertices. The texture-vector then  $T \in \mathbb{T}^{3n}$  contains the corresponding RGB values in similar fashion. New shapes and textures can now be constructed with a linear combination of the set of  $m$  available faces governed by shape and texture coefficients  $\vec{\alpha}, \vec{\beta} \in \mathbb{R}^m$ .

$$\mathcal{S}_{new} = \sum_{i=1}^m \alpha_i S_i, \quad \mathcal{T}_{new} = \sum_{i=1}^m \beta_i T_i \quad (2.1)$$

However, the goal of such a 3D face model is not just to construct arbitrary faces with linear combinations of the set of examples, but to derive plausible faces from them. This is achieved by estimating two multivariate normal distributions for the coefficients in  $\vec{\alpha}$  and  $\vec{\beta}$ . These multivariate normal distributions are constructed from the average face shapes  $\bar{S} \in \mathbb{R}^{3N}$  and textures  $\bar{T} \in \mathbb{R}^{3N}$  of the datasets and the covariance matrices  $K_S$  and  $K_T$ , which are defined over the residuals of each

example and with both the shape  $R_S = S - \bar{S}$  and texture  $R_T = T - \bar{T}$  averages. The covariance matrices are then used to perform Principal Component Analysis (PCA) which defines a basis transformation to an orthogonal coordinate systems. The axes are defined by the eigenvectors of the respective covariance matrix.

$$\mathcal{S}(\vec{\alpha}) = \bar{S} + S\vec{\alpha}, \quad \mathcal{T}(\vec{\beta}) = \bar{T} + T\vec{\beta} \quad (2.2)$$

In (2.2) the  $N = m$  principal eigenvectors of  $K_S$  and  $K_T$  respectively are assembled column-wise in  $S$  and  $T$  and scaled in a way such that the prior distribution over the face shape and texture parameters is given by a multivariate normal distribution with unit covariance (Amberg).

$$p(\vec{\alpha}, \vec{\beta}) = \mathcal{N}(\vec{\alpha} || \mathbf{0}, \mathbb{I}) \mathcal{N}(\vec{\beta} || \mathbf{0}, \mathbb{I}) \quad (2.3)$$

Calculating the likelihood for a set of coefficients is straightforward and therefore it is now possible to find out how likely a face is given the model.

## 2.2 Achieving Correspondence through Registration

In order for a 3D Morphable Model to generate plausible faces we have to make sure that all faces share the same representation, so that only the same parts of the respective faces are combined with one another. For example a nose is combined with another nose and not with a mouth. For this reason, the meshes first have to be brought into correspondence. In fact, we need dense correspondence between all mesh points. This is the case when vertices at the same semantical position in different meshes have the same vertex number, i.e the left corner of the left eye. The process of calculating a dense point-to-point correspondence between two meshes is called registration. The training data used for learning a 3D Morphable Models consists solely of registered examples of the 3D shape and texture of faces.

**Registration Algorithms** When performing registration, we start with a template shape that is a prototype of the semantic parametrization. In order to compute the parametrization for another shape we deform the template to match this shape. The shape being matched is referred to as the target. Both are considered as matched if the semantic face regions are mapped onto each other as closely as possible by satisfying a specified similarity measure. If dense correspondence is achieved, the

target shape can be replaced with the deformed template, which will further be used in the process of building a model.

Today's registration algorithms use prior information in the form of manually clicked feature points at prominent points of the face, so called landmarks. These are important for the alignment of template and target as well as achieving an accurate match.

## 2.3 Available Data

For testing the registration algorithm we have a large set of face scans given - male as well as female - all of them recorded at the University of Basel by an active stereo vision system using structured light. The scanner records four 3D surfaces, called shells, and further records three high resolution color images for every face. The 3D shells are obtained through triangulation. Each of the four

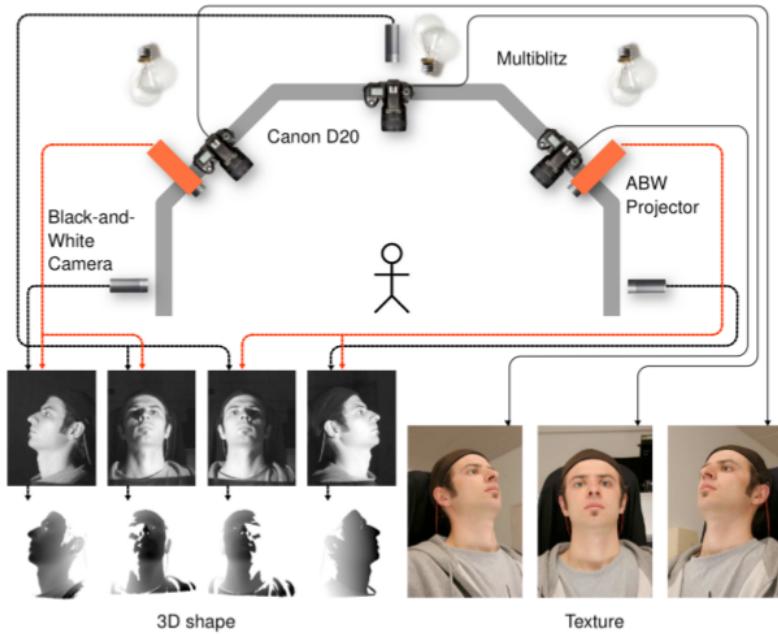


Figure 2.1: schema of the structured light 3D scanning system employed to record the facial scans. It consists of two projectors that project triangle gradient patterns on to a face from opposing angles, three black-and-white cameras used for the triangulation and three digital cameras which record the facial skin.

black-and-white cameras 2.1 records the triangle gradient patterns - stripe patterns - emitted from the two projectors. The patterns allow for the extraction of depth information by the four neighbouring projector-camera-pairs using triangulation.

The eyes and hair can not be captured due to their reflection properties. Therefore the shells only cover the front and the side of the head and have holes in the eyes and instead of parts of the ears. The face textures are obtained from the color images. To obtain a whole mesh the 3D shells first have to be cleaned and then merged.

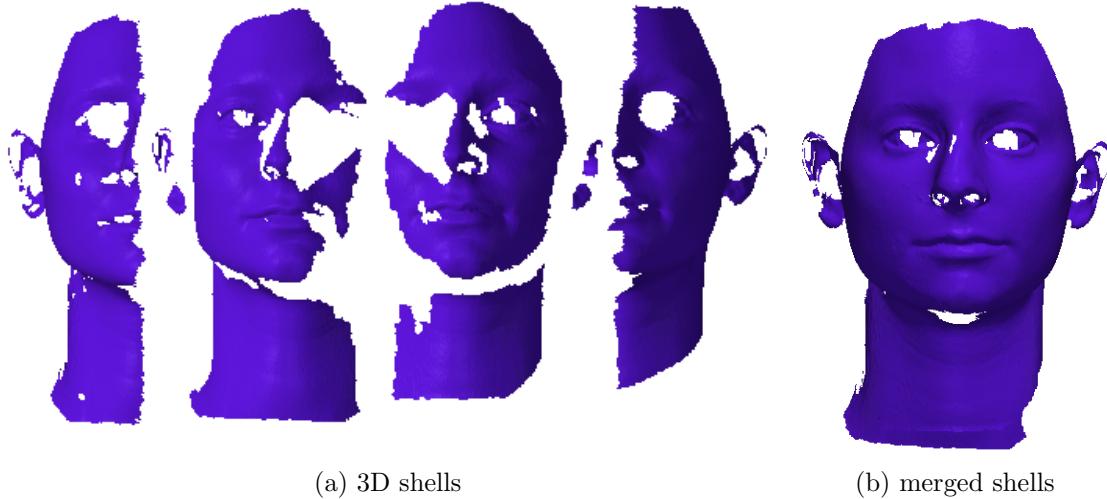
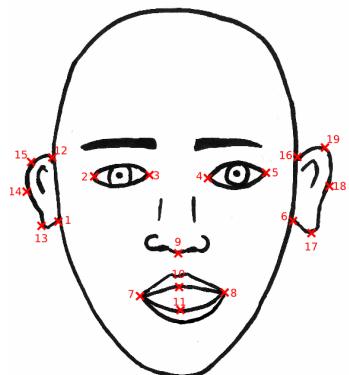


Figure 2.2: ...

