Using line features for 3D face registration

Fabian Brix

18.06.2013

Abstract

In this bachelor thesis we present the construction of a 3D face registration pipeline in a two-facetted approach. Firstly, we derive the registration algorithm from Gaussian Process Regression. Secondly, we augment registration quality by passing line correspondences to the algorithm as prior information. Gaussian Process Regression can be used to sample deformation fields that map any two shapes on to one another using a set of corresponding prominent points on the shapes as prior information. In our approach we implemented registration of face scans using additional information gained by sampling 3D points from line features that define key features of the faces, i.e. the eyes and the ears. The deformation field is optimized with the prior information used as constraints to obtain a near-optimal mapping of the template on to the target. As will be shown the use of line features has a profound impact on the accuracy and quality of the registration results. In order to use this approach for model building, however, optimization of the expressiveness of the deformed template is still necessary.

Contents

Contents **3**

1 Introduction **4**
1.1 Problem Statement 4
1.2 Review Literature 5

2 3D Model Building **6**
2.1 3D Morphable Model 6
2.2 Achieving Correspondence through Registration 7
2.3 Available Data 8

3 Gaussian Processes in 3D Face Registration **10**
3.1 Stochastic Processes 10
3.2 Gaussian Processes 10
3.3 Gaussian Process Regression 12
3.4 Application to 3D Face Meshs 14
3.5 Fitting & Optimization 16

4 Registration Pipeline using Line Features **18**
4.1 Line Features 18
4.2 Sampling 3D Points from 2D Line Features 20
4.3 Rigid Mesh Alignment 24
4.4 Prior Model 26
4.5 Posterior Model 26
4.6 Fitting 26
4.7 Robust Loss Functions 27
4.8 Varying the Variances 28

Chapter 1

Introduction

1.1 Problem Statement

Beschreibung des Problems chronologische, kurze Beschreibung des Vorgehens Fachterminologie so allgemein wie moglich whlen 1. Use Gaussian Processes - 2. Use Line Features =¿ prepare for Gaussian Process Regression In this bachelor thesis Implement 3D face registration using Gaussian Processes and Line Features. One part of the problem is to sample equidistant 3D points from 2D line features marked on images of a 3D face scan. These line features should then be used as an additional input to a registration algorithm which is based on Gaussian Process Regression. The aim is to build a pipeline which starts off with the raw scan data as well as the landmarks and line features. The feature points are used to register the mean face of the MM/BFM (Basel Face Model) on to/with the raw scan thereby obtaining a fully defined and textured 3D model representation of the face in 3D. Registration is the technique of aligning to objects using a transformation, in this case the registration is performed by adding displacements to every points in the mean face model. A model is represented as vector N*d. What is a model? A vector representation of a 3D scan? For the morphing a Posterior Shape Model is used in combination with a Gaussian Process. Image registration is a process of aligning two images into a common coordinate system thus aligning. (gaussian process + line features for accurate, reproducable registration)

1.2 Review Literature

2. Definition of terms (morphable model, 3D face registration, Gaussian Process regression, posterior shape models) 3. Review of literature (papers) Inputs: inspired by books Stochastic yada yada, Gaussian Processes in Machine Learning: Rasmussen et al, the work done by the Graphics and Vision Research Group Uni Basel, papers Using Landmarks as a Deformation Prior for Hybrid Image Registration (for GP), A unified formulation of statistical model fitting and non-rigid registration (Fitting and Optimization), Posterior Shape Models (Posterior Shape Models)

Chapter 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 fassion. New shapes and textures can now be constructed/conceived through a linear combination of the set of m available faces governed by shape and texture coefficients $\vec{\alpha}, \vec{\beta} \in \mathbb{R}^m$.

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

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 $\overline{S} \in \mathbb{R}^{3N}$ and textures $\overline{T} \in \mathbb{R}^{3N}$ 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 $R_S = S - \overline{S}$ and texture $R_T = T - \overline{T}$ averages. The covariance matrices are then used to perform two Principal Component Analysis (PCAs) which define basis transformations to two orthogonal coordinate systems the axis of which are the eigenvectors of the respective covariance matrices.

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

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).

