People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of the Requirements for the Degree of

MASTER

In **Electrical and Electronic Engineering**

Option: **Computer Engineering**

Title:

3D Statistical Shape Modeling

Presented by:

- **Omari Sabiha**
- **Soual Imene**

Supervisor:

Dr. D. CHERIFI

Registration Number:……/2016

Dedication

We dedicate this work to our dear parents, our beloved ones, precious families, Wise teachers and faithful friends.

Acknowledgment

Praise be to Allah for all of his blessings in our education.

Deep gratitude and thanks to our supervisor Dr. Dalila Cherifi for her trust and support that she provided to us through the realization of the project. Warm regards to all of the IGEE teachers

Our special thanks to Mr. Ghazi Bouabene and Mr. Marcel Lüthi from the Department of Mathematics and Computer Science of the Basel University (Switzerland); for their guidance and giving us the access of their data base.

We also offer our regards and blessings to all of those who supported us in any respect during the completion of the project.

Abstract

Statistical shape models (SSMs) have been firmly established as a robust tool for segmentation of images. Widespread utilization of three-dimensional models appeared only in recent years, primarily made possible by breakthroughs in automatic detection of shape correspondences; while 2D models have been in use since the early 1990s.

The objective of this project is to build a 3D statistical shape modeling for a given data; the implemented process goes through those basic steps, first collect the given data then apply the alignment algorithm based on the ICP (iterative closest point) method which in turn relies on Procrustes analysis result as a starting point, next we apply fitting algorithm which is also based on ICP. Finally we obtain the model using PCA (principle component analysis).

To achieve this work, we have implemented the above process on two different shape models, one tested with the Basel Face Model (BSF) and the other is the femur model data samples from the SICAS (Swiss Institute for Computer Assissted Surgery) Medical Image Repository which is used by the Basel University (Switzerland) for both samples, where these models allow the generation and the exploration of the possible shape variation.

Tables of contents

General Introduction

In the last two decades, model-based segmentation approaches have been established as one of the most successful methods for image analysis. By matching a model which contains information about the expected shape and appearance of the structure of interest to new images. Information about common variations thus has to be included in the model. A straight-forward approach to gather this information is to examine a number of training shapes by statistical means, leading to statistical shape models (SSMs). Shape is usually defined as that quality of a configuration of points which is invariant under some transformation. In two or three dimensions we usually consider the Similarity transformation (translation, rotation and scaling). The shape of an object is not changed when it is moved, rotated or scaled.

To obtain interesting information about the shapes represented by the model we apply mathematical concept which is the Gaussian Processes, GP provide us with a mathematically elegant way of modeling shape deformations. As shape modeling is an application-oriented task, we are not primarily interested in mathematical elegance, but rather in obtaining practical algorithms.

The objective of this work is to derive models which allow us to both analyze new shapes, and to synthesize shapes similar to those in a training set. The training set typically comes from hand annotation of a set of training images, though automatic land marking systems are being developed. By analyzing the variations in a shape over the training set, a model is built which can mimic this variation; this also greatly simplifies the algorithms and tools used in practical shape modeling applications.

This report consists mainly of 3 chapters. The first set the mathematical tools that is widely employed for the presentation and the estimation of the model parameters of statistical shape model. Whereas, the second defines the algorithms and methods used to achieve our objective. The last chapter presents the result of the experimental part that realizes the previously discussed concepts, finally, a general conclusion is provided.

1.1 Basic notions of shape modeling

The most fundamental concept in shape modeling is the notion of a shape; the figure below shows three different geometrical objects (see Figure 1.1). Refer to each of them as a square even though they differ in size, position, and rotation. This is reflected by the classical definition of a shape. We will focus mainly about anatomical shapes. For anatomical shapes, the shape is defined as all geometric information that remains when location and rotational effects are filtered out from an object (see Figure 1.2). We cannot give a mathematical formula to define the family of anatomical shapes but we can use mathematics to characterize what constitutes, for example a hand shape.

Figure 1. 1: *Classical definition.*

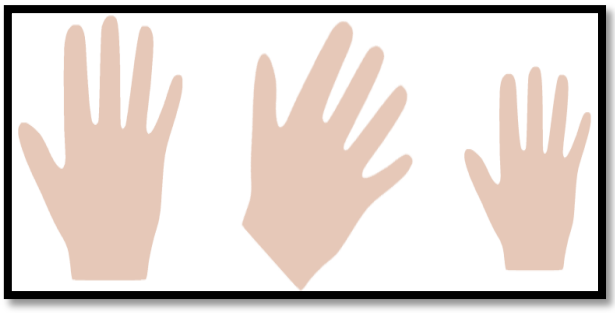

Figure 1. 2: *Anatomical definition [1].*

The idea is that we start with example shapes from the shape family, and then we use statistics to model the characteristics (see Figure 1.3). The models that built in the way, they can, on one hand, tell for any shape how likely it is that the shape is part of the family and they can generate new shapes from the same family.

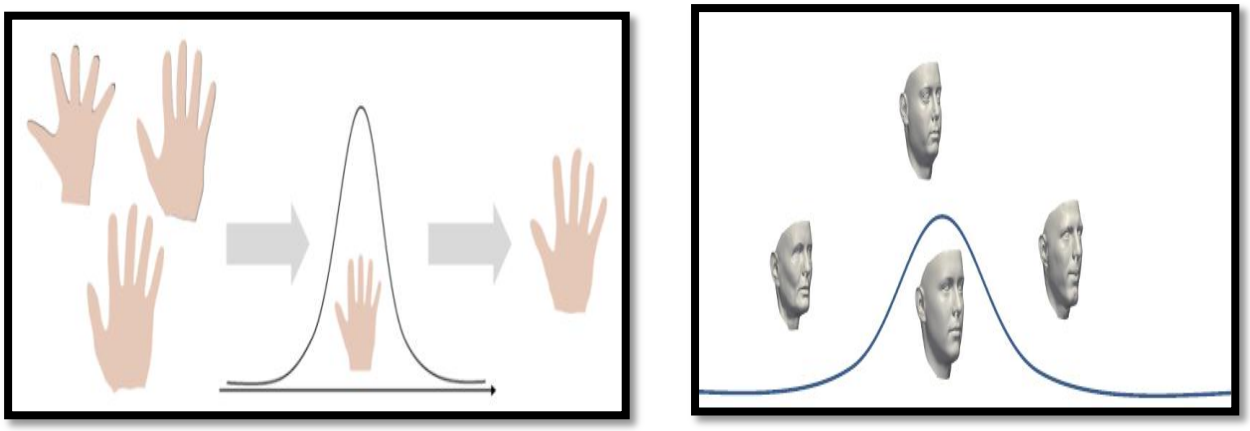

Figure 1. 3: *Statistical shapes model [2].*

The most classical representation of shapes is the landmark points. Landmark points are points that have an anatomical meaning for example the tip of the thumb (Figure 1.4). These

points can be easily found in all the shapes. The problem with landmark points is that there are only a few of them. And if we want to have a good faithful shape representation, we need many more points.

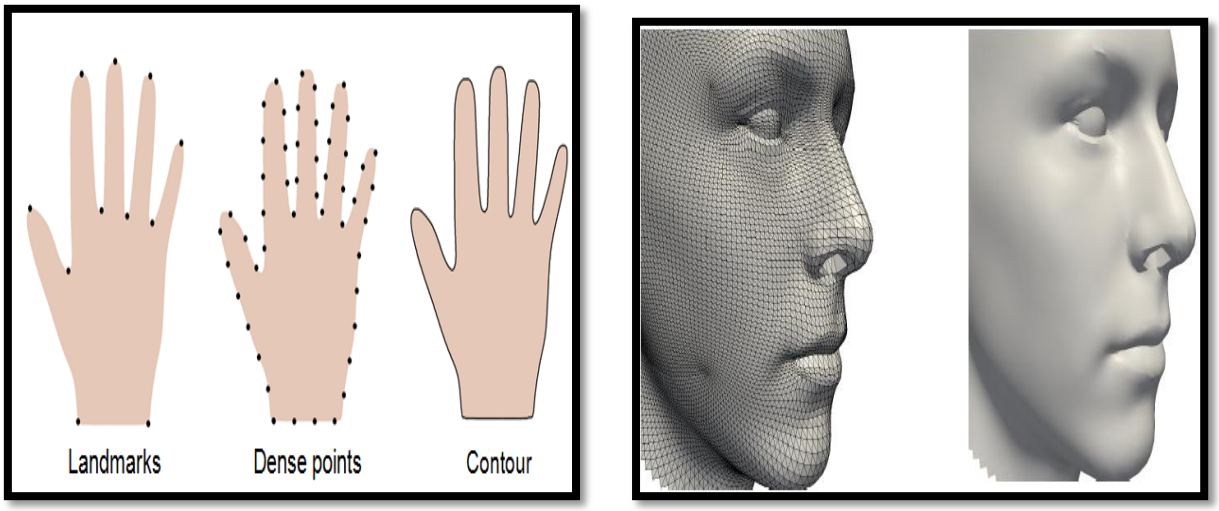

Figure 1. 4: *Shape representation: a) Hand shape b) Face shape [2].*

1.2 The normal distribution

The main assumption underlying the shape model is that the shape variations can be modeled using a normal distribution. In this part, we summaries the main properties of normal distributions and show how they manifest themselves in shape modeling.

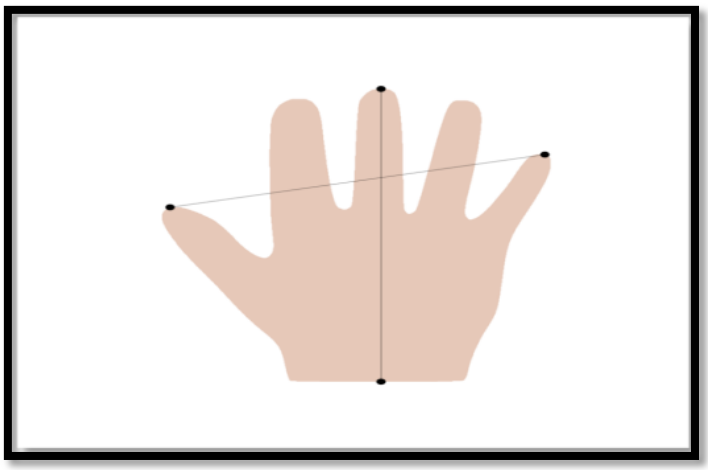

Figure 1. 5: *Measurements for the length and the span of a hand shape [2].*

The above example is a simple model of a hand shape which is described by only two parameters: the length and the span (see Fig 1.5). We start by modeling the length where the main assumption is that the length follows a univariate normal distribution [1].

1.2.1 The univariate normal distribution

The normal distribution is completely defined by two parameters: the mean μ and variance σ^2. Its probability density function is given by the following Equation 1.1 [2]:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (1.1)$$

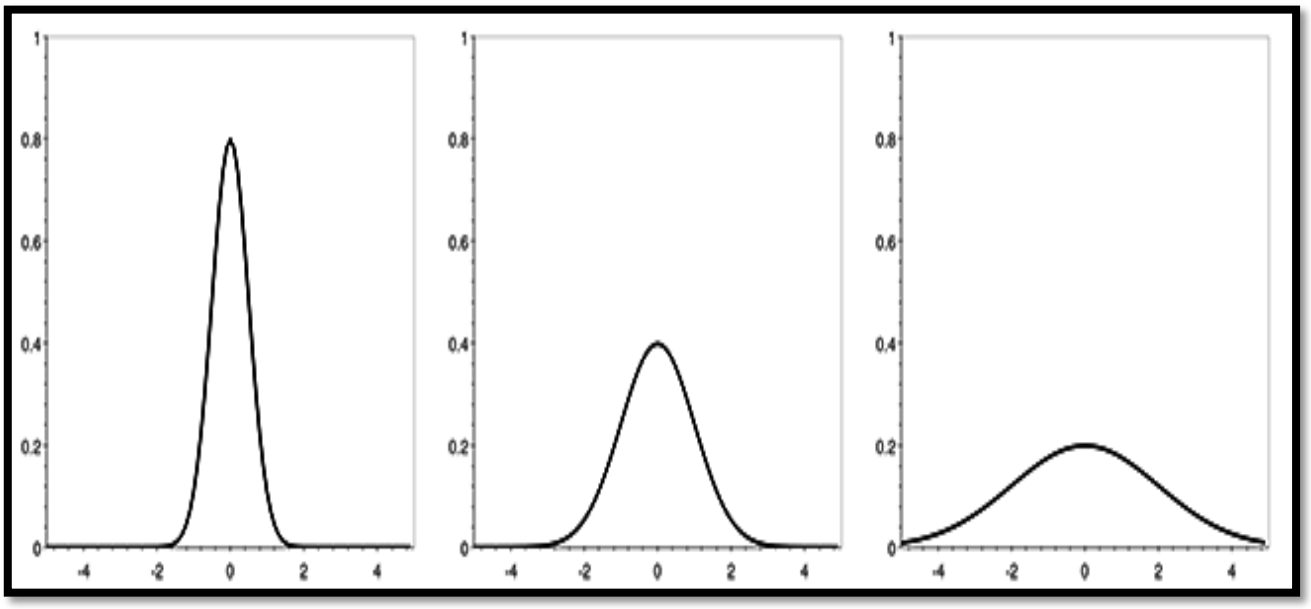

Figure 1. 6: *The density function of the univariate normal distribution: a) μ=0 and σ^2=0.25, b) σ^2=1, c) σ^2=2 [3].*