The scans are further manually annotated with a set of so called landmarks 2.3. Furthermore, they are used to create 2D parametric curves that describe key regions of the face, i.e. the eyes. The line features will be fully introduced together with the registration pipeline. In the next chapter we will elaborate on the approach of using Gaussian Processes to solving the problem 3D face registration and then describe how to incorporate the annotations.



# 3

## Gaussian Processes in 3D Face Registration

The first of our two objectives is to build a face registration pipeline. In this context we use a stochastic process, more specifically a vector-valued Gaussian process or Gaussian random field as the registration algorithm. To begin with, we recapitulate the definition of stochastic processes and extend it to the definition of Gaussian processes. In the next step we introduce Gaussian Process Regression and finally explain it can be applied 3D face mesh registration.

### 3.1 Stochastic Processes

In probability theory a stochastic process consists of a collection of random variables  $\{X(t)\}_{t \in \Omega}$  where  $\Omega$  is an index set. It is used to model the change of a random value over time. The underlying parameter time is either real or integer valued. A generalization of a stochastic process, which can handle multidimensional vectors, is called a random field.

### 3.2 Gaussian Processes

A Gaussian process is a stochastic process in which each finite collection  $\Omega_0 \subset \Omega$  of random variables has a joint normal distribution. More formally, we define the collection of random variables  $\{X(t)\}_{t \in \Omega}$  to have a d-dimensional normal distribution if the collection  $\{X(t)\}_{t \in \Omega_0}$  - for any finite subset  $\Omega_0$  - has a joint  $d \times |\Omega_0|$ -dimensional normal distribution with mean  $\mu(\Omega_0)$  and covariance  $\Sigma(\Omega_0)$ . If  $\Omega \subseteq \mathbb{R}^n, n > 1$  holds, the process is a Gaussian random field. In the further proceedings the term “Vector-

valued Gaussian Processes” will be used to refer to Gaussian random fields. Defining the random variables on an index set in an n-dimensional space, allows for spatial correlation of the resulting values, which is an important aspect of the algorithm discussed later on.

An alternative way of viewing a Gaussian process is to consider it as a distribution over functions. This allows us to look for inference in the space of these functions given a dataset, specifically to find the deformation function given a 3D face mesh. Each random variable now yields the value of a function  $f(x)$  at a location  $x \in \mathcal{X}$  in the index set of possible inputs. We now denote the index set by  $\mathcal{X}$  to stress that we are ceasing to discuss Gaussian processes defined over time. In this function-space view a Gaussian Process at location  $x$  is thus  $f(x) \sim GP(\mu(x), k(x, x'))$  defined by its mean  $\mu : \mathcal{X} \rightarrow \mathbb{R}$  and covariance  $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  functions which in turn are defined over the set of input vectors. With  $\mu(\mathcal{X}) = (\mu(x))_{x \in \mathcal{X}}$  and  $\Sigma(\mathcal{X}) = (k(x, x'))_{x, x' \in \mathcal{X}}$  we obtain the full distribution of the process  $GP(\mu(\mathcal{X}), \Sigma(\mathcal{X}))$ . For the purpose of simplifying calculations we may assume that every random variable has zero mean without a loss of generality. When modeling a deformation field with a Gaussian process this circumstance implies that the expected deformation is itself zero.

**Covariance Functions** The key feature of a Gaussian Process is its covariance function also known as “kernel”. It specifies the covariance  $\mathbb{E}[f(x)f(x')]$  between pairs of random variables for two input vectors  $x$  and  $x'$ , allowing us to make assumptions about the input space by defining the spatial co-dependency of the modelled random variables. Note that when assuming zero mean we can completely define the process’ behaviour with the covariance function.

A simple example of a covariance function is the squared exponential covariance function, defined by  $cov(f(x), f(x')) = k(x, x') = \exp(-\frac{(x-x')^2}{2l^2})$ . (derivation Rasmussen et al. p.83) *still to be continued and refined...*

It is possible to obtain different prior models by using different covariance functions. In our case, we use a stationary ( $x-x$ , invariant to translation), isotropic exponential covariance function - Squared Exponential Covariance Function (p. 38)

**Gaussian Process Prior** The specification of the covariance function implies that a GP is a distribution over functions. To illustrate this one can draw samples from a prior distribution of functions evaluated at any number of points,  $X_*$ . The Gaussian Process Prior is solely defined by the covariance matrix made up of the covariances of the input points.

$$\Sigma(X_*) = \begin{bmatrix} k(x_{*1}, x_{*2}) & \cdots & cov(f(x_{*1}), f(x_{*n})) \\ \vdots & \ddots & \vdots \\ cov(f(x_{*n}), f(x_{*1})) & \cdots & cov(f(x_{*n}), f(x_{*n})) \end{bmatrix} \in \mathcal{M}^{|X_*| \times |X_*|} \quad (3.1)$$

A sample is a random Gaussian vector  $f_* \sim \mathcal{N}(0, \Sigma(X_*))$  containing a function value for every given input point. Plotting random samples above their input points is a nice way of illustrating that a GP is indeed a distribution over functions, see figure 3.1. The GP Prior forms the basis for inference in Gaussian Process Regression.

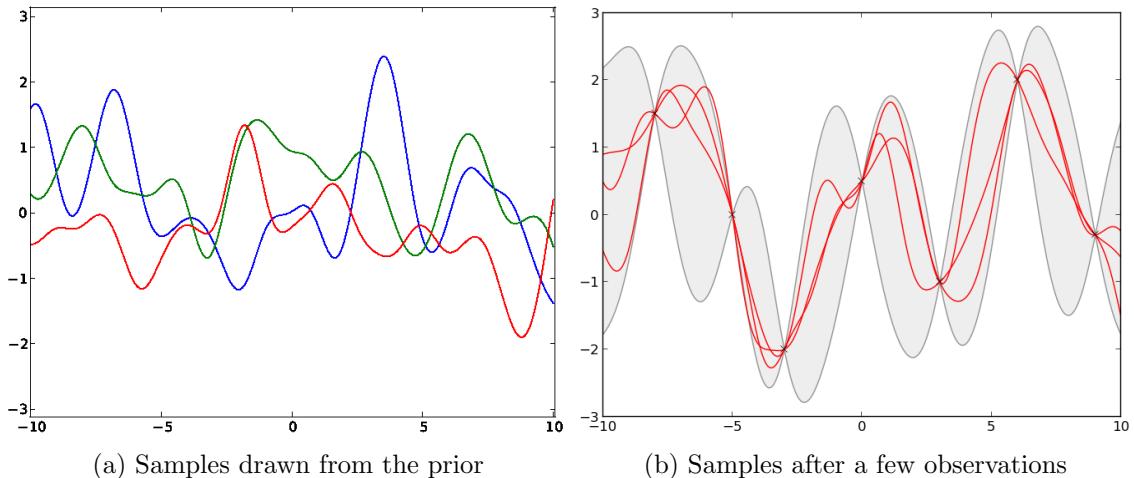


Figure 3.1: In figure a) three functions have been drawn from the GP prior, each has a sample size of 1000 points. Figure b) again shows three functions, but this time the prediction incorporates the information of random values observed at seven points in the input space

**Vector-valued Gaussian Processes** In order to use Gaussian processes to model deformation fields of three dimensional vectors as intended, there is the need for a generalization of the above definition from the function-space view. The random variables  $X_1, X_2, \dots, X_k, \dots, X_n$  are now d-dimensional vectors, yielding a covariance function of the form  $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{d \times d}$  and  $k(x, x') = \mathbb{E}[X_k(x)^T X_k(x')]$ . *Should this paragraph be continued?*

### 3.3 Gaussian Process Regression

The task of registering two 3D face meshes can be treated as a regression problem in which the goal is to predict the deformation of all floating mesh points, given

the displacement of the landmarks present in both meshes. Trying to fit an expected function - be it linear, quadratic, cubic or nonpolynomial - to the data is not a sufficiently elaborated approach to our problem. Using a Gaussian Process disposes of the need to describe the data by a specific function type, because the response for every input point is now represented by a normally distributed random value, in turn governed by the specification of the covariance function.