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

By observing the likelihood of the coefficients it is now possible to find out how likely/admissible/plausible the appearance of a corresponding face is. ***likelihood of the faces p(faces — unknown parameter values) OR what are the most likely parameters given the available set of faces***

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 oneanother, for example a nose is combined with another nose and not with a mouth. For this reason, the meshs 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 meshs, i.e the left corner of the left eye, have the same vertex number. The process of calculating a dense point-to-point correspondence between two meshs 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 Algorithm When performing registration, we start with a template shape that is a prototype of the semantic parametrization that should be established for all data sets (Prototype created manually?). In order to compute the parametrization for another shape we deform the template to match this shape,

which is then referred to as the target. Both are considered as matched if the semantic points are mapped onto each other satisfying a specified similarity measure. If a dense point-to-point 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.

A registration algorithm uses prior information in the form of manually clicked feature points at prominent points of the face, so called landmarks, on all of the face meshs for the purpose of computing an accurate deformation. Correspondence in-between these points is defined through smooth deformations of the template mesh which match the surface and feature points of the target. In this thesis we present a registration algorithm which is novel to the problem of 3D face registration in two ways: the deformation is modeled using Gaussian Process Regression, a method from the field of Machine Learning and the use of prior information is extended to parametric curves marking complex regions of the face, refered to as line features.

2.3 Available Data

For testing the registration algorithm we have given a set of about 300 face scans - male as well as female - all of them recorded at the University of Basel by an active stereo vision system with light sources and structured light, engineered by the company ABW-3D. The scanner records four 3D surfaces, called shells, for every face and further records three high resolution color images. The scan data consists

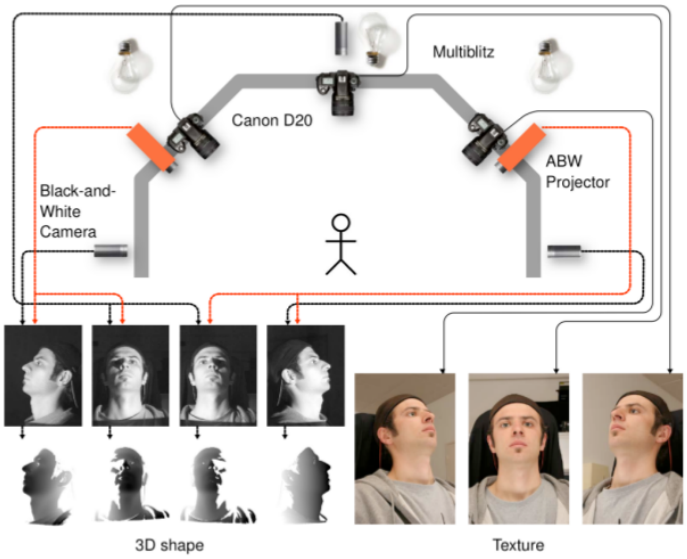


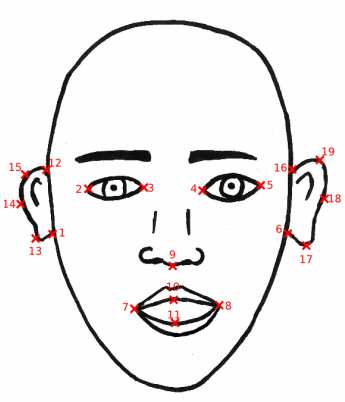
Figure 2.1: schema of the structured light 3D scanning system employed to record the facial scans. It consists of two projectors that project triangle gradient patterns on to a face from opposing angles, three black-and-white cameras used for the triangulation and three digital cameras which record the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

of four seperate 3D surfaces, called shells, and three high resolution images for the texture of the skin. The 3D shells are obtained through triangulation. Each of the four black-and-white cameras 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.

(a) 3D shells (b) merged shells

Figure 2.2: ...

show shells To obtain a whole mesh the 3D shells first have to be cleaned and then merged. Additionally to the scans, we use manually annotated data. A set of prominent points, so called landmarks, was given for every face. ***show landmarks How express that landmarks were created after feature points to enhance registration?*** Furthermore, we use the color images 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.



Chapter 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 Ω 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 Ω_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 $\mu : \mathcal{X} \to \mathbb{R}$ and covariance $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ 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.