The normal distribution is symmetric around the mean and it assigns a non-zero probability everywhere (see Figure 1.6). The variance σ^2 determines how far the values are spread around the mean. The further away a value lies from the mean the less likely it becomes. The probability of observing a value that is outside of an interval of 3σ is less than 0.01.The normal model captures assumption about the length of a hand rather well. There is an average around which most values are distributed. The possibility of observing unusually long or short hands remains, but the more the observed length deviates from the mean, the less likely we are to ever see such a hand. To define the model, we need to choose values for the parameters μ and σ^2. A reasonable approach is to take a ruler and measure the hand of a number of people. We obtain a set of values $\{l_1,\ldots,l_m\}$, which we can use to estimate these parameters using the well-known formulas for the sample mean and sample variance in following equations [3]:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} l_i \quad (1.2)$$

$$\sigma^2 = \frac{1}{m-1}\sum_{i=1}^{m}(l_i - \mu)^2 \quad (1.3)$$

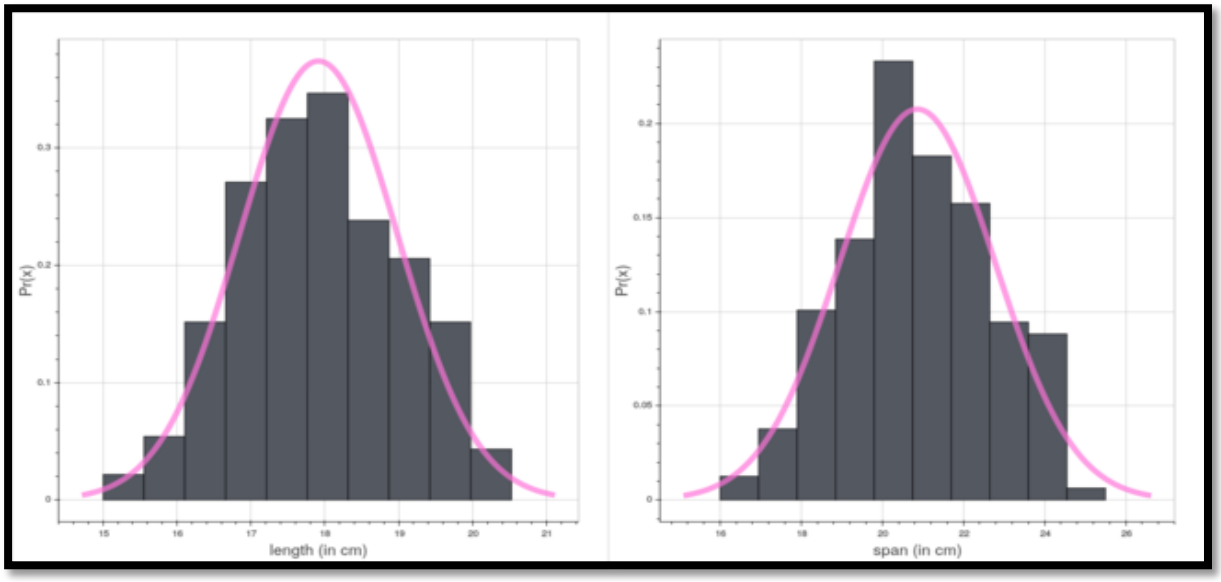

Figure 1. 7: *Histogram for the span and length obtained from 169 measurements [2].*

1.2.2 The bivariate normal distribution

We can obtain a more interesting shape model by modeling the span s and length l jointly. The assumption is that it follows a bivariate normal distribution:

$$(s, l) \sim N\left(\begin{pmatrix}\mu_s \\ \mu_l\end{pmatrix}, \begin{pmatrix}\sigma_{ss} & \sigma_{sl} \\ \sigma_{ls} & \sigma_{ll}\end{pmatrix}\right) \quad (1.4)$$

To define the distribution, we specify the mean and variance for the span μ_s and σ_{ss} and the length μ_l and σ_{ll} as well as the covariance σ_{ls}=σ_{sl}. The covariance describes the coupling between the variables. If σ_{ls} = 0, it means the two variables are independent, otherwise we know that the values are correlated. In the example of the hand shape, the span and the length of the hand are correlated, as both are related to the size of the hand.

1.2.2.1 The marginal and conditional distributions

Two distributions that can be derived from the bivariate normal distribution have a very important role. First is the marginal distribution, which gives us the distribution for s (or l) separately. The marginal distribution for s is the distribution we obtain if we do not know anything about the value of l. It is simply a univariate normal defined if we drop all variables that are not related to s, i.e. $s \sim N(\mu_s,\sigma_s)$. The second important distribution is

the conditional distribution *s (or l).* It can be shown that also this conditional distribution is a univariate normal distribution. It models the span *s* given that we know the length *l*. As the assumption is that span and length are correlated, the variance in the conditional distribution should be smaller than in the marginal distribution, where there is no assumption about the length.

1.2.4 The multivariate normal distribution

Normal models can be defined for any finite number of variables using the multivariate normal distribution N (μ, Σ). It is specified by a mean vector $\mu \in R^n$ and a covariance matrix $\Sigma \in R^{n \times n}$:

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \dots & \Sigma_n \\ \vdots & \vdots & \vdots \\ \Sigma_{1n} & \dots & \Sigma_{nn} \end{pmatrix} \right) \tag{1.5}$$

The diagonal entry Σ_{ii} defines the variance of the variable x_i, whereas $\Sigma_{ij,}\ i \neq j$ defines the covariance between x_i and x_j. A valid covariance matrix is required to be symmetric and positive semi-definite. The marginal and conditional distributions can be defined also in this more general case. They are both multivariate normal distributions themselves whose parameters can be computed efficiently using closed-form formulas.

1.3 The marginalization property

The marginalization property of a Gaussian Process is the property that makes it, not only a nice theoretical concept but actually a very practical tool that we can use for shape modeling. The assumption state that when have two sets of random variable: X=(x_1,….,x_n) and Y=(y_1,…..,y_2) .We model them using a joint multivariate normal distribution:

$$\mathrm{p(X,Y)} = \left(\begin{pmatrix} \mu_X \\ \mu_Y \end{pmatrix}, \begin{pmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{pmatrix} \right) \tag{1. 6}$$

The marginal distribution is given by:

$$\mathrm{p(X)} = \int p(X,Y)\, dY \tag{1.7}$$

$$\mathrm{p(X)} = \mathrm{N}\,(\mu_X\,,\ \textstyle\sum \mathrm{XX}) \tag{1.8}$$

This property can even be used to formally define a Gaussian Process, because a Gaussian Process is just a generalization of a multivariate normal distribution. We can formally Gaussian Process as probability distribution given below:

$$p(u) = GP(\mu, k) \tag{1.9}$$

Over functions $u: x \rightarrow R^d$ such that every finite restriction to function values: $u_X = (u(x_1), \ldots, u(x_n))$ is a multivariate normal distribution described as follow:

$$p(u_X) = N(\mu_X, k_{XX}) \tag{1.10}$$

1.4 Sampling

Since it's relatively simple to work with multivariate normal distribution and considering the familiarity with this type of distribution, interesting things can be done with it. So for example, one can easily sample from a multivariate normal distribution where shape variations can be visualized. Sampling from a probability distribution is the task of generating concrete values (samples) of a random variable, according to the probabilities defined by the distribution. With this process, samples of high probability are more likely to be generated than those with a low probability. Note that this use of the term sampling should not be confused with the process of sampling a continuous function, where the goal is to obtain a discrete representation of the function. Sampling shapes is maybe the simplest possible use of shape models. Nevertheless it has many practical applications. Sampling shapes might be a useful way to learn about the shape variations of an anatomical structure.

1.4.1 Sampling from a multivariate normal distribution

Unlike for many other distributions, sampling from a multivariate normal distribution is straight-forward. We set $x \sim N(\mu, K)$ to be a multivariate normal distribution with mean $\mu \in R^n$ and covariance matrix $K \in R^{n \times n}$; the general procedure is as follows:

1. Generate n random numbers $c_1, \ldots, c_n$ where $c_i \sim N(0,1)$ by calling multiple times a scalar Gaussian random number generator (which is available in most programming languages). Define the vector $c = (c_1, \ldots, c_n)^T$.
2. Compute $K = LL^T$ using the Cholesky decomposition. Here L is a lower triangular matrix, which results from this decomposition.
3. Compute the sample $x = \mu + L_c$, which has the desired distribution.

1.5 Finite rank representations of a Gaussian Process

In this part we will discuss an alternative representation of Gaussian Processes using the Karhunen-Loève expansion (KL). The Gaussian Processes give us both a nice theoretical

model to work with, however, there can still be a problem, and the problem can happen when we want to work with really high resolution meshes. The Karhunen-Loève expansion in the discrete case allows writing a Gaussian Process model:

u ~ GP (μ , k) As $u = \mu + \sum_{i=1}^{\infty} \alpha_i \sqrt{\lambda_i}\phi_i, \quad \alpha_i \sim N(0,1)$ (1. 11)

Φ_i is the eigenfunctions with associated eigenvalues λ_i of the linear operator that is associated with the covariance function k. If we consider deformation fields that are defined on a discrete finite domain we can represent each discretized function u as a vector $\vec{u} = (u_1, \dots, u_n)^T$ and define a distribution over u as:

$$\vec{u} \sim \mathrm{N}(\vec{\mu}, \mathrm{K}) \tag{1.12}$$

K is a symmetric, positive semi-definite matrix and hence admits an eigen decomposition [4]:

$$\mathrm{K} = \Phi \mathrm{D} \Phi^{\mathrm{T}} = \begin{pmatrix} \vdots & & \vdots \\ \varphi_1 & \dots & \varphi_n \\ \vdots & & \vdots \end{pmatrix} \begin{pmatrix} d_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & d_n \end{pmatrix} \begin{pmatrix} \vdots & & \vdots \\ \varphi_1 & \dots & \varphi_n \\ \vdots & & \vdots \end{pmatrix}^T \tag{1.13}$$

Where φ_i refers to the i-th column of Φ and represents the i-th eigenvector of K. The value d_i is the corresponding eigenvalue. This decomposition can for example, be computed using a Singular Value Decomposition (SVD) (see Appendix C). The expansion can be written in terms of the eigenpairs d_i , $\vec{\varphi}_i$ as the following:

$$\vec{u} = \vec{\mu} + \sum_{i=1}^{\infty} \sqrt{d_i}\ \vec{\varphi}_i \alpha_i \qquad \alpha_i \sim \mathrm{N}(0, 1) \tag{1.14}$$

Using the Karhunen-Loève expansion (KL) we can obtain algorithms that can deal with shapes that are represented using millions of points. Furthermore, the representation gives a simple and efficient means for computing the probability of any shape in the family.

1.6 The Principal Component Analysis

A very popular method in shape modeling is Principal Component Analysis (PCA). PCA is closely related to the Karhunen-Loève (KL) expansion. It can be seen as the special case where the Gaussian Process is only defined on a discrete domain (i.e. it models discrete functions) and the covariance function is the sample covariance estimated from a set of example data. In this part, the connection between the PCA and the KL expansion is shown, and a discussion about how PCA can be used to visualize and explore the shape variations in a systematic way is made. PCA is a fundamental tool in shape modeling. It is essentially the

KL expansion for a discrete representation of the data, with the additional assumption that the covariance matrix is estimated from example datasets. More precisely, assume that we are given a set of discrete deformation fields $u_1,\ldots,u_m$, which we also represent as vectors $(\overrightarrow{u_1},\ldots,\vec{u}_m)$ $\vec{u}_i \in R^n$. Note that, as the vector $\vec{u}_i$ represents a full deformation field, *n* is usually quite large. As mentioned before, PCA assumes that the covariance function Σ is estimated from these examples:

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(\vec{u}_i - \bar{u})(\vec{u}_i - \bar{u})^T = \frac{1}{m}XX^T \tag{1.15}$$

Where we defined the data matrix X as $X = (\vec{u}_1 - \bar{u}, \ldots, \vec{u}_m - \bar{u}) \in R^{n\times m}$, and $\bar{u}$ is the sample mean where:

$$\bar{u} = \frac{1}{m}\sum_{i=1}^{m}\vec{u}_i \tag{1.16}$$

We note that in this case, the rank of Σ is at most *m*, which is the number of examples. This has two consequences: first, it allows the computation of the decomposition efficiently, by performing an SVD of the much smaller data matrix X. Second, Σ has in this case only *m* non-zero eigenvalues. The expansion reduces to [4]:

$$\vec{u} = \bar{u} + \sum_{i=1}^{m}\sqrt{\lambda_i}\ \vec{\varphi}_i\alpha_i \qquad \alpha_i \in N(0,1) \tag{1.17}$$

This implies that any deformation $\vec{u}$ can be specified completely by a coefficient vector $\vec{\alpha} \in R^m$.

1.7 Summary

This chapter sets the main mathematic tools that are the basic properties used in the algorithms of building shape model starting by the normal distribution and how it manifest in shape modeling.

Chapter 2: Theoretical Concepts of Building Statistical Shape Models

2.1 Modeling shape deformation

The basic question in shape modeling is how to model the shape variations within a shape family. The answer of this question is by means of the normal distribution, which is introduced in this part. Modeling shape deformation of the hand shape example is that: the original hand, together with the deformation vectors defines the hand shape (see figure 2.1). Where instead of modeling different shapes directly, the shape changes as deformations from a given shape are modeled. Starting from the same reference shape, two different shapes are defined by inducing two different types of deformation fields.

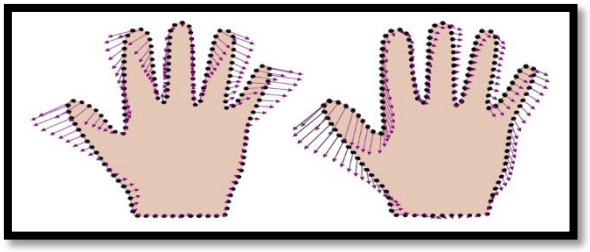

Figure 2. 1: *Hand shape*

To define this idea more formally; fix an arbitrary shape and call it the reference shape.

$\boldsymbol{\Gamma}_{\mathbf{R}}=\{x \mid x \in R^2\}$, $\boldsymbol{\Gamma}_{\mathbf{R}}$ is just a set of points defining a reference shape; to describe a shape variation, define a vector field $\mathbf{U}: \Gamma_{\boldsymbol{R}} \longrightarrow \mathbf{R^2}$ modeling the deformation. To model this shape deformation we start by specifying the deformation U (x) at one point x (tip of the thumb as showing in Figure 2.2).

$$u(x) = \begin{pmatrix} u_1(x) \\ u_2(x) \end{pmatrix} \tag{2. 1}$$

$$\sim \; N\left(\begin{pmatrix} \bar{u}_1(x) \\ \bar{u}_2(x) \end{pmatrix}, \begin{pmatrix} \sum_{11}(x) & \sum_{12}(x) \\ \sum_{21}(x) & \sum_{22}(x) \end{pmatrix}\right) \tag{2. 2}$$

Make the assumption that the deformation follows a multivariate normal distribution. To specify the model define the mean deformation and the covariance matrix that defines how much the position of the tip of the thumb can vary.