Key assumption: data can be represented as a sample from a multivariate gaussian distribution  $P$

**Regression Problem** Assume a training set  $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$  where  $x \in \mathbb{R}^d$  and  $y$  is a scalar output or target. The task is now to infer the conditional distribution of the targets for yet unseen inputs and given the training data  $p(\mathbf{f}_* | \mathbf{x}_*, \mathcal{D})$

**Noise-free Prediction** First we assume the observations from the training data to be noise-free so that we can fix the training data to these observations  $\mathbf{y}$  without complicating the model. The joint prior distribution with training  $\mathbf{f}$  and test  $\mathbf{f}_*$  outputs indicated is the following:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left( \mathbf{0}, \begin{bmatrix} \Sigma(X) & \Sigma(X, X_*) \\ \Sigma(X_*, X) & \Sigma(X_*) \end{bmatrix} \right) \quad (3.2)$$

We obtain the posterior samples illustrated in 3.1 b) by conditioning the above joint Gaussian prior distribution on the observations  $\mathbf{f}_* | \mathbf{f} = \mathbf{y}$  which results in the following distribution:

$$\mathbf{f}_* | X_*, (X, \mathbf{f}) \sim \mathcal{N} (\Sigma(X_*, X) \Sigma(X)^{-1} \mathbf{f}, \Sigma(X_*) - \Sigma(X_*, X) \Sigma(X)^{-1} \Sigma(X, X_*)) \quad (3.3)$$

**Prediction with Gaussian Noise Model** In most real world applications

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} \sim \mathcal{N} \left( \mathbf{0}, \begin{bmatrix} \Sigma(X) + \sigma^2 \mathcal{I}_{|X|} & \Sigma(X, X_*) \\ \Sigma(X_*, X) & \Sigma(X_*) \end{bmatrix} \right) \quad (3.4)$$

The distribution - now conditioned on the noisy observations - is thus

$$\mathbf{y}_* | \mathbf{f} = \mathbf{y} \sim \mathcal{N} (\bar{\mathbf{y}}_*, \Sigma(\mathbf{y}_*)) \quad (3.5a)$$

where the mean depends on the observed training targets

$$\bar{\mathbf{y}}_* = \Sigma(X_*, X) (\Sigma(X) + \sigma^2 \mathcal{I}_{|X|})^{-1} \mathbf{y} \quad (3.5b)$$

whilst the covariance depends only on the input points

$$\Sigma_* = \Sigma(X_*) - \Sigma(X_*, X) (\Sigma(X) + \sigma^2 \mathcal{I}_{|X|})^{-1} \Sigma(X, X_*) \quad (3.5c)$$

*Conclusion, how does this help us to proceed?*

## 3.4 Application to 3D Face Meshes

In this section we adapt the above presented theory to our case of 3D face mesh registration. The task at hand is to register a reference or template face mesh with a scanned face mesh. *registration and correspondence already explained in model building, deformation field bold instead of calligraphic?* We therefore strive to predict a deformation field  $\mathcal{D} : \mathcal{M} \subset \mathbb{R}^3 \rightarrow \mathbb{R}^3$  which assigns a displacement vector to every vertex in the template mesh. During registration we refer to the template as the moving mesh  $\mathcal{M}$ . Adding the displacement field to the moving mesh should then provide an accurate mapping to the target mesh  $\mathcal{T}$  and thereby perform the registration. Our objective is to register the template with multiple meshes of scanned faces. *Andreas: don't refer to 3DMM mean, because we haven't built a model yet! Leave out "triangulated", kind of mesh topology is not important in this thesis*

**Reference Mesh Prior** As defined by the deformation field the output the regression problem is in  $\mathbb{R}^3$  calling for the use of a Vector-valued Gaussian Process with random variables  $d \subseteq \mathbb{R}^3$  where  $d$  stands for deformation. After the template and target have been aligned ?? a Vector-valued Gaussian Process can be initialized by defining the prior over all vertices of the template mesh. For this purpose the covariance function has to be redefined to handle 3-dimensional vectors. *Prior consists of smooth deformations of the mean face*

$$k \left( \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} \right) = xy^T \in M^{3 \times 3} \quad (3.6)$$

Each covariance entails 9 relationships between the different components of the vectors, yielding a  $3 \times 3$  matrix. The covariance matrix then grows to become:

$$\Sigma_{\mathcal{X}} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \in M^{3n \times 3n} \quad (3.7)$$

The template mesh is defined by a set of vectors  $\mathcal{X} \in \mathbb{R}^3$  and a set of landmarks  $L_{\mathcal{M}} = \{l_1, \dots, l_n\} \subset \mathbb{R}^3$ . The mean vector  $\mu$  is made up of the component-wise listing of vectors so that it has dimensionality  $3n$ . Setting the whole mean vector to zero, as discussed before, implies a mean deformation of zero and makes perfect sense in this setting, because we are modelling deformations of the template surface. The prior distribution over the template mesh is therefore defined as

$$\mathcal{D} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathcal{X}}) \quad (3.8)$$

meaning that a sample deformation field can be directly drawn from the prior distribution of the template mesh.

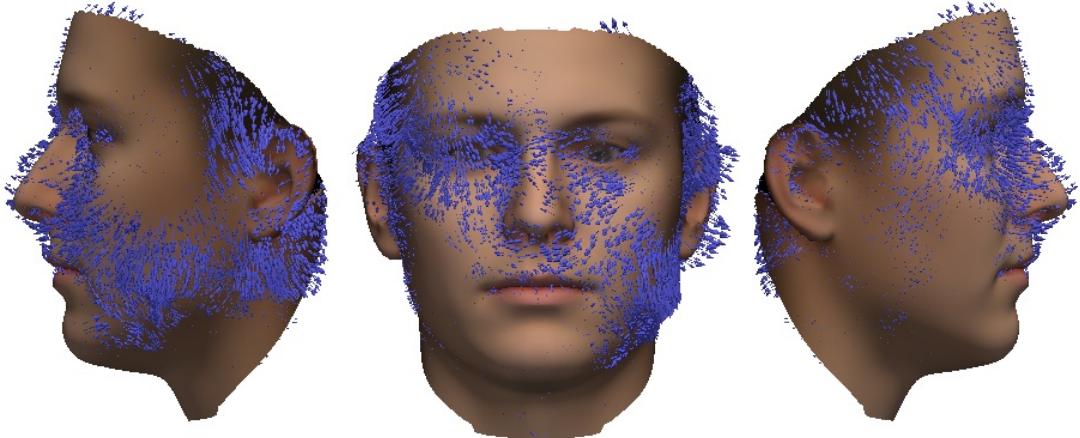


Figure 3.2: a sample **prior** deformation field - depicted with a set of **blue arrows** on the textured template face.

**Reference Mesh Posterior** The target landmarks also consist of a set  $L_{\mathcal{T}} = \{l_{\mathcal{T}1}, \dots, l_{\mathcal{T}N}\} \subset \mathbb{R}^3$ . Fixing the prior output to the deformation vectors  $\mathcal{Y} = \{\mathbf{t} - \mathbf{m} | \mathbf{t} \in L_{\mathcal{T}}, \mathbf{m} \in L_{\mathcal{M}}\}$  - defined by the distance between the template and target landmarks - and assuming additive i.i.d Gaussian noise, the resulting posterior distribution is

$$\begin{bmatrix} \mathcal{Y}_{\varepsilon} \\ \mathcal{D} \end{bmatrix} \sim \mathcal{N} \left( \mathbf{0}, \begin{bmatrix} \Sigma(\mathcal{Y}) + \sigma^2 \mathcal{I}_{3|\mathcal{Y}|} & \Sigma(\mathcal{Y}, X_*) \\ \Sigma(X_*, \mathcal{Y}) & \Sigma(X_*) \end{bmatrix} \right) \quad (3.9)$$

### *Is this a correct definition for the distribution?*

The deformation model is now rendered approximately fixed at the landmarks in the target mesh and the goal is to find valid deformations through the set of fixed targets, analogous to the case of eq. 3.5a. In other words the respective displacements at the template landmarks co-define the distribution of possible deformations of all vertices