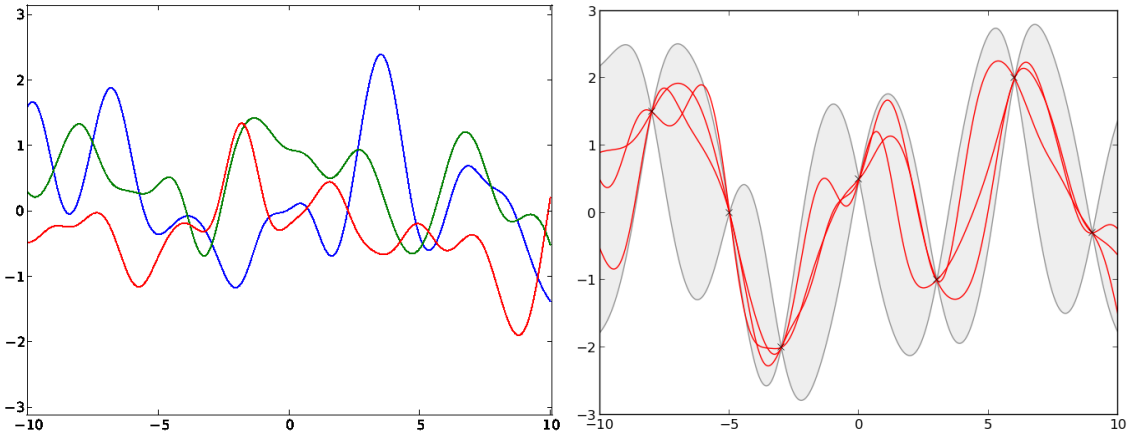
Covariance Functions The key feature of a Gaussian Process is its covariance function also known as "kernel". It specifies the covariance $\mathbb{E}[f(x)f(x')]$ between pairs of random variables for two input vectors x and x', allowing us to make assumptions about the input space by defining the spatial 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 $cov(f(x), f(x')) = k(x, x') = \exp(-\frac{(x-x')}{2l^2})$. (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.

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

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 indead a distribution over functions, see figure 3.1. The GP Prior forms the basis for inference in Gaussian Process Regression.



(a) Samples drawn from the prior (b) Samples after a few observations

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 $X_1, X_2, \ldots, X_k, \ldots, X_n$ are now d-dimensional vectors, yielding a covariance function of the form $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{d\times d}$ 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 meshs 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 meshs. 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 $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ 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:

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

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:

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

Prediction with Gaussian Noise Model In most real world applications

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

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

$$\mathbf{y}_*|\mathbf{f} = \mathbf{y} \sim \mathcal{N}\left(\overline{\mathbf{y}}_*, \Sigma(\mathbf{y}_*)\right) \tag{3.5a}$$

where the mean depends on the observed training targets

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

whilst the covariance depends only on the input points

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

Conclusion, how does this help us to proceed?

3.4 Application to 3D Face Meshs

In this section of 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 $\mathcal{D} : \mathcal{M} \subset \mathbb{R}^3 \to \mathbb{R}^3$ 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 meshs 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***

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

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

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

The template mesh is defined by a set of vectors $\mathcal{X} \in \mathbb{R}^3$ and a set of landmarks $L_{\mathcal{M}} = \{l_1, \cdots, l_n\} \subset \mathbb{R}^3$. ***Introduce landmarks in model building*** The mean vector μ 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

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

meaning that a sample deformation field can be directly drawn from the prior distribution of the template mesh. ***show two or three samples of prior here, next to template/mean mesh or reference images in chapter4???***

Reference Mesh Posterior The target landmarks also consist of a set $L_{\mathcal{T}} = \{l_{\mathcal{M}1}, \cdots, l_{\mathcal{M}N}\} \subset \mathbb{R}^3$. 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

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

Is this a correct definition for the distribution?

The deformation model is now rendered fixed at certain landmark points 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 deformations at the template landmarks co-define the distribution of possible deformations at all vertices of 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 deformation vectors for every template landmark with added noise.

$$\mathcal{D} | \mathcal{X} \rightarrow \mathcal{Y}_\varepsilon. \tag{3.10}$$

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 template mesh which are fixed at the target landmarks. ***What about maximum a posteriori estimate? Does it return mode, most likely sample of the distribution? show images of mean, prior and posterior with added landmarks***

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.