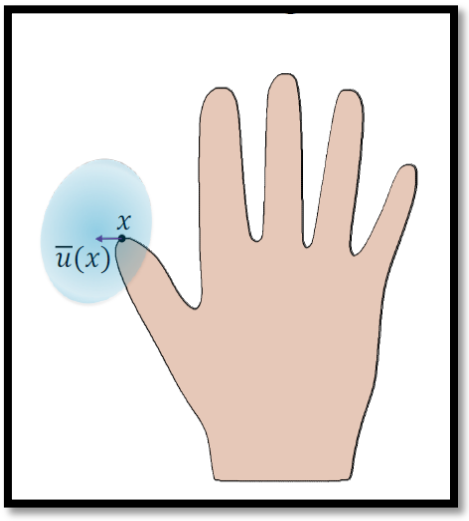

Figure 2. 2: *Tip of the thumb [1]*

The model used to model shape deformation is called a Gaussian Process model. It is a model that simply specifies a multivariate normal. It is a bit more general than a standard multivariate normal distribution. As it turned out, this is also works if we defined a mean and covariance function on an infinite number of points. The Gaussian Process $u\sim\ (\mu,\ k)$ is defined by a mean function μ: $\Gamma_R \rightarrow \mathbb{R}$ and a covariance function k: $\Gamma_R \times \Gamma_R \rightarrow \mathbb{R}^{2\times2}$ on arbitrary (even infinite) sets Γ_R.

To summarize, a Gaussian Process is an extension of the multivariate normal distribution. While we think of a multivariate normal distribution as giving a distribution over vectors, we think of a Gaussian Process as providing a distribution over functions. Gaussian Process has been used for decades in statistics and it is also a very popular tool in machine learning. This has the huge advantage that there are many well established and useful concepts that we can just use for shape modeling.

2.2 Gaussian Processes: from random vectors to random functions

Gaussian Processes generalize the concept of multivariate normal distributions. Whereas the multivariate normal distribution models random vectors, Gaussian Processes allow defining distributions over functions and deformation fields. In the case of discrete functions, a Gaussian Process is simply a different interpretation of a multivariate normal distribution. The goal of this part is to discuss this point in more detail and to provide the intuition for the general case.

2.2.1 Representing discrete scalar-valued functions using a multivariate normal distribution

We have Ω is an arbitrary continuous domain and let $\Omega=\{x_1,\ldots,x_N\}\subset \Omega$ be a discretization of that domain. We consider a discrete function $\tilde{f}: \tilde{\Omega} \to \mathbb{R}$, the discrete function $\tilde{f}$ can be represented by a vector:

$$\vec{f} := \left(\tilde{f}(x_1), \ldots, \tilde{f}(x_N)\right)^T \in \mathbb{R}^N \tag{2.3}$$

Vice versa, the vector $\vec{f}$ completely defines the function $\tilde{f}$ as we can define $\tilde{f}(x_i) := \vec{f}_i$ Figure 2.3 illustrate this situation:

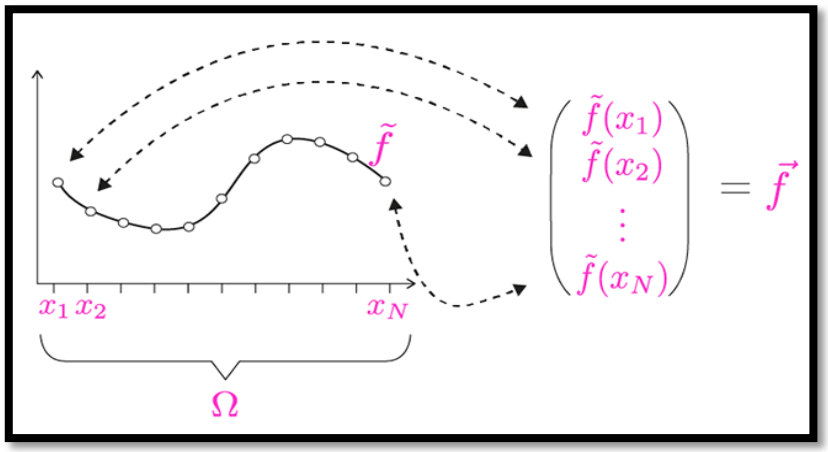

***Figure 2. 3**: Discrete function can be represented by a vector and vice versa [1].*

The main goal is to model distributions over functions. We can model $\vec{f}$ using a multivariate normal distribution i.e. $\vec{f} \sim N(\vec{\mu}, \Sigma)$ This in turn defines a distribution over the discrete functions $\tilde{f}$ for example; we can draw a random function, by drawing a sample $\vec{f}$ from the normal distribution, and use it to define the corresponding discrete function $\tilde{f}$.

2.2.2 Representing discrete vector-valued functions using a multivariate normal distribution

In shape modeling, one is interested in modeling vector fields and not scalar-valued functions. Fortunately, this is not much more complicated. Let $\tilde{u}: \tilde{\Omega} \to \mathbb{R}^2$ be a discretely defined vector field. The difference to the previous case is that each component is now a vector, i.e. $\vec{u} := \left(\tilde{u}_1(x_i), \tilde{u}_2(x_i)\right)^T$ as before, one can identify the vector field $\tilde{u}$ with a vector as the following equation:

$$\vec{u} := \left(\tilde{u}_1(x_1), \tilde{u}_2(x_2), \ldots, \tilde{u}_1(x_N), \tilde{u}_2(x_N)\right)^T \in \mathbb{R}^{2N} \tag{2.4}$$

Modeling $\vec{u}$ using a multivariate normal distribution results a distribution over the (discrete) vector field $\vec{u}$.

2.2.3 from vectors to function

Gaussian Processes make it possible to model a distribution over functions without choosing a discretization up front. The intuition is the following: to model a 2D vector field, define a Gaussian Process $u \sim GP(\mu, k)$ with mean function $\mu : \Omega \to \mathbb{R}^2$ and covariance function $k: \Omega \times \Omega \to \mathbb{R}^{2\times 2}$. these functions define the mean deformation $\mu(x)$ for all the points $x \in \Omega$ and the covariance $k(x, x')$ between the deformations for any pair of points x and x′. This allows defining normal models for functions that are defined using an arbitrarily fine discretization. Since, in practical applications (i.e. for computer implementations), we always work with a discretization, this is already an almost perfect situation. It allows choosing for any application the discretization that yields the desired accuracy.

2.3 Constructing kernels for shape modeling

The covariance function should be positive semi-definite (p.s.d) to define a valid Gaussian Process model.

Any real $n \times n$ matrix K that satisfies $v\,K\,v^T \geq 0$ for all vectors $v \in \mathbb{R}$; is called a positive semi-definite matrix. A kernel $k : X \times X \to \mathbb{R}^{d\times d}$ is called positive semi-definite, if it gives rise to a positive-semi-definite kernel matrix K with $K_{i,j} = k(x_i, x_j)$, $i, j = 1,\ldots, n$ for any choice of n and $X = (x_1, \ldots, x_n)$. Once we know that a kernel is positive semi-definite, it can be used to construct new positive semi-definite kernels using a set of rules for combining kernels. k_1, $k_2 : \Omega\times\Omega\to R^{d\times d}$ are two positive semi-definite kernels and $f: \Omega\to R^d$ a vector-valued function. Then the following rules can be used to generate new positive semi-definite kernels $k:\Omega\times\Omega\to R^{d\times d}$ [5]

1. $k(x,x')=k_1(x,x')+k_2(x,x')$.
2. $k(x,x')=\alpha k_1(x,x')$, $\alpha\in R^+$.
3. $k(x,x')=k_1(x,x')\odot k_2(x,x')$.
4. $k(x,x')=f(x)f(x')^T$.
5. $k(x,x')=B^T k(x,x')B$, $B\in R^{d\times r}$.

2.3.1 Other kernels and kernel combinations

Many other kernels exist beside the Gaussian kernel. These kernels attribute different similarities to the same domain points and therefore give rise to differently behaving deformations. The power of the Gaussian process framework is that it is completely agnostic to the kernel function being used. We can strongly modify the behavior of the deformation by simply plugging a different kernel function.

2.4 Superimposing shapes (Alignment)

When given shapes in correspondence, computing a shape model is straight-forward, however before building the model, we have to correct the pose of the shapes. This is usually done using a method called Procrustes alignment.

2.4.1 Procrustes alignment of two point sets

If two shapes Γ_R and Γ_T are in correspondence, this means that for every point $x \in \Gamma_R$, we can identify the corresponding point $y \in \Gamma_T$; consider two finite sets of points on the contour of the shape are as following

$$X=\{x_1,\ldots,x_n\} \subset \Gamma_R \tag{2.5}$$

And the corresponding points

$$Y=\{y_1,\ldots,y_n\} \subset \Gamma_T \tag{2.6}$$

To eliminate the effect of rotation and translation, we can minimize the following optimization problem using the equation below (see appendix B):

$$(t^*, R^*) = \arg\min \sum_{i=1}^{n} \| x_i - R(y_i - t) \|^2 \tag{2.7}$$

$$t \in R^2, R \in R^{2\times 2}\ s.t\ R^T R = I, \det(R) = 1$$

To obtain the translation t^* and rotation R, this can be used to superimpose the two shapes. It turns out that this problem has a closed-form solution both in 2D and 3D, which can be efficiently computed using singular value decomposition [6].

2.4.2 Generalized Procrustes alignment

Given a set of shapes $\Gamma^{(1)},\ldots, \Gamma^{(m)}$, one of the shapes is selected as a reference shape, suppose that $\Gamma_R=\Gamma^{(1)}$ where the rest of all shapes are aligned to this reference using the above procedure. This choice of reference may seem a bit arbitrary. However, a more principled way

of doing this, which is usually referred to as generalized Procrustes alignment. The idea is that, since we have correspondence between all the shapes, we can compute a mean shape defined by $\Gamma_\mu = \{\mu_1,\ldots,\mu_n\}$ with

$$\mu_i = \frac{1}{m}\sum_{k=1}^{m} x_i^{(k)} \quad , \mathrm{i} = 1, \ldots\ldots, \mathrm{n} \tag{2.8}$$

The generalized Procrustes alignment is obtained using the following procedure [5]:

- Choose an arbitrary shape as the reference shape Γ_R.
- Align all shapes $\Gamma^{(1)},\ldots,\Gamma^{(m)}$to Γ_R using the procedure above.
- Compute the mean shape Γ_μ for the set of aligned shapes.
- Iterate using the mean shape as the new reference shape.

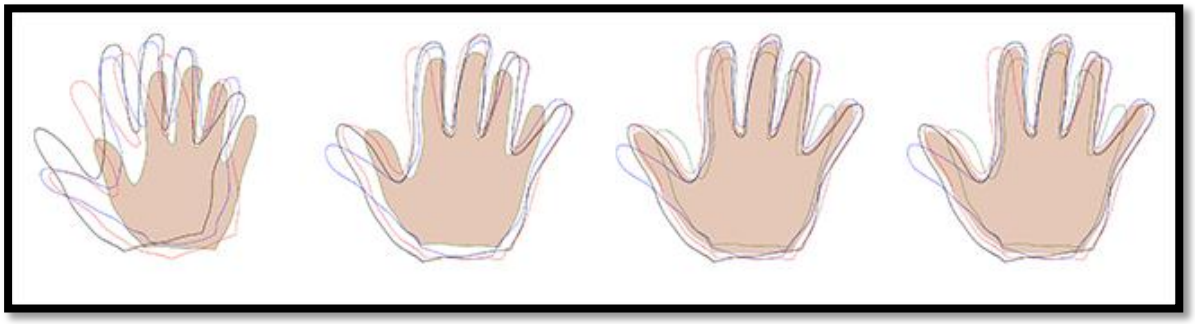

Figure 2. 4: *An illustration of the generalized Procrustes alignment[1]*

This procedure is illustrated in Figure 2.3, where the solid shape indicates the current reference. The left most image shows the initial situation, where an arbitrary shape is chosen as a reference. The shapes are aligned to the reference (2nd image) and the mean is recomputed (3rd image). The rightmost image shows the situation after an additional alignment step. We notice that already in this step only minor adjustments are made. Indeed, the procedure usually converges within a few iterations

2.4.3 Iterative Closest Points

In order to use the ICP in combination with Procrustes analysis for finding the best rigid transform aligning one mesh to another. The method proceeded as follows:

1. Suggest candidate correspondences between the mesh to be aligned and the target one, by attributing the closest point on the target mesh as a candidate.
2. Solve for the best rigid transform between the moving mesh and the target mesh using Procrustes analysis.

3. Transform the moving mesh using the retrieved transform and loop to step 1 if the result is not aligned with the target (or if we didn't reach the limit number of iterations).

2.5 Model fitting

The model fitting is an important first step in analyzing new shapes by comparing them with the shape model. We will show that the problem of fitting a model to a new shape arises also in the process of model building itself, namely when correspondence has to be established between the example shapes. In this context, model fitting is known as registration [6].

2.5.1 Model building revisited

The key idea in statistical shape modeling is to estimate a distribution for the shape deformations from a set of example shapes $\Gamma_1,\ldots,\Gamma_n$. So far we assumed that we are given a reference shape Γ_R and a set of deformations fields $u_1,\ldots,u_n$ where each deformation field $u_i:\Gamma_R \rightarrow R^2$ defines a mapping between every point of the reference shape $x \in \Gamma_R$ and the corresponding point $x + u_i(x) \in \Gamma_i$. This allowed us to estimate the shape variations by means of a Gaussian Process defined by the sample mean and sample covariance of these deformation fields. In practice, however, the target shapes $\Gamma 1,\ldots,\Gamma n$, are usually not in correspondence with each other and the deformation fields cannot be easily calculated. The remaining step to complete our tool chain for shape modeling is therefore the computation of these deformation fields. To this end, we select one of the $\Gamma_1,\ldots,\Gamma_n$ be the reference shape Γ_R. We then seek a deformation field $u_i:\Gamma_R \rightarrow R^2$ between the reference and the example shape Γi, which is not in correspondence. This is the very problem that can be solved with the Iterative Closest Point (ICP) algorithm.