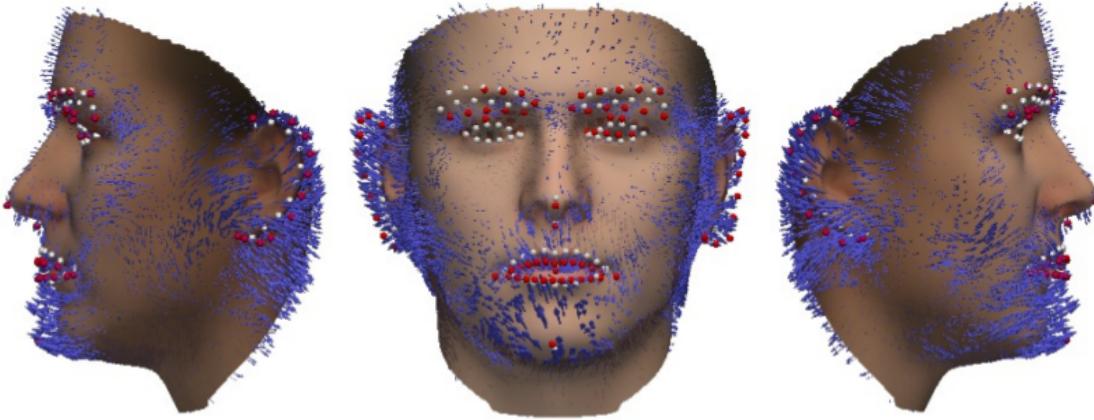


Figure 3.3: a sample **posterior** deformation field depicted with a set of **blue arrows** on the textured template face. The **white spheres** are the landmarks of the template mesh which are mapped on to the **red landmarks** of the target.

*make same images without line features and display instead*

in the template mesh. The posterior model is defined as the joint distribution of all template mesh points and the template landmarks, conditioned on the output displacement vectors for every template landmark with added noise.

$$\mathcal{D}|\mathcal{X} \rightarrow \mathcal{Y}_\varepsilon. \quad (3.10)$$

We now have defined a distribution over our template mesh. Sampling the conditional distribution and adding the resulting deformation onto the template creates deformed 3D surfaces of the template mesh which are fixed at the target landmarks 3.4.

## 3.5 Fitting & Optimization

In the previous paragraph we modelled the space of feasible deformations as a vector-valued Gaussian Process. Due to the fact that the posterior model is fixed at the target landmarks it is possible to perform the registration by drawing a shape from the posterior model. The sample, however, has to be optimized according to the shape of the target face. In order to find the deformation corresponding to the optimal fit  $d_*$  a linear optimization is employed.

$$d_* = \arg \min_{d \in \mathbb{D}} L[\mathcal{T}, \mathcal{M} \circ d] + \lambda R[d] \quad (3.11)$$

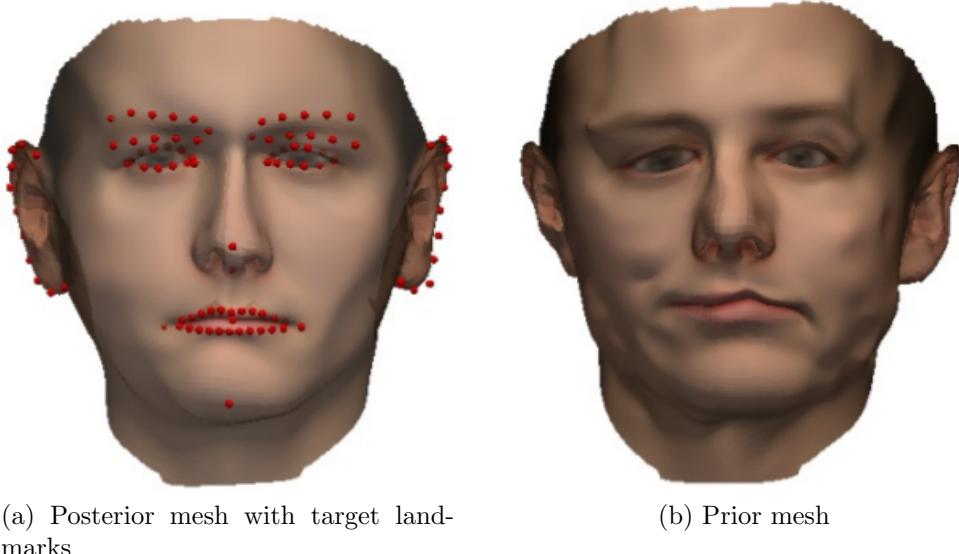


Figure 3.4: a) is a posterior mesh, obtained by adding a sample deformation field to the template. In comparison to the mesh in b) the mesh in a) is much more face-like. The reason for this is that the displacement of the landmarks has constricted the space of deformations to solely admissible deformations

Therein the deformation vectors of the landmarks and the posterior process act as constraints in the regularization term  $R$ . Minimizing the loss function  $L$  - for example the mean square error (MSE) - on the target and the deformed template should provide us with the optimal deformation under the given constraints.  $\mathbb{D}$  denotes the space of possible deformations in the posterior model.

The problem with the above approach to optimization is that the deformations are modelled by a distribution and are therefore non-parametric. In order to properly perform optimization, however, we need a representation that can be controlled through/ is governed by a set of parameters. For this purpose we would like to specify a parametric model  $M[\alpha](x) = \mu(x) + \sum_{i=1}^n \alpha_i \lambda_i \phi_i(x)$  which is defined over a finite set of basis functions. The question remains how the Gaussian Posterior Model can be expressed as a linear model. Fortunately, the answer lies in Mercer's Theorem. It states that a symmetric, non-negative definite, continuous function  $k$  can be expressed by the following linear combination of a countable sequence of functions  $\{\phi_i\}_{i \in \mathbb{N}}$

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(x') \in \mathbb{R}^{3 \times 3} \quad (3.12)$$

Through the theorem we can define the covariance function of our Gaussian Process Posterior in an infinite dimensional space made up of its eigenvalues  $\lambda_i$

and eigenvectors  $\phi_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3$  which are functions in this case. This infinite basis of functions corresponds to all possible deformations at a specific vertex in the template mesh. By evaluating a basis function at every vertex we receive a basis deformation field. We can further use Mercer's theorem to obtain the sought parametric model by estimating the covariance function. In order to achieve this a low-rank approximation of the covariance function - in terms of the first leading terms of its Mercer expansion - is performed  $k(x, x') = \sum_{i=1}^n \lambda_i \phi_i(x) \phi_i(x')$ . For this to work the modelled deformations have to be sufficiently smooth, because we are disregarding a number of basis functions. Smoothness is guaranteed by the square exponential covariance function. We can further express the deformation at a given mesh vertex by a linear combination of the selected basis functions and their eigenvalues with  $\alpha_1 \dots \alpha_n \in \mathbb{R}$  as the linear parameters

$$f(x) = \sum_{i=1}^n \alpha_i \lambda_i \phi_i(x) \quad (3.13)$$

In effect, we have formulated the Gaussian Process Registration problem in parametric form. Next, we want to minimize the residuals for all the points on the template surface according to optimal parameter values  $\alpha_i$

$$\arg \min_{\alpha \in \mathbb{R}^n} \sum_{x_i \in \mathcal{M}} (f(x_i) - \varphi_T(x_i))^2 \quad (3.14)$$

where  $f(x_i)$  is the deformation function and  $\varphi_T(x_i)$  returns the nearest point on the target mesh. The optimization problem therefore becomes

$$\alpha_* = \arg \min_{\alpha \in \mathbb{R}^n} L[\mathcal{T}, \mathcal{M} \circ \mathbb{M}[\alpha]] + \lambda R[\alpha] \quad (3.15)$$

Write as statistical PCA based shape model

# 4

## Registration Pipeline using Line Features

In this chapter we describe how the Vector-valued Gaussian Model is utilized in our implementation of a 3D Face Registration Pipeline. To enhance the registration outcome of this pipeline we use additional annotations to extend the prior correspondences given by the landmarks described in ???. These annotations are parametric curves, we call line features, that mark key regions on the 2D colored images of a face scan. In the following we provide a description of line features and their use as well as a description of the registration pipeline.

### 4.1 Line Features

Line features serve the purpose of augmenting the quality of registration by initiating it with a set of corresponding lines. They define correspondence in feature regions of the given face scans, i.e. the eyes and ears and lips. These regions have to be accurately mapped on to one another for the registration to produce acceptable results. For this purpose we model their contour lines with parametric curves.