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

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 $\mathbb{M}[\alpha](x) = \mu(x) + \sum_{i=1}^{n} \alpha_i \lambda_i \phi_i(x)$ 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 $\{\phi_i\}_{i \in \mathbb{N}}$

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

Through the theorem we can define the covariance function of the our Gaussian Process Posterior in an infinite dimensional space made up of its eigenvalues λ_i and eigenvectors $\phi_i : \mathbb{R}^3 \to \mathbb{R}^3$ 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 $\alpha_1 \cdots \alpha_n \in \mathbb{R}$ as the linear parameters

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

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 α_i

$$\underset{\alpha \in \mathbb{R}^n}{\textbf{arg min}} \quad \Sigma_{x_i \in \mathcal{M}} (f(x_i) - \varphi_T(x_i))^2 \tag{3.14}$$

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

$$\alpha_* = \underset{\alpha \in \mathbb{R}^n}{\textbf{arg min}} \quad L[\mathcal{T}, \mathcal{M} \circ \mathbb{M}[\alpha]] + \lambda R[\alpha] \tag{3.15}$$

Write as statistical PCA based shape model

Chapter 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 additionally enhance the registration outcome of this pipeline we use face data where the key regions have been marked with contour lines. In the following we provide a description of line features and their use as well as a specification of the registration pipeline. ***pipeline type important?***

4.1 Line Features

We use a large set of corresponding points... Line features serve the purpose of augmenting the quality of registration by initiating it with a larger set of corresponding points, by sampling points from the lines themselves. They are used to mark complex regions of the face, i.e. the eyes and ears, so that the registration process produces an accurate mapping of the contours of these organs which would otherwise not be possible. Without the prior information provided by line features, an accurate mapping in these regions is hard to achieve, because they have a dense abundance of points, while regions like the cheeks are scarcely defined by points.
the following is some old text ready to incorporate The idea behind the use of sampled points from the line features was to have more point correspondencies in complex regions as for example the eyes and the ears where there is a great abundancy of pixels and the algorithm isnt likely to create a flow field which is accurate not enough to describe these regions ***Und sehr spezifische zuordnung, wenn die Augenkante versetzt ist, sieht man das sofort***, because of its smoothness

constraint.

We have 8 contours given for every scan we want to register. These have been marked on three images of every face, see Fig section 1.3. The contours we call line features depict the eyebrows, eyes, ears and lips of a face. They are made up of a set of segments, each of which is modelled with a **Bézier curve** of a specified order. Bézier curves are often used in Computer Graphics for modelling smooth curves of varying order. Given a set of control points $\mathcal{P} = \{P_0, P_1, P_2, \ldots, P_n\}$ the Bézier curve through these points is given by

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

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

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

and $t \in [0, 1]$ is the curve parameter. The Bernstein polynomials of degree n form a basis for the power polynomials of degree n. Due to the nature of the objects depicted, there are open as well as closed curves.