2.5.2 Model fitting and registration problem

In the literature, registration and model fitting have usually been treated as two separate problems. Instead of being treated as the task of fitting a model of smooth deformations, registration is usually formulated as an optimization problem [7]:

$$u^* = \ arg\,min_u\ D\ [\Gamma_R,\ \Gamma_T, u] + \ \eta\ R[u] \qquad (2.9)$$

D is a distance measure between two surfaces, R *is* a regularization term and $\eta \in \mathbb{R}$ a weight factor. In this alternative approach, the regularization term R encodes the prior assumption. It penalizes deformations that do not match the encoded assumption. It turns out that many popular regularization approaches can in fact be formulated as a Gaussian Process model.

2.6 Regression in shape modeling

In shape modeling, the regression problem can be formulated as follows, the points $x_1,\ldots,x_n \subset \Gamma_R$ are points on the reference shape Γ_R. The regression model is defined as [7]:

$$u_i = u_i(x_i) + \epsilon \tag{2.10}$$

There are two typical applications of the regression problem. The first one arises when we have obtained a sparse set of measurements of the shape and would like to infer the full shape (see Figure 2.5 left). In the second application, it is possible to obtain an arbitrary number of measurements, but only for a part of the shape.

This setting is typical for shape reconstruction problems, where we are given only a part of a shape and the goal is to infer the shape of the unseen part (see Figure 2.5 right):

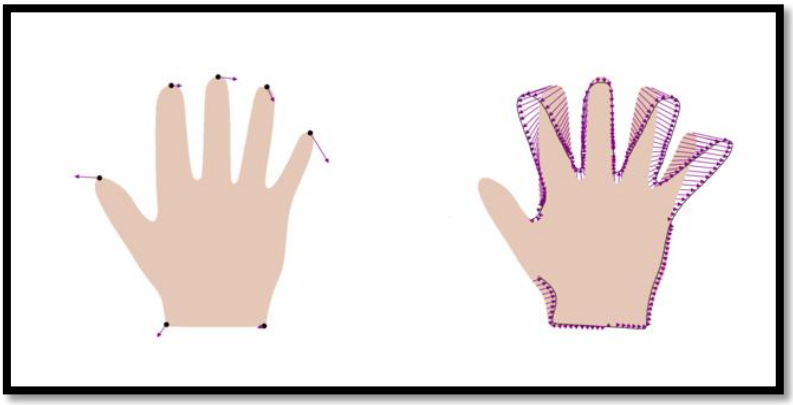

Figure 2. 5: *Two typical applications of regression in shape modeling: inferring the full shape from a sparse set of measurements (left) and reconstructing the complete shape from a given part of the shape (right).*

2.7 Using PCA to visualize shape variation

In PCA, the eigenvectors $\vec{\varphi}_i$ of the covariance matrix Σ are usually referred to as principal components or Eigen modes. The first principal component is often called the main mode of variation, because it represents the direction of highest variance in the data. Accordingly, the second principal component represents the direction that maximizes the variance in the data under the constraint that it is orthogonal to the first principal component, and so on. This property allows us to systematically explore the shape variations of a model;

To visualize the variation represented by the j-th principal component by set the coefficient $\hat{\alpha}_j = v$ and $\hat{\alpha}_{i\neq j} = 0$ and by draw the corresponding sample defined by [4]:

$$\hat{u} = \vec{\mu} + \sum_{i=1}^{m} \sqrt{d_i}\, \varphi_i \alpha_i = \bar{u} + v\sqrt{d_i}\vec{\varphi}_j \tag{2.11}$$

Typically, v is chosen such that $v \in \{-3, 3\}$, which corresponds to a deformation that is 3 standard deviations away from the mean. The figure 2.6 shows the shape variation associated to the first principal component for the hand example.

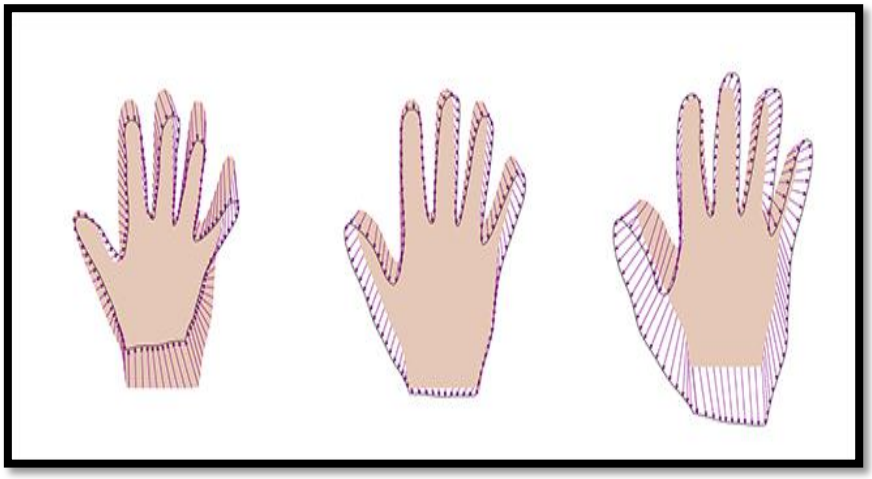

Figure 2. 6: *The shape variation represented by the first principal component of the hand model, the hand on the left shows a deformation with* $\hat{\alpha}_1 = -3$ *, the middle hand shows the mean deformation* $\hat{\alpha}_1 = 0$ *and the hand on the right the deformation with* $\hat{\alpha}_1 = 3$

2.8 Summary

This chapter introduced all the theoretical concepts needed of building statistical shape models that are used in the experimental part

3.1 Scalismo

Scalismo (Scalable Image Analysis and Shape Modeling) is an open source library for statistical shape modeling and model-based image analysis in Scala. It has its origin in the research done at the Graphics and Vision Research Group at the University of Basel. The fact that it's a software library means that it gives the ability either to build an application or integrate it into bigger software projects. The main functionalities of Scalismo is shape modeling. We can build free-form deformation models where we can tune the properties of our model. We can also learn the deformations from data; we can also fit the shape models to image data and surface data, Scalismo also supports 2 and 3D medical image analysis and allows performing image-to-image registration. It also allows manipulating 3D surfaces that can come either in the form of triangle meshes or point clouds. And it also provides tools to perform mesh-to-mesh registration [1].

3.2 Scala language

Scala is a modern programming language that runs on the Java virtual machine. Scala is both objects oriented and functional, which makes it close to mathematics. It is also a language that is both compiled and scriptable. And all these properties make it a scalable programming language that can be used to write programs that are either small scripts or large and heavy. Scala source code is intended to be compiled to Java byte code, Java libraries may be used directly in Scala code and vice versa (language interoperability). Like Java it uses curly-brace syntax reminiscent of the C programming language. Unlike Java, Scala has many features of functional programming languages like Scheme, Standard ML and Haskell, including currying, type inference, immutability, lazy evaluation, and pattern matching. It also has an advanced type system supporting algebraic data types, covariance and contravariance, higher-order types (but not higher-rank types), and anonymous types. Other features of Scala are not present in Java like: operator overloading, optional parameters, named parameters, raw strings, and no checked exceptions. The name Scala is a portmanteau of "scalable" and "language", signifying that it is designed to grow with the demands of its users [8].

3.3 Application on the face shape

3.3.1 Modeling deformations using Gaussian processes

In shape modeling one is interested in studying variations of deformation fields defined over both continuous and discrete domains. Gaussian processes being mathematical

tools allow studying normal distributions over both discrete and continuous functions; they are the perfect tool to model shape deformations.

3.3.1.1 Creating a continuous Gaussian Process

Similar to the Multivariate Normal Distribution (MVN), a continuous Gaussian Process is entirely defined by 2 components, the mean and the covariance. The only difference being that these 2 components are now defined over a continuous domain since we wish to model function variations at any set of points we choose (and not just at the points at which we collected the samples, as is the case for a regular MVN).In order to experiment on GPs, we loaded the following mesh on which to operate:

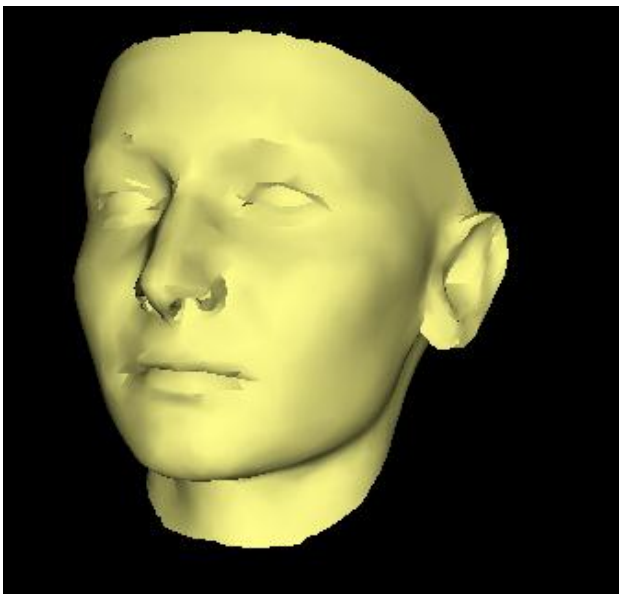

Figure 3. 1: *The reference mesh.*

3.3.1.2 The mean

All the deformations sampled from the Gaussian process will be normally distributed around this mean, which itself is a deformation field.

3.3.1.3 The covariance function

Given that we wish to evaluate covariance at any sample of points we choose, the notion of covariance is no longer given by a discrete matrix but by a covariance function also referred to as kernel. A kernel function guarantees that any matrix is Positive Semi-Definite and is therefore a valid covariance matrix to be used in a Multivariate Normal Distribution.

3.3.1.4 Intuition for the kernel function

An intuitive way of understanding kernel functions, in the context of Gaussian processes, is to regard them as functions that determine the similarity between function values at 2 points of the domain, based on the similarity between the 2 points. In the context of shape modeling, the kernel function therefore specifies the similarity between the deformation vectors defined at 2 points x and x' of our reference shape, based on how similar the kernel judges x and x' to be. We will experiment with the Squared Exponential Kernel, also known as the Gaussian Kernel.

3.3.1.5 Gaussian Kernel or Squared Exponential Kernel

This is perhaps the most common and intuitive kernel. According to its formula, the similarity between the function values should be at highest when the associated points are near and decreases exponentially the further they are. In the formula, two parameters allow to tune the kernel. An intuitive interpretation for these is [5]:

$$K_{SE}(x, x') = S \exp\left(-\frac{(x,x')^2}{L^2}\right) \quad (3.1)$$

1- **L** allows scaling distances in the domain of the input points. This allows tuning how far inter-dependencies should span between the points of the shapes.
2- **S** is a scaling factor that emphasizes the returned similarity. This generally affects the variance of the output values (codomain) and results in more pronounced deformations when increased.

3.3.1.6 Building the Gaussian Process

Since we have defined a mean and covariance functions, we built a Gaussian process and sample deformations from this GP at any desired set of points we choose at the loaded mesh(the reference mesh)in this case as shown below:

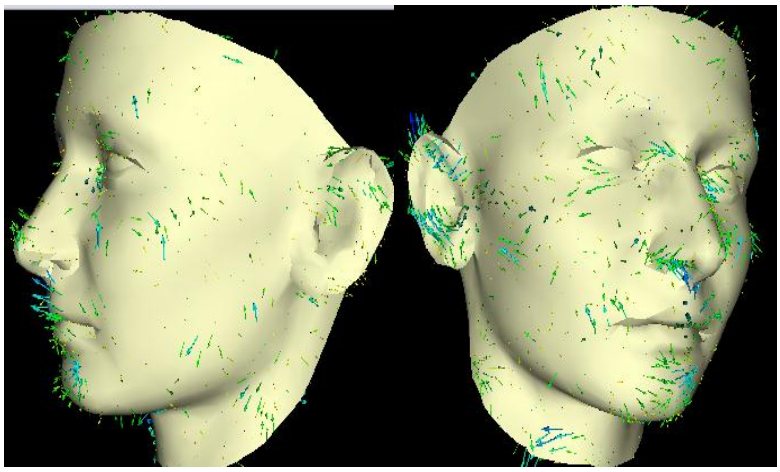

Figure 3. 2: *The desired points to sample the deformation from the defined GP.*

At any desired set of points**.** we can see an instance (or a random sample function) from the Gaussian Process evaluated at the points we indicated; in this case on the reference mesh points. The next figure represent a visualization to the effect of this deformation field on the reference mesh initially we assign S=L=10 as an initial values:

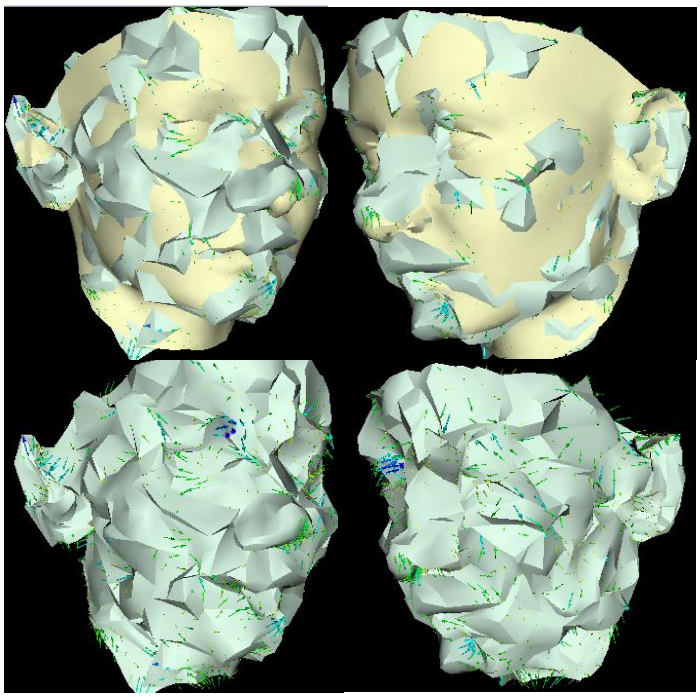

Figure 3. 3: *The result of the deformation With: S=10 L=10.*

We obtained an unsatisfactory result as it yields a very rough mesh. As we are interested in obtaining smoother deformations, we changed the *L* parameter in the chosen Gaussian kernel to enforce more correlation between distant points of the shape, so we reiterated all steps

performed above while simply changing the value of *L* to 40 mm instead of 10 as shown below:

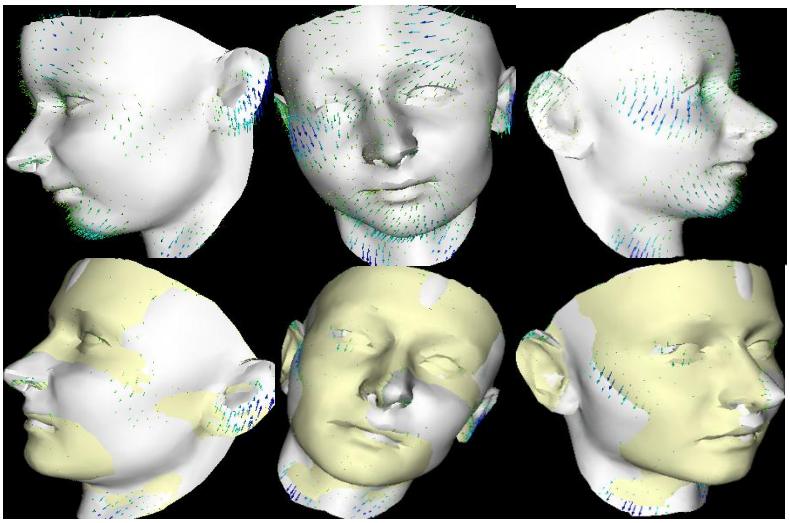

Figure 3. 4: *The result of the deformation with: S=10 L=40*

We obtained a smooth deformation field which in turn results in a smooth deformation of the reference mesh. Again we reiterated all steps performed above while simply changing the value of L to 50 mm instead of 10, see the next figure:

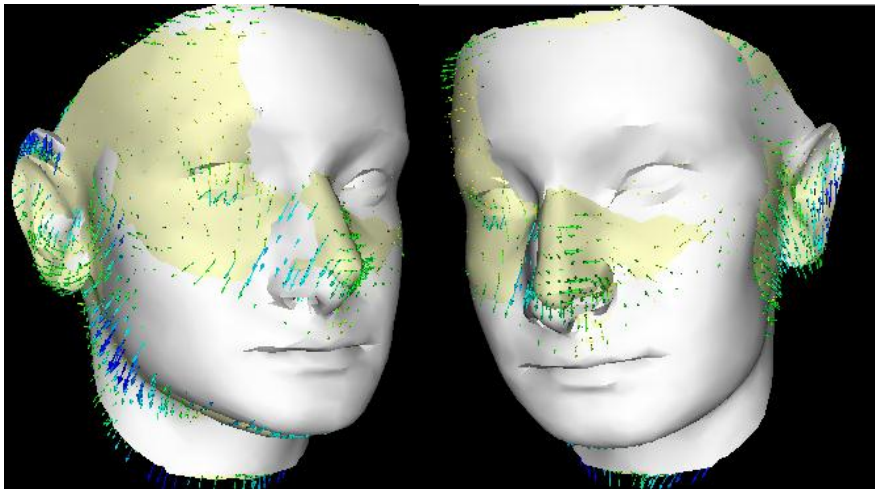

Figure 3. 5: *The result of the deformation with: S=10 L=50.*

The result is a smoother mesh then before, still we reiterated all steps performed above while simply changing the value of *l* to 90 mm instead of 10 as the following:

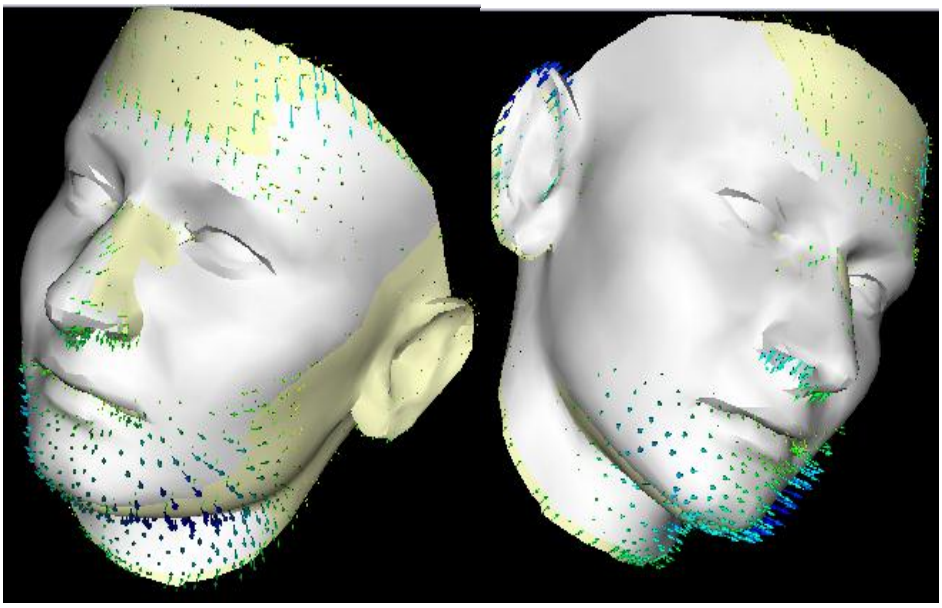

Figure 3. 6: *The result of the deformation with: S=10 L=90.*

We obtained yet a smoother one then before. We really proved that the higher L the smoother the deformation field we get. Next we reiterated all steps performed above while simply changing the value of *S* to 40 mm instead of 10. And fix L to 10 as indicated in the coming figure:

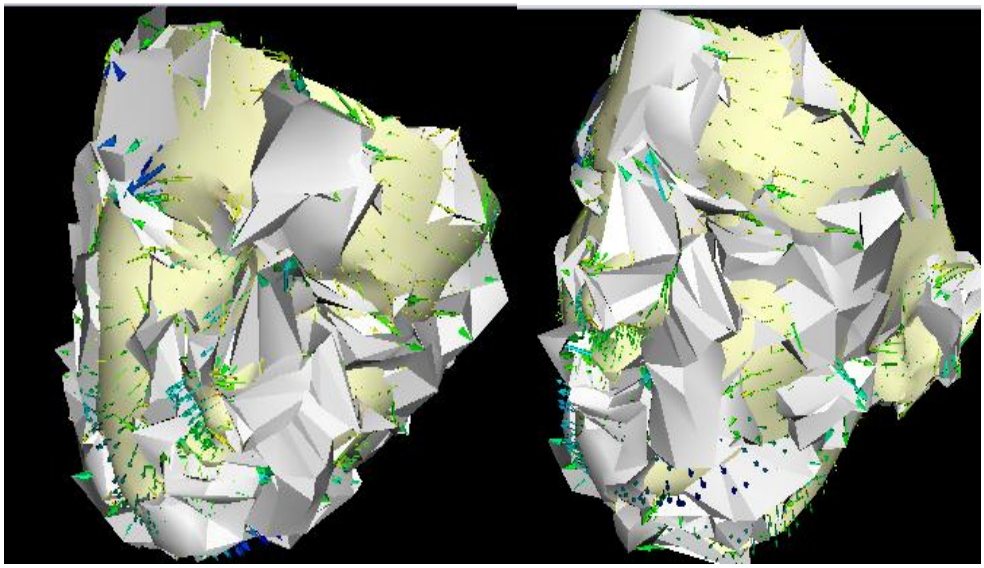

Figure 3. 7: *The result of the deformation with: S=40 L=10.*

In this one we have more pronounced deformation when S parameter is increased. Again, we reiterated all steps performed above while simply changing the value of *S* to 90 mm instead of 10. And fix L to 10 as the following:

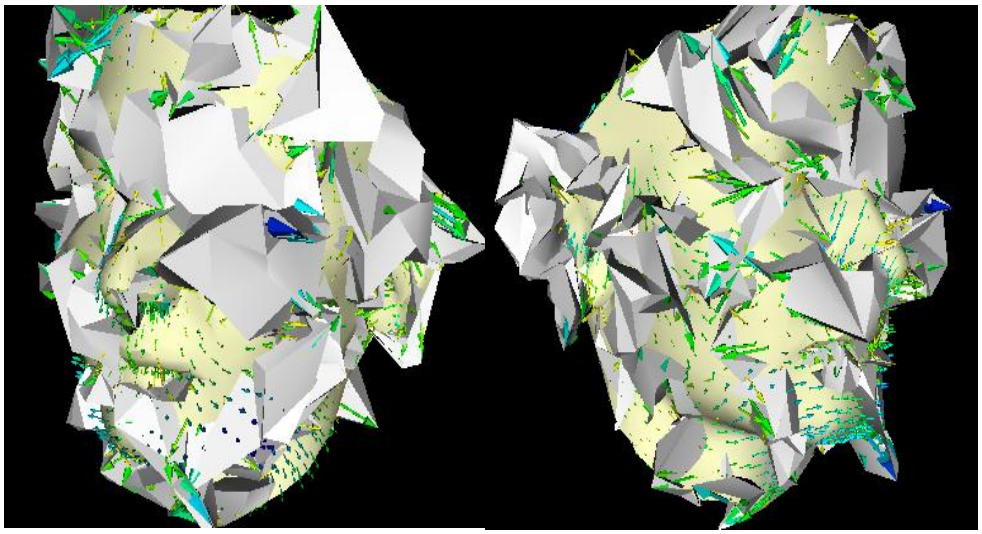

Figure 3. 8: *The result of the deformation with: S=90 L=10.*

Yet more deformation is obtained while we kept increasing the S parameter .So we proved that the S parameter should kept small to avoid high edgy deformations. We finally confirmed the effect of the parameters on the behavior of the kernel. To conclude this part we should mention that there exist many other kernels beside the Gaussian kernel. We can strongly modify the behavior of the deformation by simply plugging a different kernel function.

3.3.2 Building a shape model from data

Our goal is to build a Statistical Shape Model from data in correspondence and assess the importance of rigid alignment while doing so. First, we started by loading a dataset of faces based on which we would like to model shape variation:

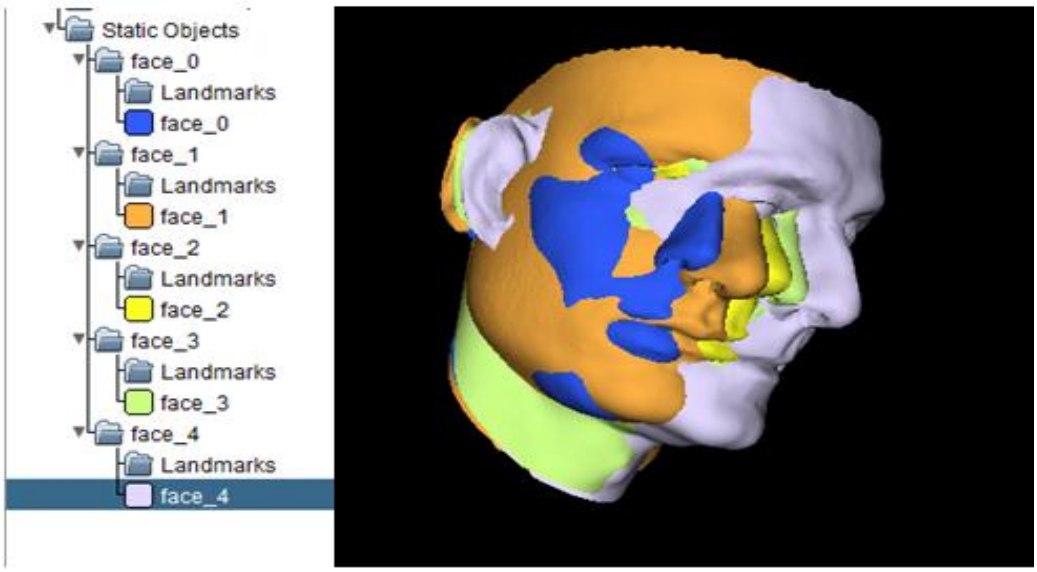

Figure 3. 9: *Visualization of the dataset.*

Two things are to notice regarding this dataset:

1 - The shapes are not aligned, as shown in the above figure.

2 - The shapes are in correspondence: That means that for every point on one of the face meshes (corner of eye, tip of nose ...), one can identify the corresponding point on other meshes. In order to study shape variations, we have to eliminate variations due to relative spatial displacement of the shapes (rotation and translation).We can achieve this by selecting a reference first and then aligning the rest of the dataset to the reference.

3.3.3 Alignment

We started this part by loading both meshes that are misaligned as shows below :

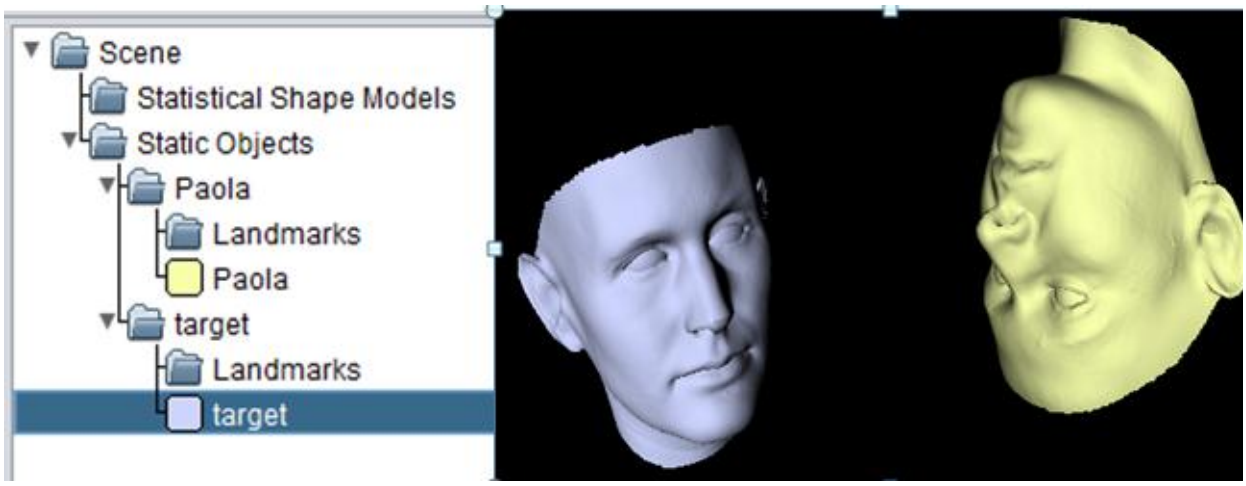

Figure 3. 10: *Miss aligned faces*

A key algorithm used in building this program was the Iterative Closest Point (ICP) method used to perform this rigid alignment.