We have 8 line features given for every scan we want to register. These have been marked on three images of every face, see 2.1. They are made up of a set of segments, each of which is modelled with a **Bézier curve** of a specified order, see 4.1. Bézier curves are often used in Computer Graphics for modelling smooth curves of varying order. Given a set of control points  $\mathcal{P} = \{P_0, P_1, P_2, \dots, P_n\}$  the Bézier curve through these points is given by

$$C(t) = \sum_{i=0}^n P_i B_{i,n}(t) \quad (4.1)$$

where  $B_{i,n}(t)$  is a Bernstein polynomial

$$B_{i,n}(t) = \binom{n}{i} (1-t)^{n-i} t^i \quad (4.2)$$

and  $t \in [0, 1]$  is the curve parameter. The line features consists of multiple segments of Bézier curves. Due to the nature of the objects depicted, there are open - eyebrow - as well as closed - eye - curves.



Figure 4.1: Line Features of the left eyebrow and eye, consisting of b  ezier curves defined by visible control points (white). The line features are marked on 2D color images taken during scanning procedure.

## 4.2 Sampling 3D Points from 2D Line Features

The line features provide us with additional prior information about the nature of the deformation field. In order to incorporate the line features in the Vector-valued Gaussian Model, we agreed to use them as additional point-wise information. The points are obtained by sampling the line features at discrete intervals resulting in a set of additional landmarks  $L_{Add} = \{l_1, \dots, l_N\}$ . These define the mapping  $\Omega : L_{AddM} \rightarrow L_{AddT}$  of the line features in the template face mesh on to those in the

target face mesh.

Establishing correspondence for a pair of ears for example is very difficult, even for the trained eye, because of their very different topology. The structure of the concha is very complex. For this reason, we define correspondence between lips, eyes and ears we through the contour lines of their rims. Further modelling the ratio between the different parts of these contours would again be tedious. Therefore, we simplify the assumptions for similarity to an equidistant parametrization of the line features so that the mapping  $\Omega$  is approximately plausible. In effect, when a curve is sampled at  $n$  points, these  $n$  points are all at equal parametric intervals.

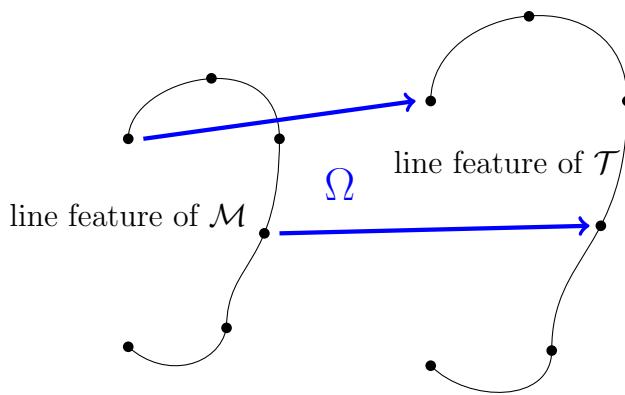


Figure 4.2: Mapping of equidistant samples of **ear** line features from the reference (left) on to the target (right)

## Arc Length Parametrization

The underlying parameter  $t \in \mathbb{R}$  of a b  ezier curve segment of a single line features is dictated by the curve's velocity. In order for us to be able to sample the line feature in equidistant intervals, however, the parameter has to be linear in respect a the curve segment's length. In consequence, we have to compute an equidistant parametrization for each of the curve segments of a line feature before sampling it. The solution to this problem is based on the respective curve's arc-length, which is defined as the length of the rectified curve. The underlying parameter  $t$  must then correspond - at every point of the curve - to the ratio between the length of a fraction of the curve  $l$  and the total curve length  $L$ :  $t = \frac{l}{L}$ .

**In theory** It is possible to get the arc length  $L(t) = \int_{t_0}^{t_1} |C'(t)| dt$  for given parameters  $t_0, t_1$  where  $C'(t)$  is the derivative of the curve  $C : t \in [0, 1] \rightarrow \mathbb{R}^2$ . We, however,

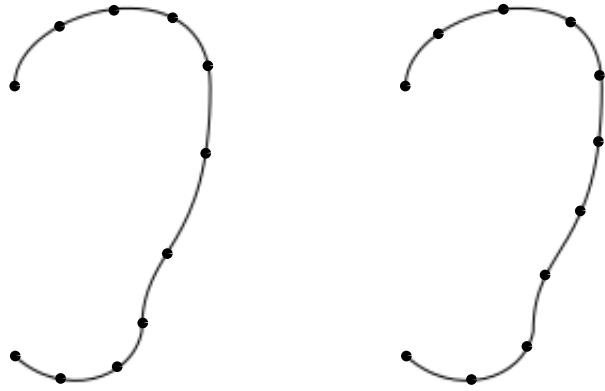


Figure 4.3: a simplified - the depicted **ear** line feature is treated as one sole Bézier segment - illustration of the difference between Bézier (**left**) and equidistant (**right**) parametrization.

want to find a reverse mapping from the length of a fraction of the curve  $l$  to the curve parameter  $t = L^{-1}(l)$ . This mapping can of course be derived analytically, but it is far easier to implement it using a numeric approximation.

**In practice** As we are not in need of a subpixel resolution, we can skip the formal math and use a lookup table to compute the arc-length. First, we calculate a large number of points on each segment of the curve using the parametrization of the corresponding Bézier curve. For each sampled point, we save a key-value-pair to a new entry of the lookup table. The approximate distance of the point from the origin of the segment is the key, while the point's coordinates is the value. The distance is approximated by summing up the Euclidean distances of each point to its respective predecessor starting from the origin. The resulting lookup table contains the approximate distances and coordinates of a large number of points from the origin of a curve segment. Assembling the segments' lookup results in the lookup table of the whole curve where the last key represents the curve's arc-length.

In a second step, the curve can easily be sampled by computing the length of parametric intervals  $\frac{L}{N}$  for a specified number of points  $N$ .  $l = k \cdot \frac{L}{N}$  returns the current length of the curve for the sampling point of index  $k$ , where  $k = [0, \dots, N]$  for open curves and  $k = [0, \dots, N - 1]$  for closed curves. To get the point coordinates for a fraction of the curve, we now simply perform a binary search on the lookup table for its approximated distance from the origin  $l$ . We choose the index which returns

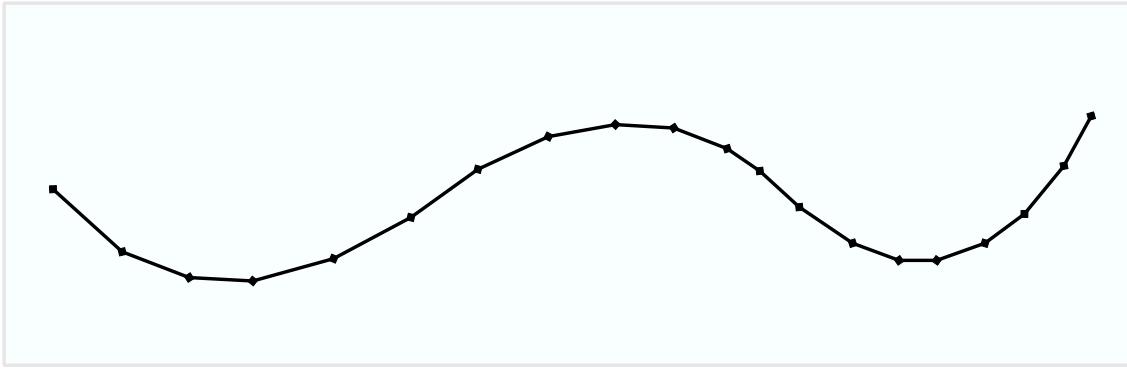


Figure 4.4: Visualization of euclidean distances between points computed with Bézier curve parametrization. This is an arbitrary line feature used solely for demonstration purposes.

the coordinates for the exact fraction length and if this requirement can't be met the index with the next smaller length. The returned point coordinates are used as the approximate coordinates of the sought sample point.