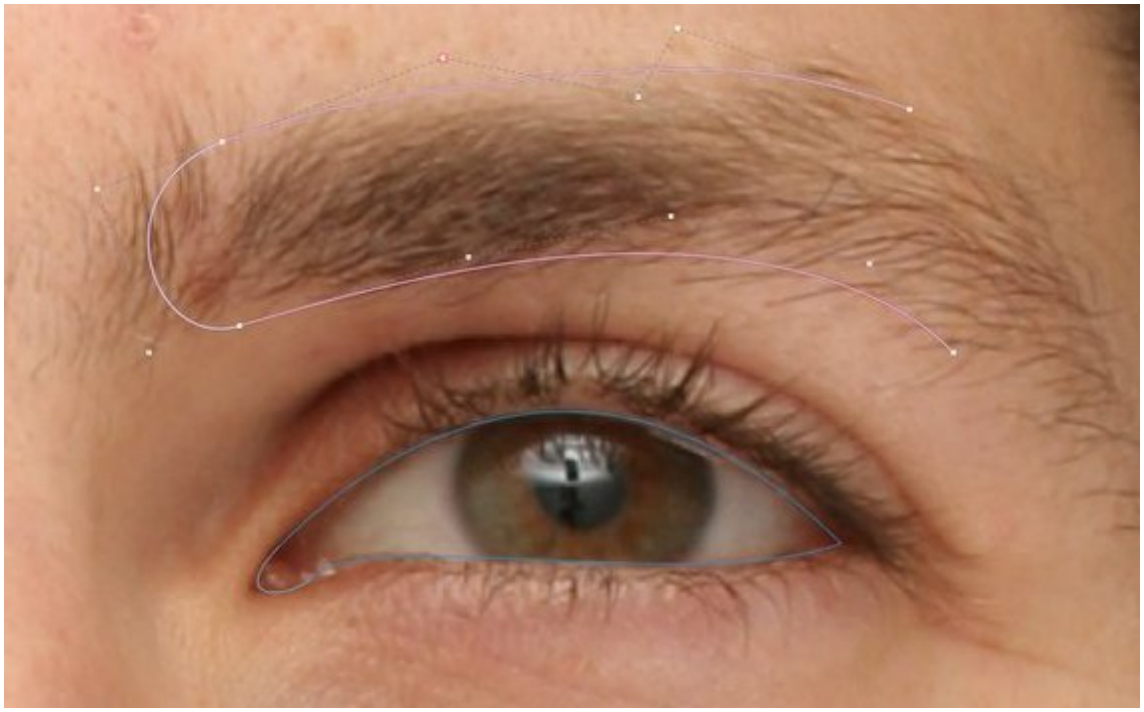


Figure 4.1: Line Features of the left eyebrow and eye, consisting of bézier curves defined by visible control points (white)

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. In order to get this point-wise information the line features are sampled at discrete intervals resulting in a set of additional landmarks $L_{Add} = \{l_1, \cdots, l_N\}$. These define the mapping $\Omega : L_{Add\mathcal{M}} \rightarrow L_{Add\mathcal{T}}$ of the contours - describing the different important features present in the faces - in the template face mesh on those of the target face mesh. In order for the mapping Ω to be approximately plausible, we choose an equidistant parametrization. In effect, when a curve is sampled at N points, these N points are all at equal parametric intervals. ***equidistant parametrization is not a fact, it is a choice. Different ears have different topology?***

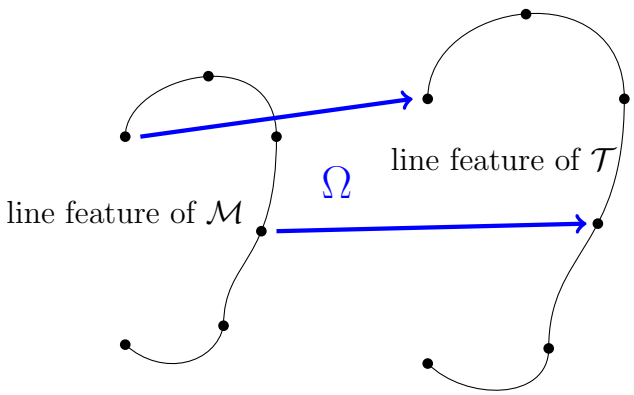


Figure 4.2: Mapping of equidistant samples of **ear** line features from the reference (left) on to the target (right)

Arc Length Parametrization

We decided to use equidistant parametrization, but bezier curves don't allow it. The first problem which becomes apparent when trying to sample the line features is that the bézier curve segments don't allow for equidistant parametrization, because the underlying parameter $t \in \mathbb{R}$ is not linear in respect to the length of the curve. The growth of the parameter of a bézier curve is instead dictated by velocity.
Consequently, the imperative must be instead to evaluate the curves based on their arc-length, which is defined as the length of the rectified curve. The underlying parameter must then correspond - at every point of the curve - to the ratio between the length of the part of the curve that has been traversed and the total curve length.

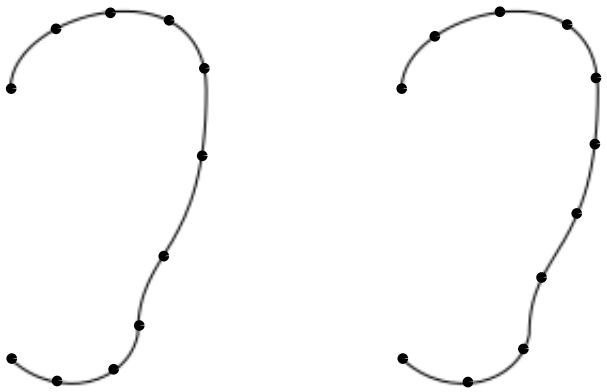


Figure 4.3: a simplified - the depicted **ear** line feature is treated as one sole bézier segment - illustration of the difference between bézier (**left**) and equidistant (**right**) parametrization.