3.3.3.1 Selecting Candidate correspondence

The best rigid transformation when given correct correspondences can be found in a closed-form solution using Procrustes analysis. Which is The first main idea behind the ICP algorithm, even though we do not have correct correspondences, we can still propose candidate correspondences and use the closed form solution to find a candidate rigid transformation. To do so, we started by selecting a set of points from Paola's mesh (the Basel face model) as the upcoming figure illustrates:

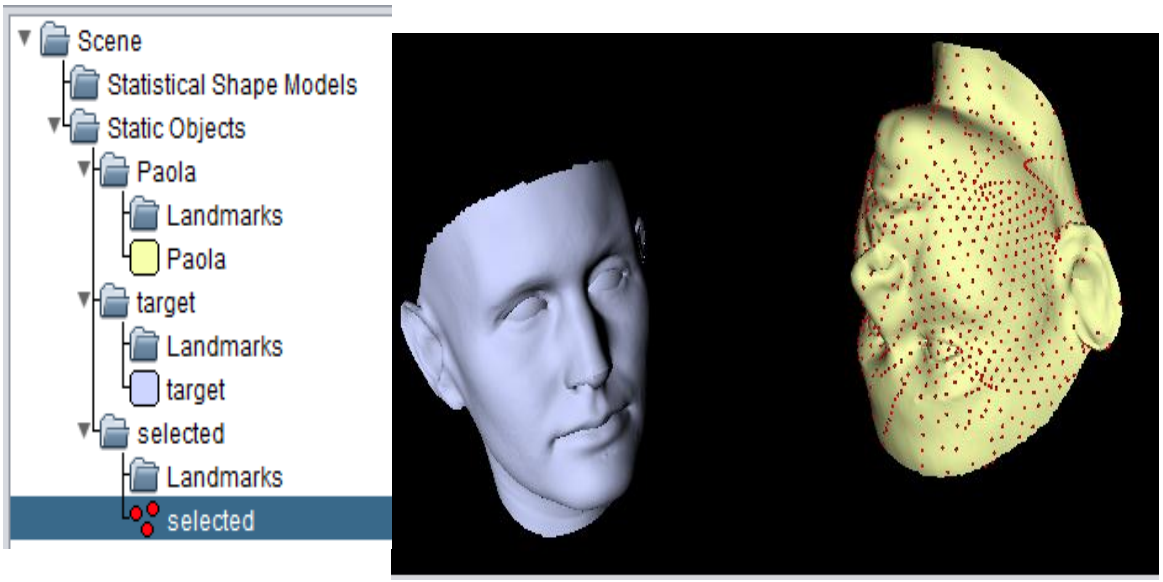

Figure 3. 11: *Selecting the landmarks*

Here, we selected one in every fifty point identifiers on Paola's mesh to obtain a uniformly distributed sample of points on the surface.

3.3.3.2 Finding candidate correspondences for the selected set of points

The idea is to attribute to each selected point its closest point on the target mesh as a candidate corresponding point. we define a function that does exactly that, Given a triangle mesh to be rigidly aligned to the target (in our case this will be Paola's mesh), this function loops on the point identifiers selected above and associates to each selected point on the mesh to be transformed, its closest point on the target. The function therefore returns an indexed sequence of point tuples, where the elements of the tuple are candidate corresponding points on the mesh to be transformed and on the target respectively as shown next:

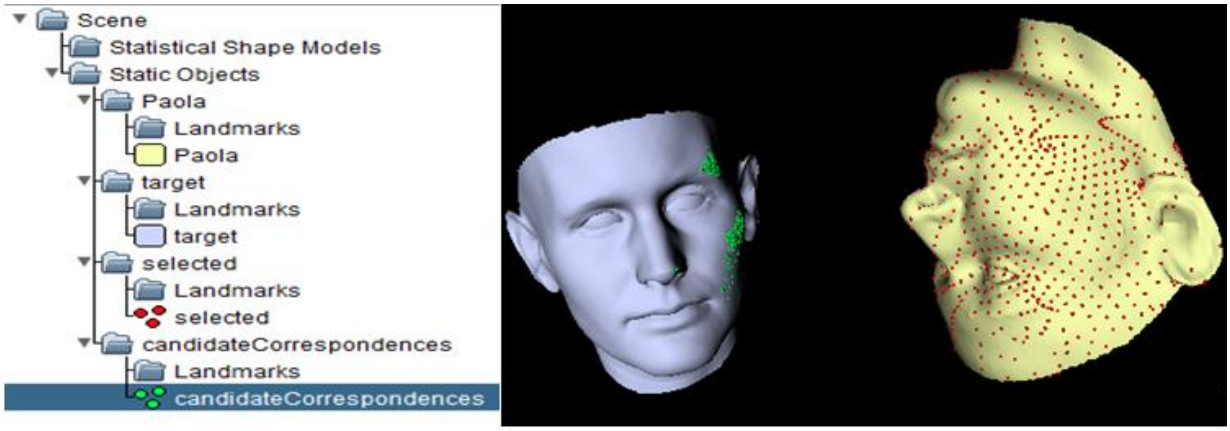

Figure 3. 12: *The correspondance points*

The Result obtained here is that the candidate correspondences were not good correspondences as they tended to focus on only one side of the target face.

3.3.3.3 Procrustes analysis

Although we had bad correspondences we yet applied *Procrustes* analysis based on these candidate correspondences and retrieve a candidate rigid transformation to align Paola:

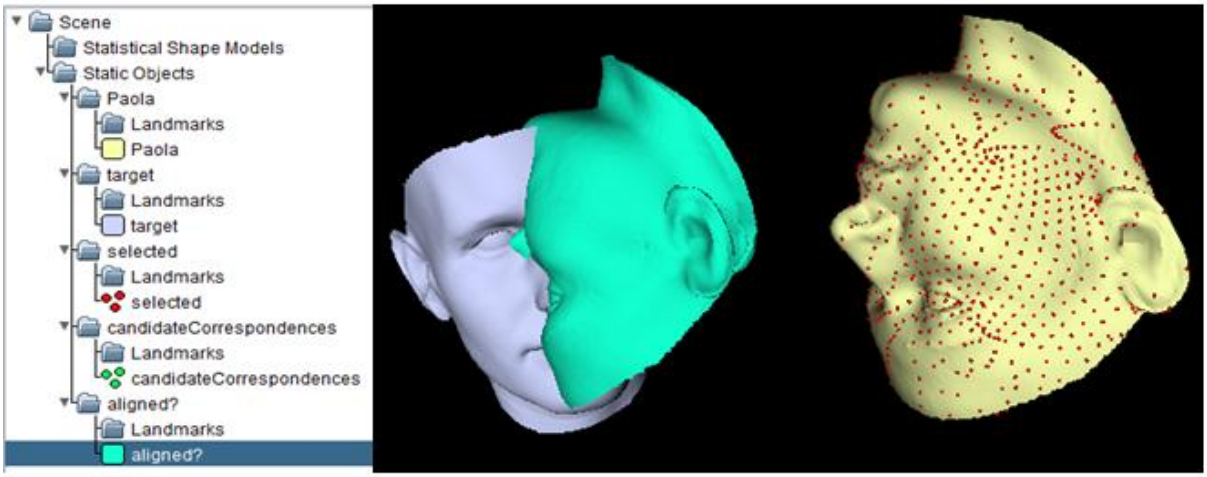

Figure 3. 13: *Result of Procrustes method*

Given the poor quality of the candidate correspondences, we obtained a poor rigid alignment. But considering where we started from, that is the original position of Paola's mesh; we did get closer to the target.

3.3.3.4 ICP and the recursion concept

This is where the second important idea of the ICP (iterative closest point) algorithm comes into play: which is iteration. Now that we have a new starting position that is closer to the target mesh, if we were to repeat the procedure above, we hoped to get better candidate correspondences thus resulting in a better rigid alignment. Here, we just reiterated all the steps performed above while simply using the transformed mesh (named *aligned* in the 3D scene) as a new starting point for attributing candidate correspondences.as a result the candidate correspondences were still wrong, but started to be more spread around the target face. Also the resulting rigid transformation seemed to bring our mesh a bit closer to the target as shown next:

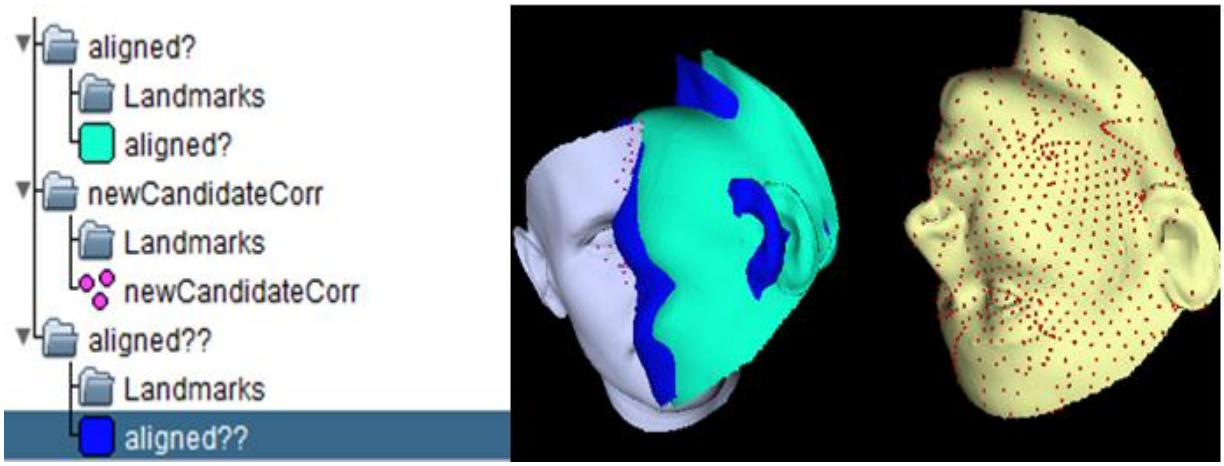

Figure 3. 14:*First iteration result*

After that we simply wrapped the steps performed above in a recursive function such that we can do them repeatedly. At every recursive call, we start by attributing candidate correspondences based on the latest position of Paola's transformed mesh. We then solve for the best rigid transform, use it to transform the moving mesh and then recurse (transformed mesh will then be used for attributing candidate correspondences, et ...).As a halting condition for the recursion, we indicated a number of iterations to stop recurring when the counter is at 0.Additionally, at every 15th iteration, the current moving mesh is displayed in order to follow the progress of the iteration. We started testing the program with 50 iterations and the result is shown below:

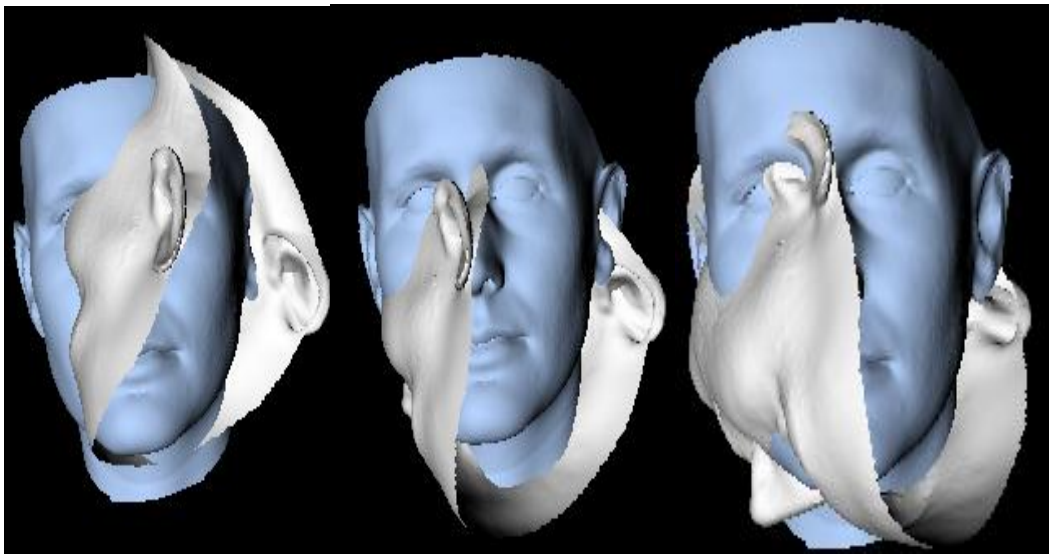

Figure 3. 15: *The alignment result with 50 iteration.*

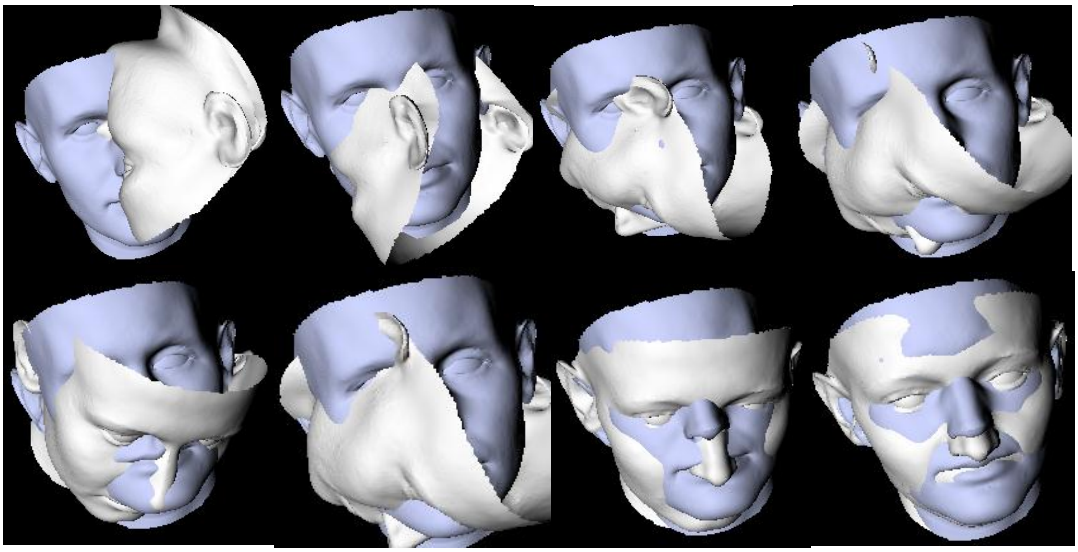

Figure 3. 16*: Alignment result for150 iteration.*

We retested the program again with 150 iterations and the result is shown in the above figure, we obtained better alignment then the previous test; the two meshes are aligned on each other but looking carefully, one can see some rough edges along the nose area. Therefore We retested the program with 200 iterations to see if we can get better result than this, the result is shown in the figure below:

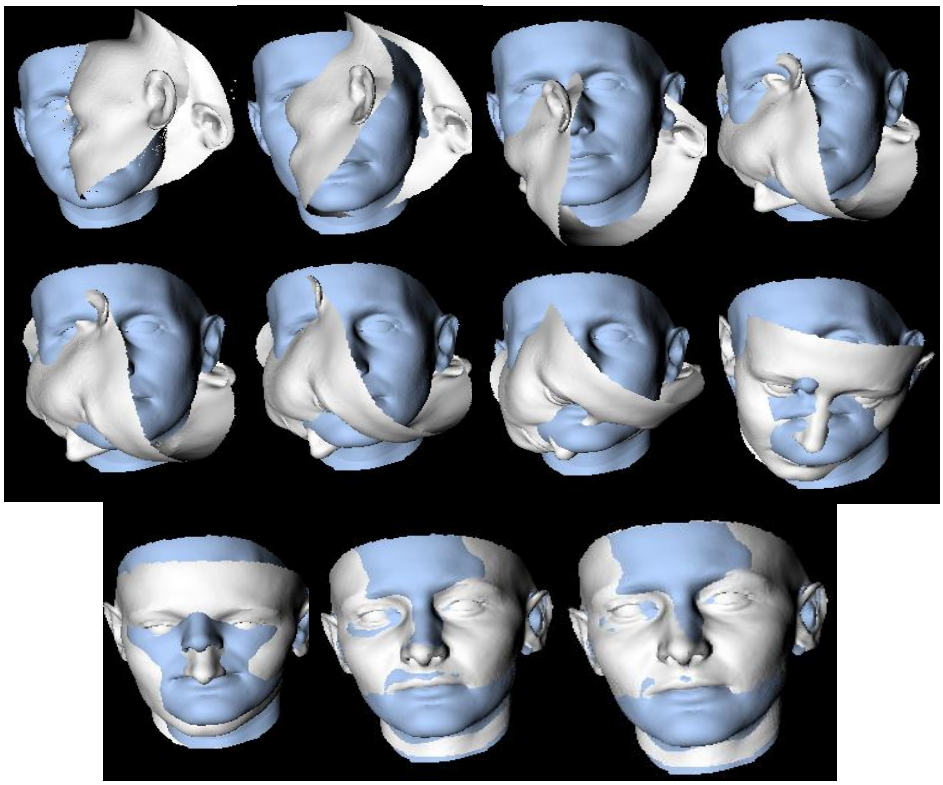

Figure 3. 17: *the alignment result for 200 iteration*

Indeed The result is better than the one obtained before. There is a clear match between the two meshes. Now In case our data set is not in correspondence we need to apply a fitting algorithm to make our data in correspondence.

3.3.4 Model fitting with Iterative Closest Points

We are currently displaying in the 3D scene an instance of the Basel Model (the mean), that does not resemble to the target face. The goal in shape model fitting is therefore to find an instance of the shape model that resembles at best the given target face. We loaded a mesh to fit a (target) and the Basel Model face:

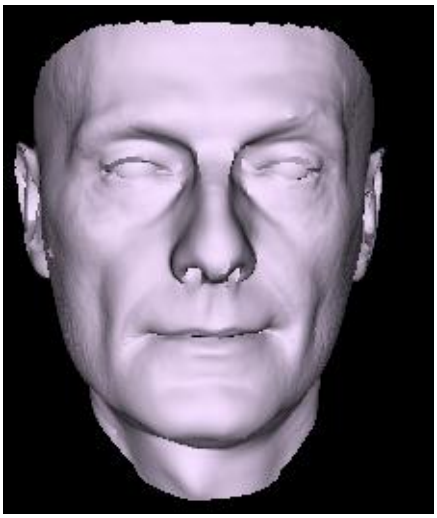

Figure 3. 18: *The Target*

Figure 3. 19: *The Model*

3.3.4.1 Iterative Closest Points (ICP)

We've seen before how to use ICP in combination with Procrustes analysis in order to find the best rigid transform aligning one mesh to another. Here, we will perform exactly the same steps to fit our model to the target while substituting Procrustes analysis with Gaussian process regression.

3.3.4.2 Fitting a few characteristic points

To keep things simple, we first try to find correspondences for a few characteristic points like the corners of the eyes, tip of the nose, corners of the lips and at the level of the ear borders, down on the lobe and up on the helix as the following figure illustrates:

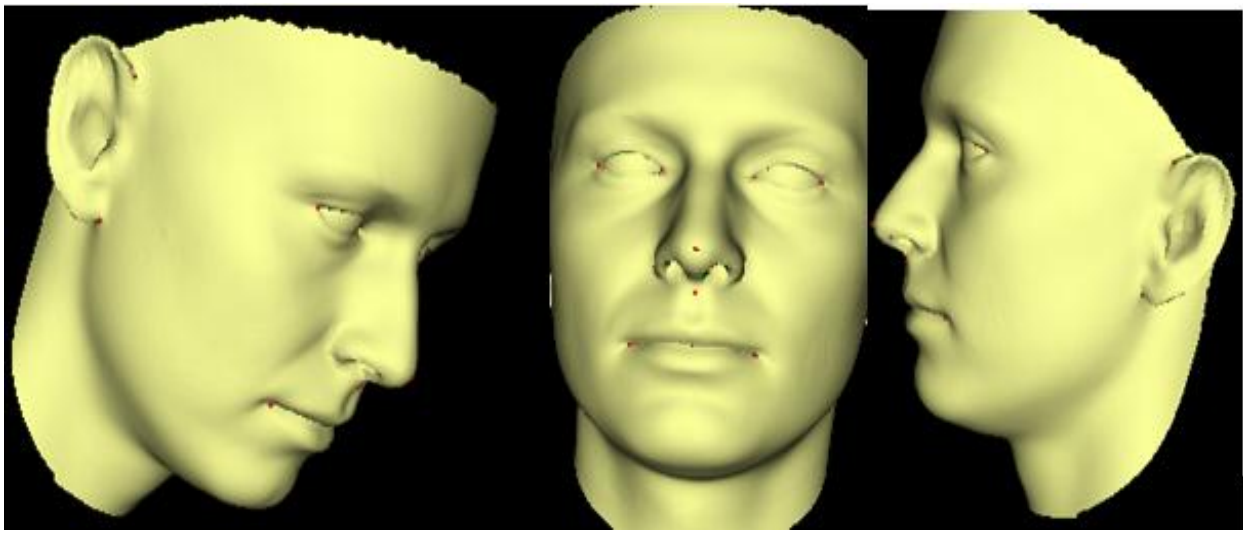

Figure 3. 20: *Selecting landmarks on the model.*

We selected a set of easily identifiable points on the model (see in 3D scene), for which we will seek corresponding points on the target mesh.

3.3.4.3 Obtaining candidate correspondences

We started by defining our candidate correspondences attribution method and used it on the loaded points:

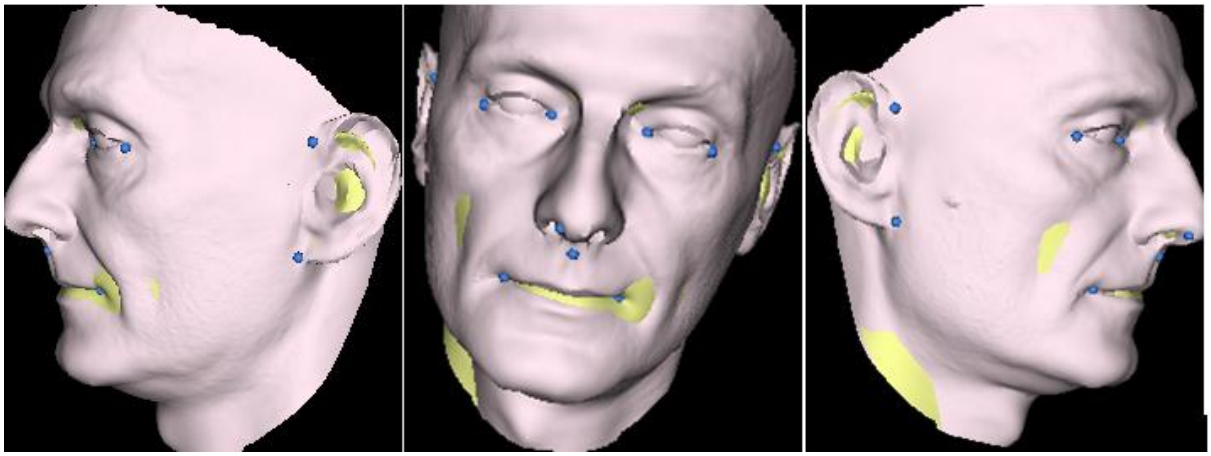

Figure 3. 21: *Correspondance points on the target.*

The defined function is very similar to the one we defined for the rigid alignment ICP introduced previously. Given a set of points, in this case belonging to an instance of our shape model, we attribute the closest point on the target as a candidate correspondence to each one of these points. When visualizing the attributed candidate correspondences, we see that we get an unsatisfying initialization, while some points such as some corners of the eyes are well initialized, other points such as the tip of the nose are still poorly fit. The first main idea

behind ICP is, even though the candidate correspondences are still not perfect, to still proceed to step 2 and use them in a GP regression to find the best model instance explaining the observed deformations. Let us start by visualizing the partial deformation field (or candidate observations) that we will feed to our regression as follow:

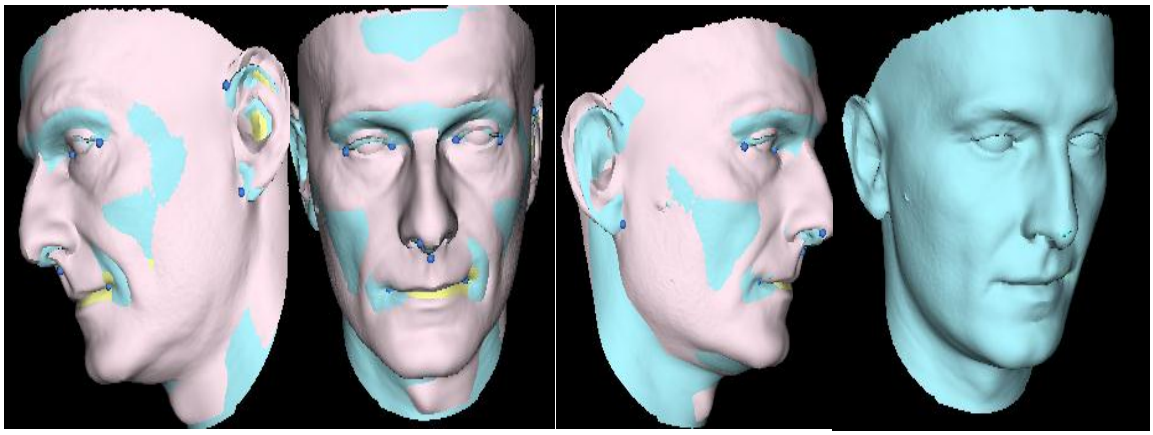

Figure 3. 22: *The correspondence result with few points.*

When given a sequence of identifiers of model points and their candidate correspondence positions, computes a GP regression based on the resulting deformation field and returns the model instance fitting at best the candidate deformations. As we are now limiting our interest to the few characteristic points, let us visualize their new position on the obtained fit:

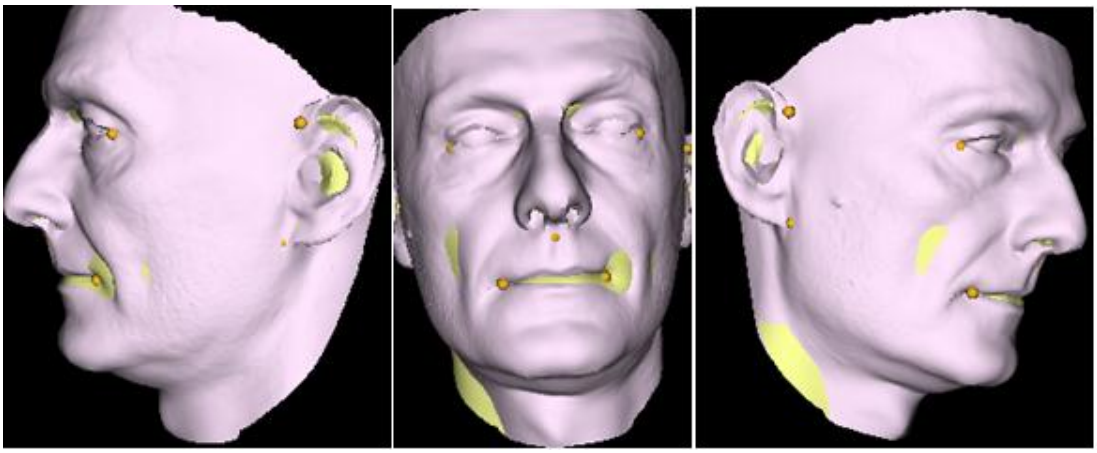

Figure 3. 23: *New positions of landmarks.*

We make the target slightly transparent and color fitted Points to visualize better, Notice here that the set of points we just displayed are real corresponding points to the set of chosen landmarks on the model as they are taken from the model fit using the same point identifiers.

Figure 3. 24: *Making the target slightly transparent to visualize the landmarks*.