### 3D Mesh Projection of Sampled Points

Having implemented arc length parametrization it is possible to draw an arbitrary amount of sample points  $p \in \mathbb{R}^2$  from the 2D line features. These are defined as a set of points  $S \subset \mathbb{R}^2$ . Our goal is, however, to use these additional points as landmarks describing the features on the 3D mesh of the corresponding face. For this to work, we have to project the sampled points of each line feature on to the corresponding face mesh in order to obtain their approximate 3D representation, because we have no information on the depth of the line features. In the camera model - derived from the calibration of the scanner - the image is located on the viewing plane or viewport opposite of the focal point. The direction of the 3D representation of a point on a curve is given by the vector defining the position of the point on the viewing plane from the perspective of the focal point of the camera. We have the discrete mesh of the scan given and seek a mesh vertex that is the most accurate representation of a sample point on the 2D curve. The dot product of their normalized directions is used as a similarity measure. In order to find vertices with similar directions, we save all the distances of mesh vertices for which the similarity measure is higher than a specified threshold in a list. We then select the distance of the vertex with the maximum similarity. Finally, we project the distance of this vertex onto the direction of our sample point and thereby obtain an approximation

of the points position in the mesh. A new vertex is created at this location.

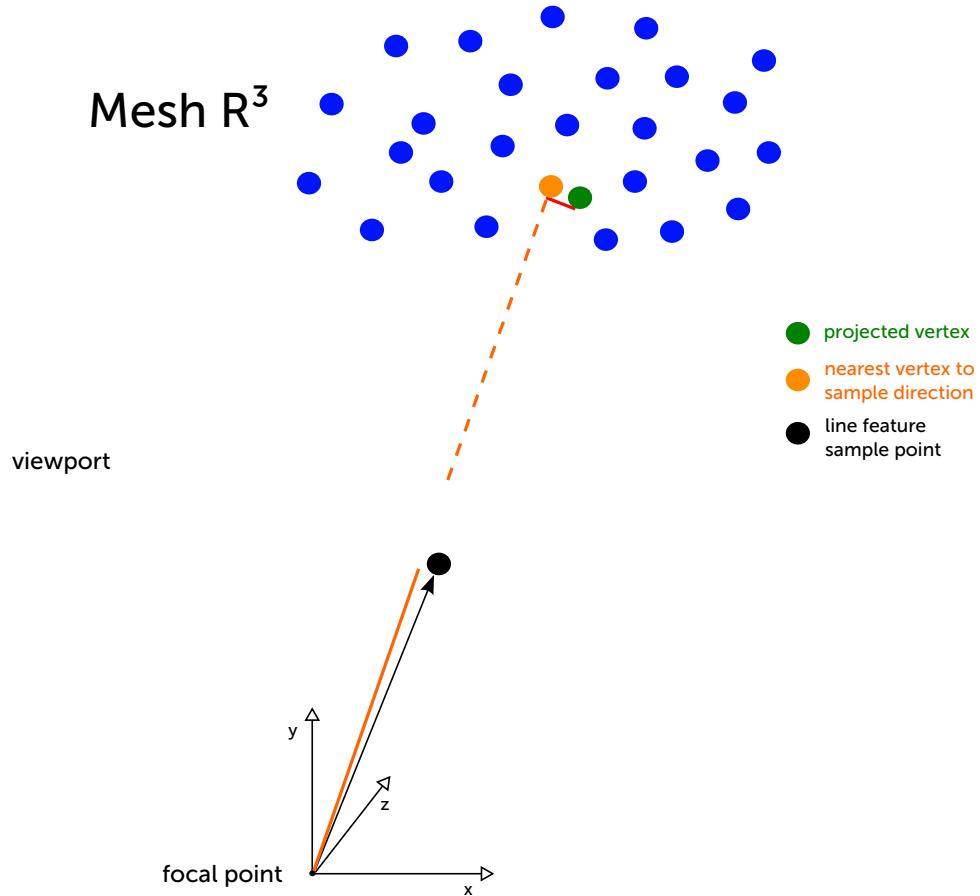


Figure 4.5: shows the projection of a sample point - on a 2D line feature - from the viewport onto a triangulated 3D mesh. The distance vector of the vertex with the most similar direction vector (orange) is projected (red) onto the direction of the sample point (black arrow), resulting in the projected vertex (green)

## Preparing the Template Mesh

For the line features projected on to the scans to be of use, line features also have to be marked on the template mesh. We use the mean mesh of the 3D Morphable Model of the Graphics and Computer Vision Group at the University of Basel as the template mesh. It has had its holes manually filled in and has had a texture applied to it. With it comes a set of landmarks larger than that of the scans.

The mean mesh has to be rendered with three different camera calibrations for marking the line features. The 2D line features can then be projected back onto the mesh using these calibration parameters. Through the projection process equidistant line feature samples to the set of landmarks. We therefore have provided the registration algorithm with additional prior information to define the deformation field by and to produce better fitting results.



Figure 4.6: mean mesh of the 3DMM



Figure 4.7: overlap of textured mean template with gray target scan

### 4.3 Rigid Mesh Alignment

Before we can start creating Gaussian Models, we first have to ensure that for each registration template and target mesh are aligned in the same coordinate system. After all, we want to model the variability of different faces without incorporating an additional offset. We therefore have to perform a rigid transformation consisting of a translation and a rotation to align the template and target meshes according to their landmarks. To compute the transformation we use the set landmarks pairs whose identifiers exist for both meshes. Before the transformation is applied the targets have to be clipped at the neck and around the ears where the scanner has left artifacts. The computed transformation is then applied to all vertices of the respective face scan to align it with the template. The aligned mesh pair serves, see fig. 4.7, as the starting point for the actual registration.

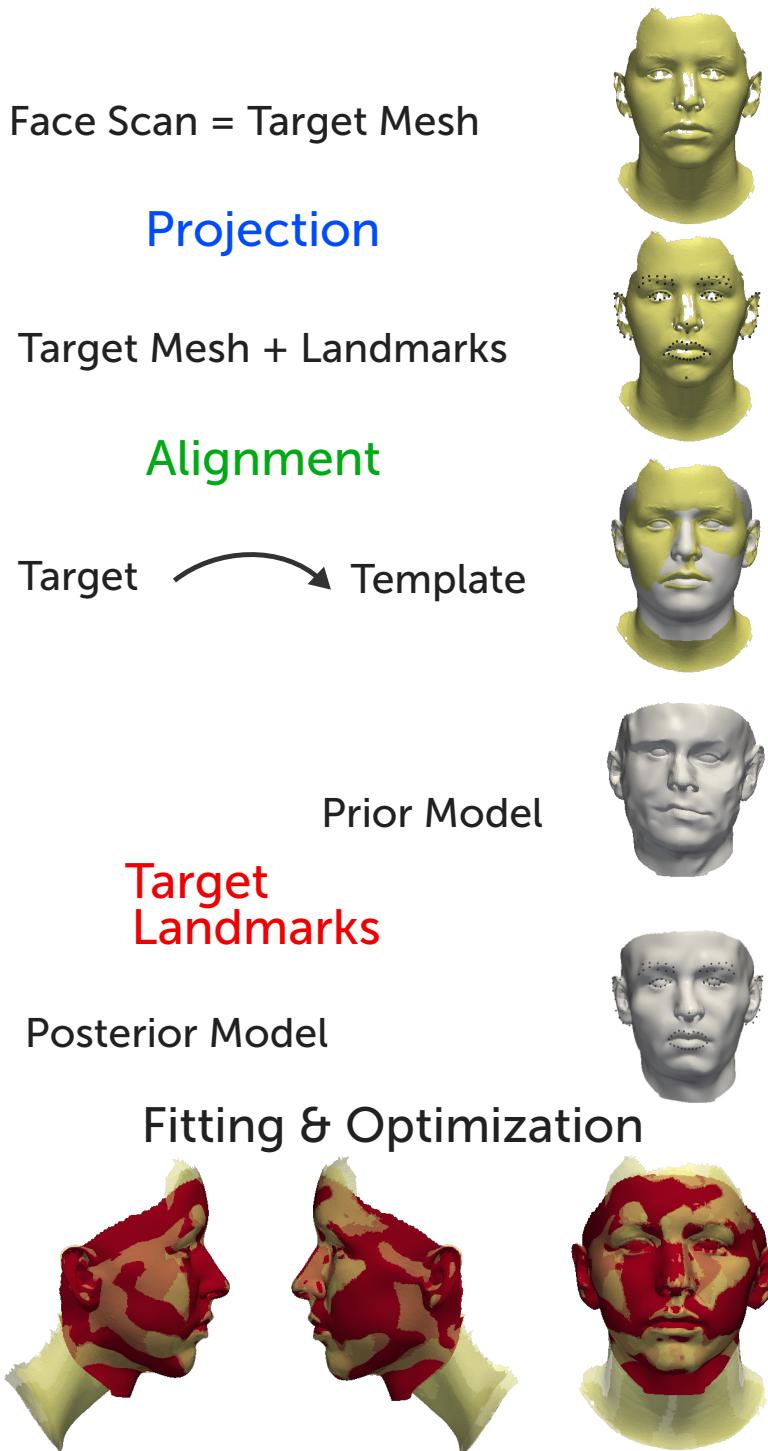


Figure 4.8: an overview of the different stages of the registration pipeline with an exemplary visualization of face mesh at every stage.