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 $C : t \in [0,1] \to \mathbb{R}^2$. We, however, want to find a reverse mapping from the length of a fraction of the curve 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 point we save its approximate distance from the origin of the segment as a key into a new slot in the lookup table of this segment, while its coordinates act as the slot 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 approximative distances and coordinates of a large number of points from the origin of a curve segment. Assembling the segments' lookup tables gives us the table table for the whole curve with the last key representing its arc-length.
Second, the curve can now 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 $k = [0, \cdots, N]$ for open curves and

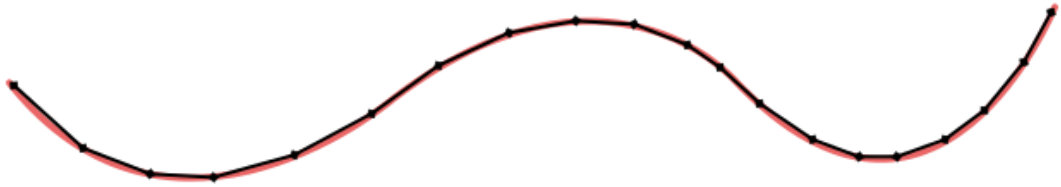


Figure 4.4: Visualization of euclidean distances between points computed with bézier curve parametrization. This is an arbitrary line feature used solely for demonstration purposes.

$k = [0, \cdots, N-1]$ 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 this distance. We choose index which returns the coordinates for the exact fraction length and if that is not the case the index with the next smaller length. The coordinates of the point either exactly or approximately computed for the given fraction length of the curve is used as the sampled point.

3D Mesh Projection of Sampled Points

Having implemented arc length parametrization it is possible to draw an arbitrary amount of samples $x \in \mathbb{R}^2$ from the line features. They are then defined as a set of points $S \subset \mathbb{R}^2$. Our goal is, however, to have these additional landmarks describing the features on the mesh of a face itself and not a 2-dimensional snapshot thereof. Because we have no information on the depth of the line features, we have to project the sampled points of each line feature onto a face mesh in order to obtain their approximate 3D representation. ***Introduce camera model with schema here?*** In the camera model the image is located on the viewing plane or viewport opposite of the focal point. (computes 3D direction of 2D sample point) The direction of the 3D representation of a point on a curve is given by the (normalized) vector defining the position of the point on the viewing plane from the perspective of the focal point of the camera. Given: diskrete mesh, how to choose point? We now a 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 a corresponding vertex, we save all the distances of mesh

vertices in a list which have a similarity measure that is higher than a specified threshold. 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.

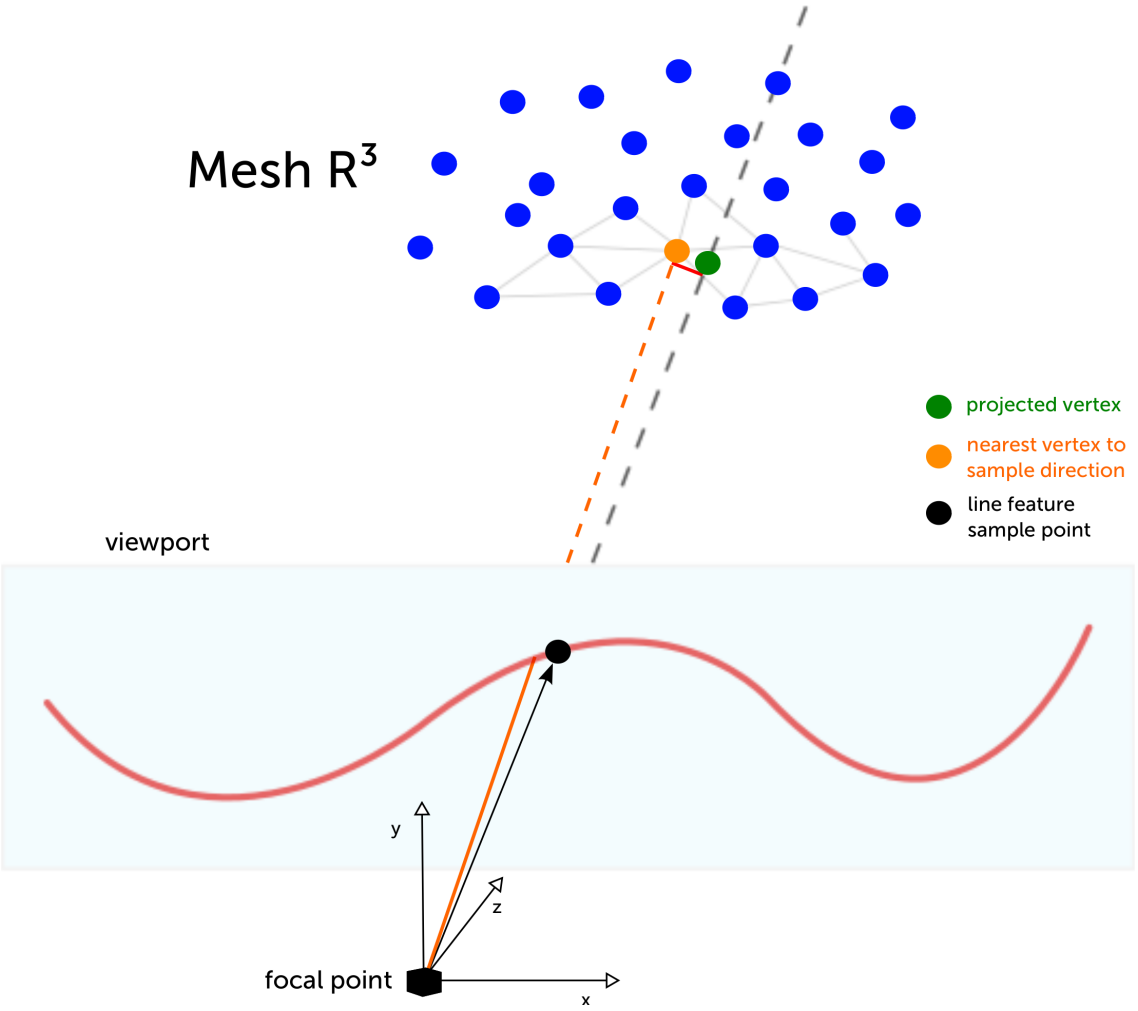


Figure 4.5: shows the projection of a sample point - on a 2D line feature - from the viewport onto a triangulated 3D mesh. The distance vector of the vertex with the most similar direction vector (orange) is projected (red) onto the direction of the sample point (black arrow), resulting in the projected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

Problems & Inacurracies The face meshs 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 meshs - it lacks generality and generality is exactly the basis for feasible and reproducable registration results.

Preparing the Template Mesh In the next step, the line features also had to be marked on the template mesh. As the template mesh we use the mean mesh of the 3DMM of the Graphics and Computer Vision Group at the University of Basel. In order to mark the features, the mean mesh first had to be rendered with three different camera callibrations. The 2D line features where then projected back onto the mesh using these callibration parameters. The mean mesh already had 60 feature points clicked manually, many of which were not added to the face scans. After the projetion, we have added samples from the line features to our set of landmarks. We therefore have provided the registration algorithm with additional prior information to define the deformation field by and to produce a better fitting result.

4.3 Rigid Mesh Alignment

Before we can start with initializing Gaussian Models, we first have to ensure that 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 ***What kind of RT?*** rotation + translation aligning the meshs according to their landmarks. We use the set of landmark whose identifiers are present in both meshs. ***The face scans have to be clipped at the neck and around the ears where the scanner has left artifacts.*** The computed transformation is applied to all vertices of the respective face scan. The mean face was broader in shape than the scan and was

perfectly coated in texture for the simple reason that hours of manual labour have been invested to render this important piece of data a perfect reference. ***Show 3 images of overlap of mean with different scans*** The aligned meshs serve as the starting point for the registration.