We are now set for the actual registration involving the computation of the Prior and Posterior Gaussian Models and the Fitting of the Posterior Model to a face scan. For the definition of the Gaussian Process distributions we use the software framework

statismo ???. It is a framework for PCA based statistical models. These are used to describe the variability of an object within a population, learned from a set of training samples. We use it to generate a statistical model of the template mesh equaling the parametric representation of the Gaussian Process Posterior described in sec. ??.



(a) sample of the GP Prior



(b) sample of the GP Prior

Figure 4.9: Prior faces computed from samples drawn from the Gaussian Process Prior Model. The deformations are not truly face-like, because they are modeled only through the covariance of the template mesh vertices.

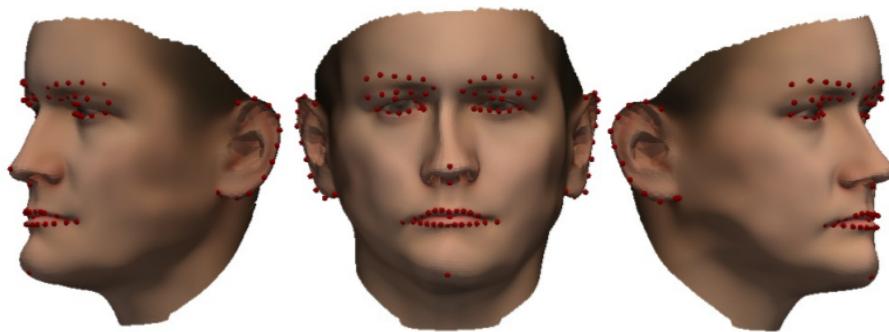
## 4.4 Prior Model

After aligning the target with the template mesh, we want define a distribution of possible deformations of the template mesh, as described in ???. This Gaussian Process (GP) Prior distribution is represented as a statistical model and the parameters are saved on to disk. This allows for samples to be drawn from this representation. These samples are 3D face meshes, prior faces in fig. ???, that are the result of the deformations generated by the GP Prior and then added to the template mesh. The

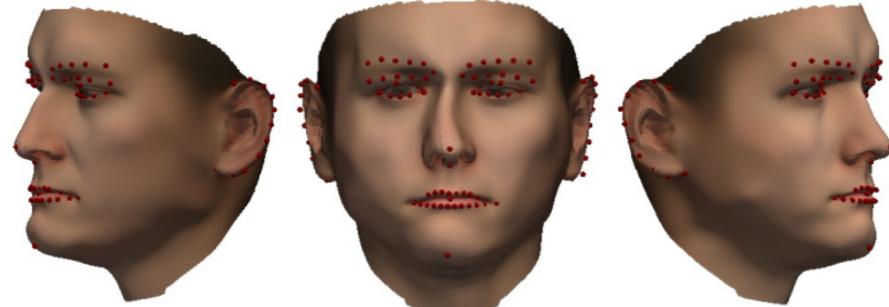
deformations are up until now defined solely on the ground of the covariance of the template mesh points.

## 4.5 Posterior Model

The deformations generated by the GP Prior are of course not admissible. We receive admissible deformations by adding the target landmarks and the additional set of sampled line features to the GP Prior distribution. Using statismo we create a statistical model, which is partially fixed at the target landmarks, from the template mesh. This time we have incorporated the fixed displacements of the landmarks and in return get obtain a model that leads to meshes that a more face-like.



(a) sample of GP Posterior



(b) sample of GP Posterior

Figure 4.10: Posterior faces computed from samples drawn from the GP Posterior. The resulting faces are considerably more face-like than those in fig. ??

## 4.6 Fitting

Using statismo to represent the parametric representation of the Gaussian Process Posterior Distribution Perform optimization ***Schema of ITK optimizer***

What are we optimizing? L2??? – $\downarrow$  not robust to outliers, protruding regions.

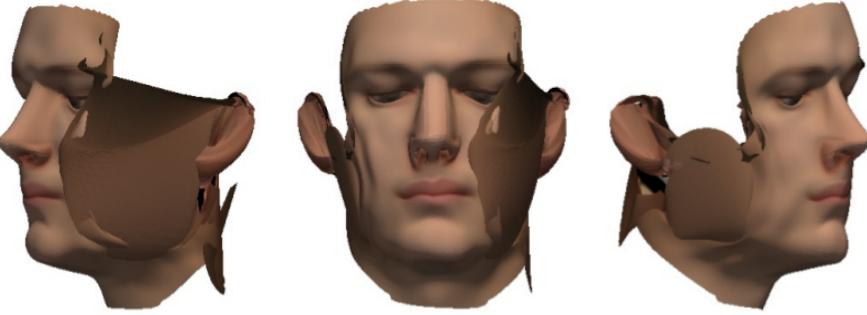


Figure 4.11: Mean Squares as distance metric

## 4.7 Robust Loss Functions

After the alignment of template and target mesh, the template protrudes over the target on the side of the head, as can be seen in fig. ???. Performing optimization as described in ?? using a simple Mean Square Error(MSE) as a distance measure between the template and target mesh penalizes the protruding regions of the template with a strong gradient towards the rims of the template and therefore causes strong distortions. Our approach to tackling this problem was to try out a range of different robust estimators, namely the Tukey, Huber, and Fair estimators. The advantage of these estimators lies therein that they are less sensitive to outliers, reducing registration artifacts considerably. (Outliers are in this case template mesh points that farther away than a certain threshold from the next point on the target mesh. However, as can be seen from the formulas, these techniques require finding appropriate parameters first which produce reasonable/acceptable visual results.

Fair

$$\rho(x) = c^2 \left[ \frac{|x|}{c} - \log\left(1 + \frac{|x|}{c}\right) \right] \quad (4.3a)$$

$$\psi(x) = \frac{x}{1 + \frac{|x|}{c}} \quad (4.3b)$$

Huber

$$\rho(x) = \begin{cases} \frac{x^2}{2} & \text{if } |x| < k \\ k(|x| - \frac{k}{2}) & \text{if } |x| \geq k \end{cases} \quad (4.4a)$$

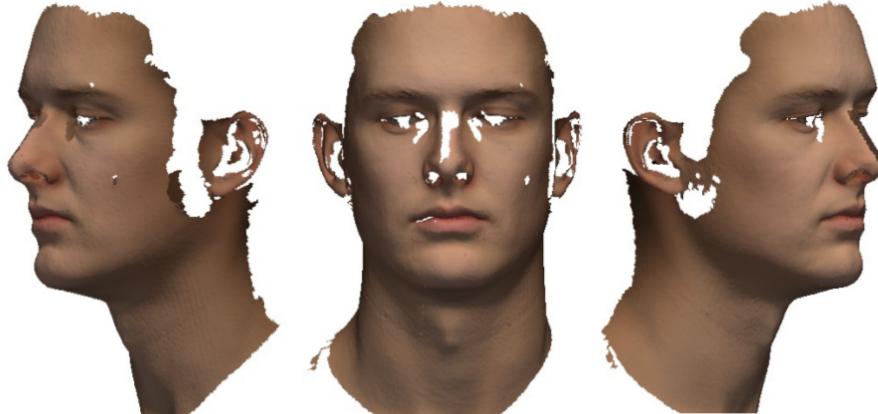
$$\psi(x) = \begin{cases} x & \text{if} \\ k sgn(x) & \text{if} \end{cases} \quad (4.4b)$$