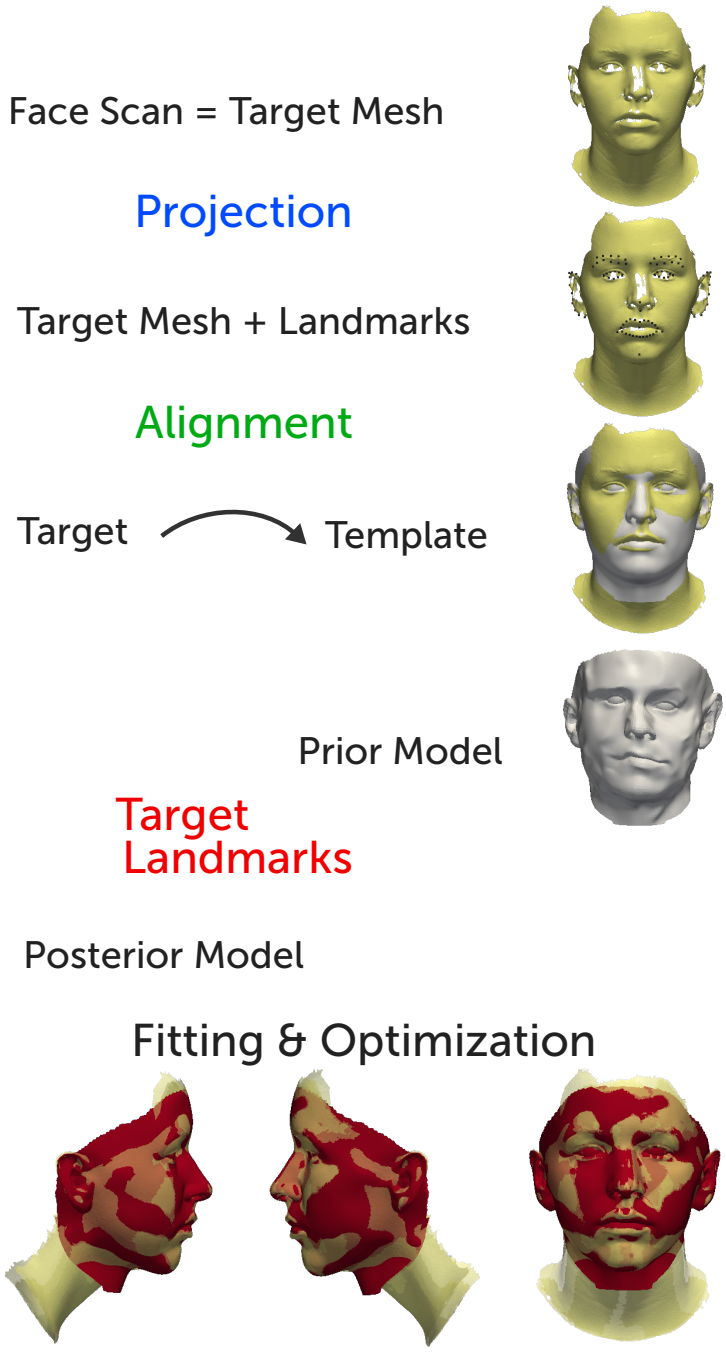


Figure 4.6: Fitting pipeline: blablabla

We are now set for the actual registration involving the Prior and Posterior Gaussian

Models and the Fitting. For the definition of the Gaussian Process distributions we use the software framework statismo developed at the Computer Science Department of the University of Basel. 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. Furthermore we use the software package gpfitting for the actual fitting. ***gpfitting*** is also statismo

4.4 Prior Model

imperative: build prior faces resulting from possible deformations of template The Gaussian Process Prior distribution of the template mesh is represented as an object and the parameters are saved on to disk. This allows for samples to be drawn from this representation of the Gaussian Process Prior Model. These samples are 3D face mesh that are the result of the deformations generated by the Gaussian Process Prior and then added to the template mesh. They are examples for possible deformations defined solely on the ground of the covariance of the template mesh points. ***show some examples here, or already in chapter 3***

4.5 Posterior Model

imperative: build posterior model from combination of the prior model and the target landmarks. ***Landmarks are added to the distribution as additional points and co-define the deformation for the mesh vertices*** Build Model from mean and target landmarks. This time we incorporate the mapping information of the landmarks and get a model where the We get valid face like meshs for different deformations, which are fixed at the target landmarks.

4.6 Fitting

Perform optimization Show some fits?
What are we optimizing? L2??? –¿ not robust to outliers, protruding regions.

4.7 Robust Loss Functions

Optimizing the loss function? After the alignment of template and target mesh, the template protrudes over the target on the upper side of the head and the side of the neck. ***show an image with template and target on top of each other*** Performing optimization as described in **??** using a simple Mean Square Error(MSE) as a distance measure between the template and target mesh penalizes the portruding regions of the template with a strong gradient towards the rims of the template and therefore causes strong distortions. ***show image of failed fitting, next to target***

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

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

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

Huber

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

$$\psi(x) = \begin{cases} x & \text{if} \\ ksgn(x) & \text{if} \end{cases} \tag{4.4b}$$

Tukey

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

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

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

4.8 Varying the Variances