Tukey

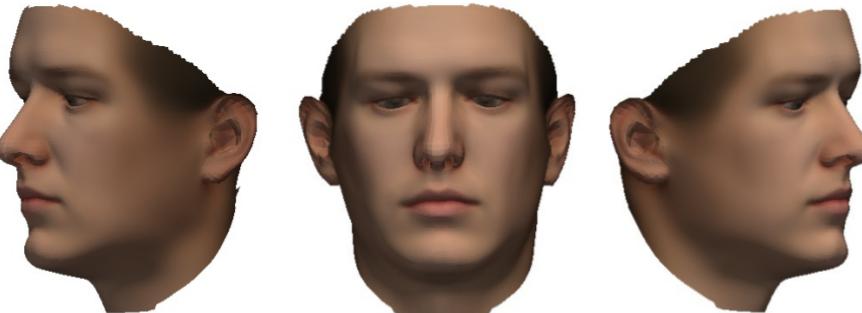
$$\rho(x) = \begin{cases} \frac{c^2}{6} \left( 1 - \left[ 1 - \left( \frac{x}{c} \right)^2 \right]^3 \right) & \text{if } |x| \leq c \\ \frac{c^2}{6} & \text{if } |x| > c \end{cases} \quad (4.5a)$$

$$\psi(x) = \begin{cases} x \left[ 1 - \left( \frac{x}{c} \right)^2 \right]^2 & \text{if } |x| \leq c \\ 0 & \text{if } |x| > c \end{cases} \quad (4.5b)$$

*for each estimator show a sequence of fits for different parameters and 3 different meshes?*



(a) target with mean texture applied



(b) optimized mean posterior fit

Figure 4.12: The fitting result in a) was obtained with the Tukey estimator, parameter  $c=7.0$ . The texture of the mean mesh has been applied to the target to make the meshes comparable. The fit in a) has considerable likeness to the target b), although it lacks expressiveness

# 5

## Results

### 5.1 Sampling

**Problems & Inaccuracies** The face meshes used in the face registration pipeline contain large holes around the ears and the eyes. In effect, the projected sample points are off target, because now the mesh vertex with the most similar direction is likely farther away at a suboptimal location. This circumstance leads to the projected line not clearly being distinguishable as a contour line. On different data sets the performance of the projection of the line features for a large number of samples, i.e. 30, varied significantly. Overall one can say that the distortion of a projected line increases with the amount of sample points. *Compute some landmarks with 30 samples up close image of eye holes of face scan* An easy workaround was to reduce the number/amount of sample points. By using only 5-10 sample points per curve some datasets rendered near perfect results on a “control dataset”. *Compile list of datasets, show samples* However, when the holes are too large this workaround also fails. This circumstance leaves room for discussion. As long as the method is dependent on the data from the scans - the size of the holes in the meshes - it lacks generality and generality is exactly the basis for feasible and reproducible registration results.

### 5.2 The effect of Line Features

(b)  
**with-**  
**gat**  
**with**  
**fexes,**  
**twes,**  
**with**  
**texe**  
**ture**

Figure 5.1: Eyes

(b)  
**with-**  
**gat**  
**with**  
**fexes,**  
**twes,**  
**with**  
**texe**  
**ture**

Figure 5.2: Ears

(b)  
**with-**  
**gat**  
**with**  
**fexes,**  
**twes,**  
**with**  
**texe**  
**ture**

Figure 5.3: Nose

## 5.3 Registration quality

### Visualized Distance

with line features

### Caricatures

Caricatures rendered by adding scaled displacement fields

00125 - scan of old man - wrinkles can't be modelled through this smooth definition of covariance - adjust covariance function

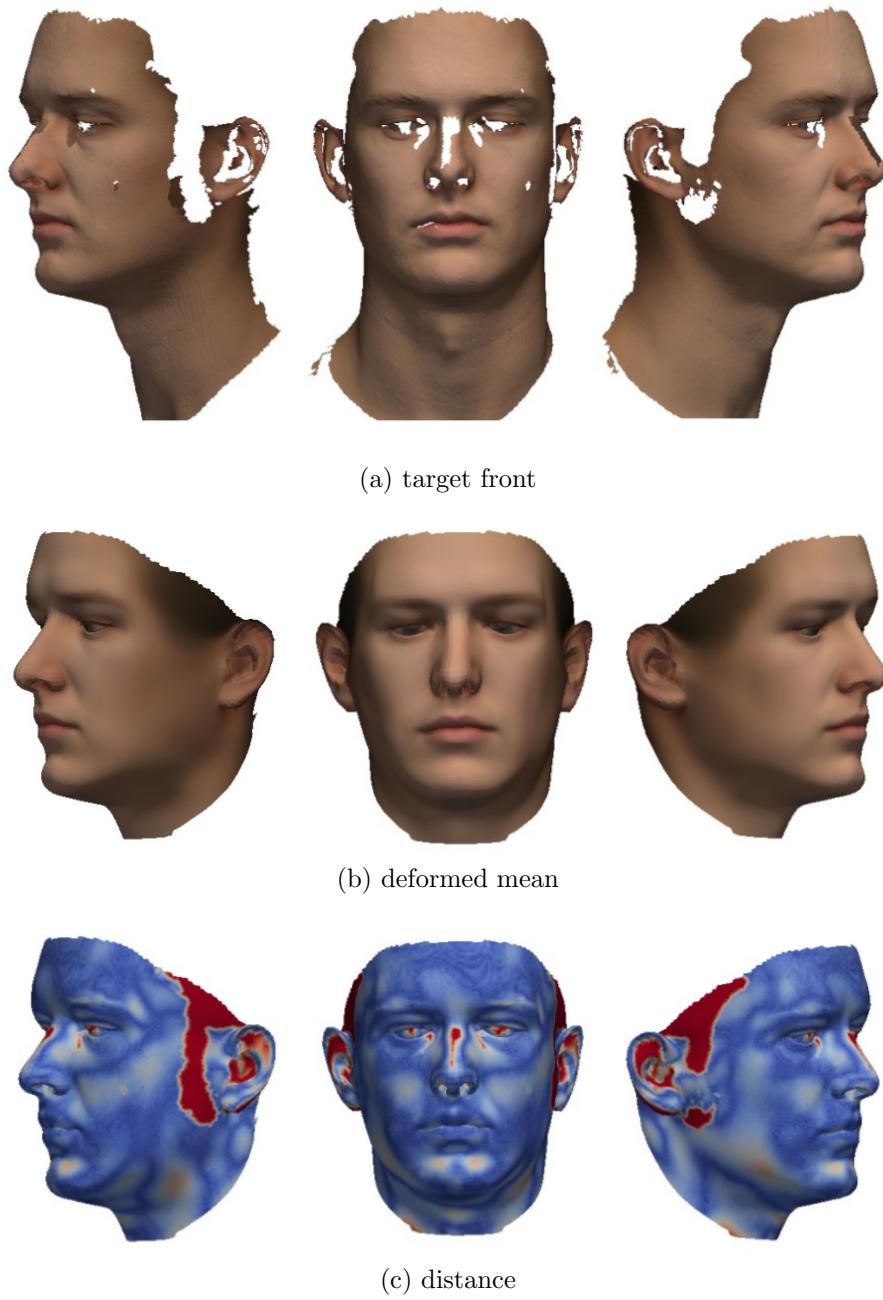


Figure 5.4

(b)  
dis-  
gained  
front

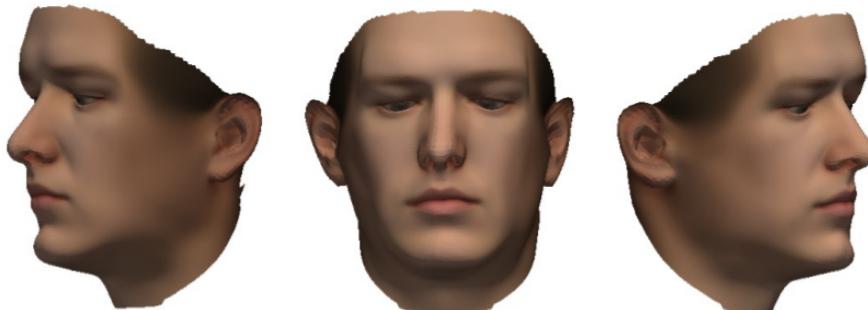
(f)  
dis-  
gained  
leftan

(h)  
dis-  
gained  
rightan

Figure 5.5

(b)  
dis-  
gained  
mean

Figure 5.6



(a) 130% scale



(b) 160% scale



(c) 200% scale

(d) 130% scale

(e) 160% scale

(f) 200% scale

Figure 5.7