When comparing the positions of these points with regard to the target mesh, we see that they are still far from their corresponding position on the target (corner of eye, tip of nose, etc. ...). However, when considering the initial position of these points, that are the loaded landmarks positions, one might argue that we are at a better position than where we started. This is where the second important idea of ICP comes again into play: iteration. If we were now to repeat the same operations above (attribute candidate correspondences and use them to obtain a model fit), this time however starting from the new points positions (new Points in the scene), and we could hope to get an even better model fit. Here we defined a recursive function that repeats the exact same procedure we performed above for a given number of iterations. At every iteration, candidate correspondences are attributed to the input list of point positions. A Gaussian process regression is then performed and a new position for the points of interest is computed. The new point positions are then used in the recursive function call to indicate the new starting position. At every iteration, the new positions of the corresponding characteristic points are displayed (with a 3 second delay for us to be able to follow the fitting progress).When now calling the recursive function for 5 iterations, starting from the loaded landmark positions, we should see the characteristic points gradually getting to better positions.as the following figure:

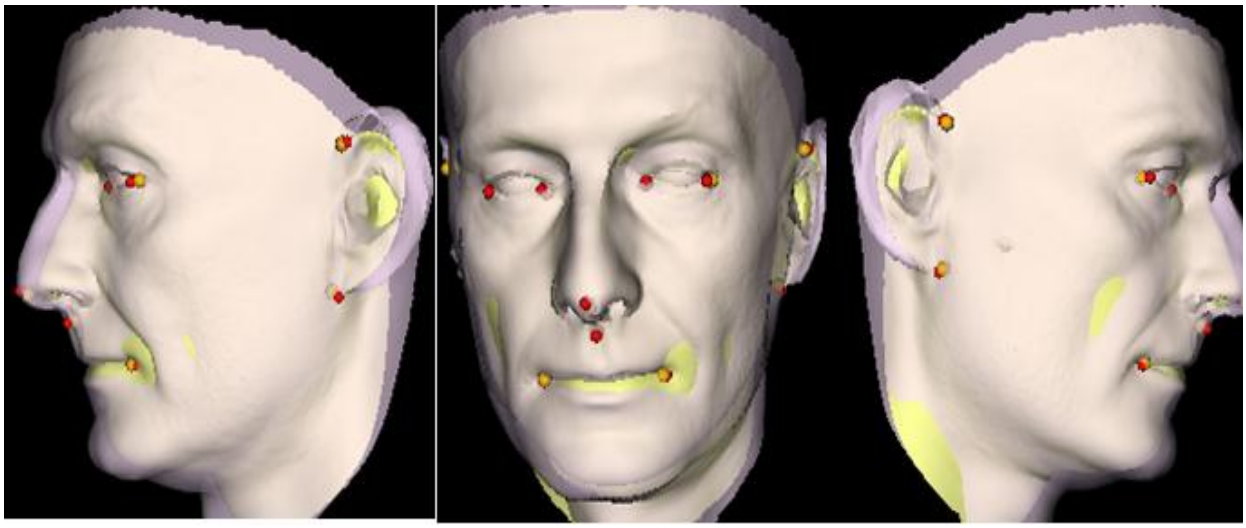

Figure 3. 25: *The improvement of the characteristic point result.*

The final position of the characteristic points is then considered to be our model fit. These points are therefore the obtained correspondences with the target shape.

3.3.4.4 ICP with more points

Now do the exact same operations above, this time with much more points of interest, Here we start by uniformly sampling 5000 points on our model and saving the identifiers of the closest mesh vertices to these points. These vertices will now be the points for which we will seek correspondences. We then define a recursive fitting function again, very similar to the one above, with the only difference that we are now visualizing the entire fitted mesh at every iteration and no longer limiting our attention to the fitted points. We can and this shown next:

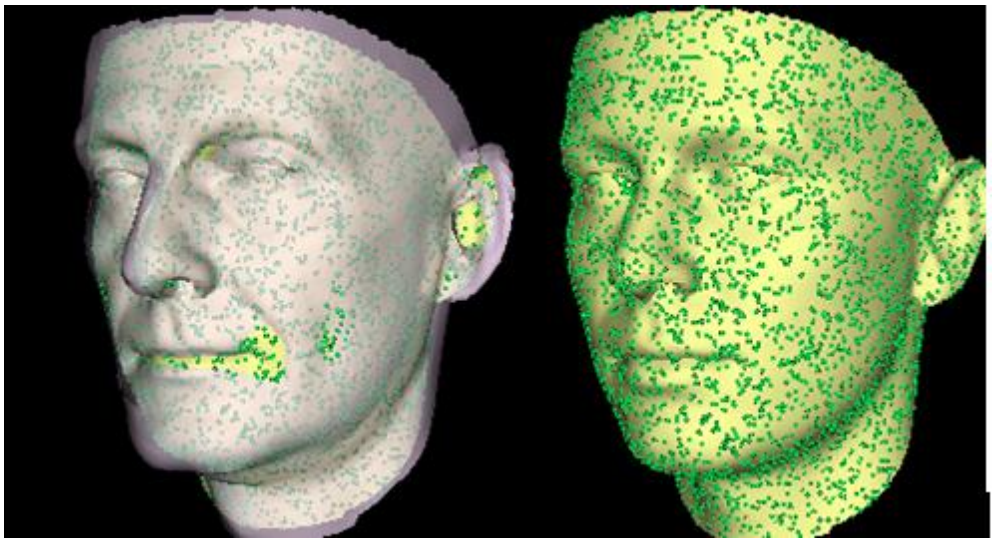

Figure 3. 26: *Taking a large number of landmarks.*

We can now call the recursion with 10 iterations to obtain our model fit; that is an instance of our face model that resembles the given target mesh; the results are visualized in the coming figure:

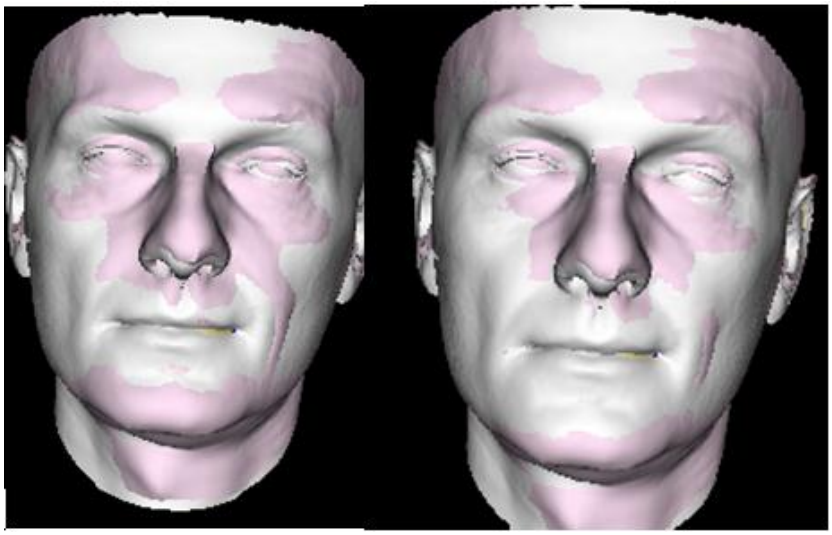

Figure 3. 27: *The fitting result for 10 iterations.*

We have obtained a good result of fitting as the figure above shows .We retested the program of fitting with 50 iterations, the result is shown below:

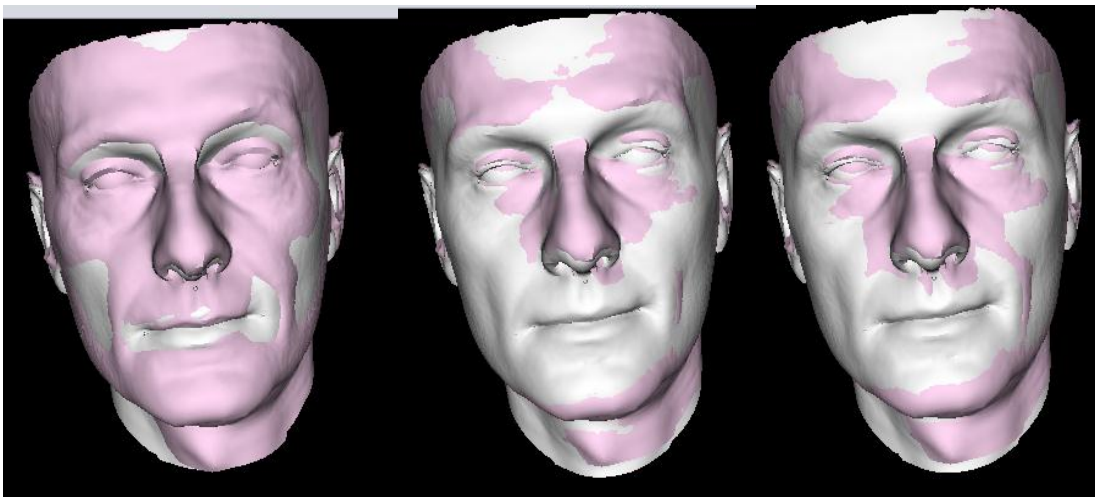

Figure 3. 28: *fitting result for 50 iterations.*

We obtained the same result as we did when we iterate 10 times which proves that the program converges after some iteration. Finally we can here finally build a model from our giving dataset.

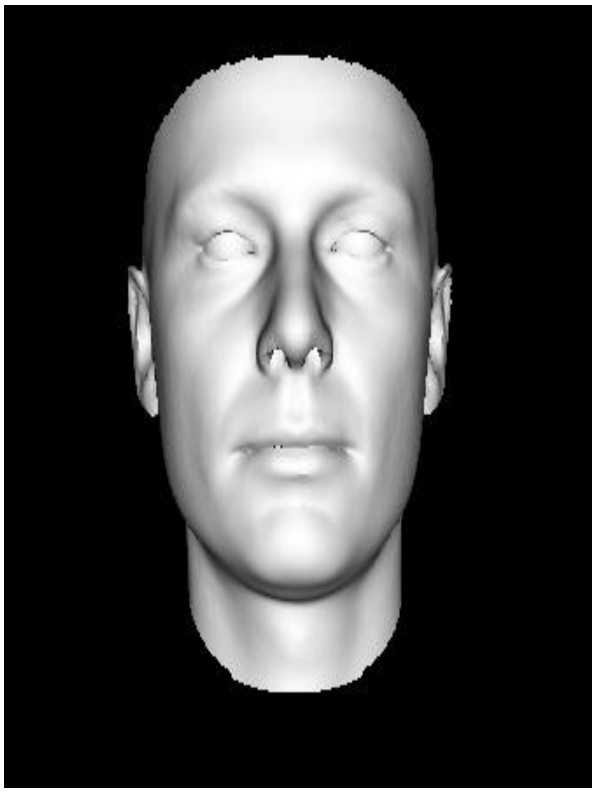

Figure 3. 29: *The face model.*

We can now sample from our model to obtain faces that are not originally in our data set by changing the PCA components .Since we have 5 items in our data set we have 5 components (parameters) in our PCA that controls the shape variation in our model. By changing the parameters we get many samples as the following:

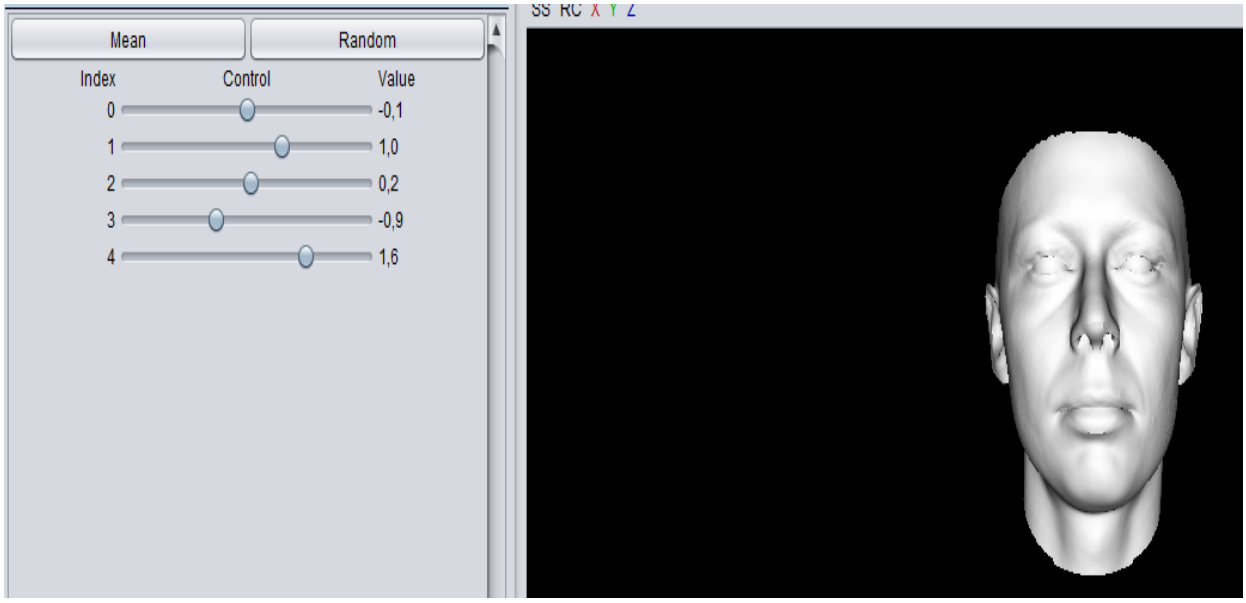

Figure 3. 30: *The first generated face.*

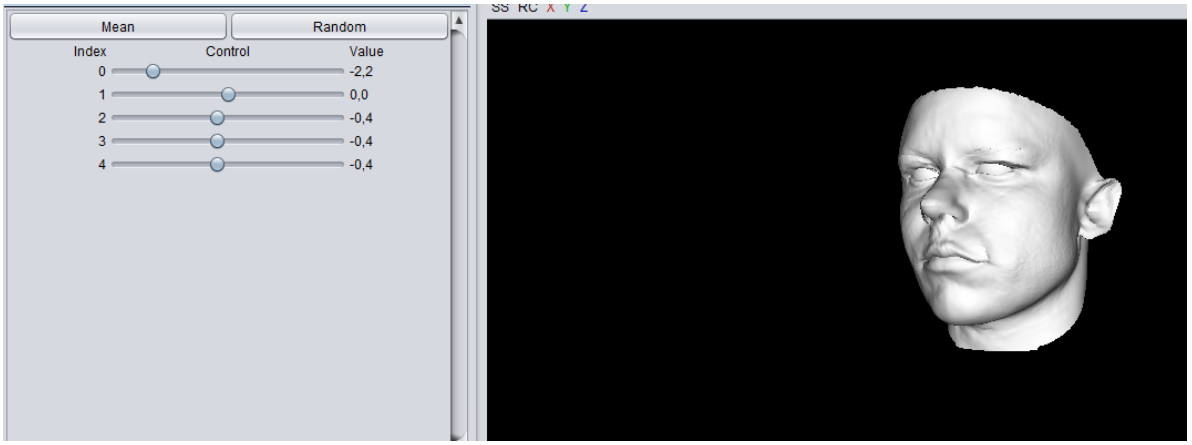

Figure 3. 31: *The second generated face.*

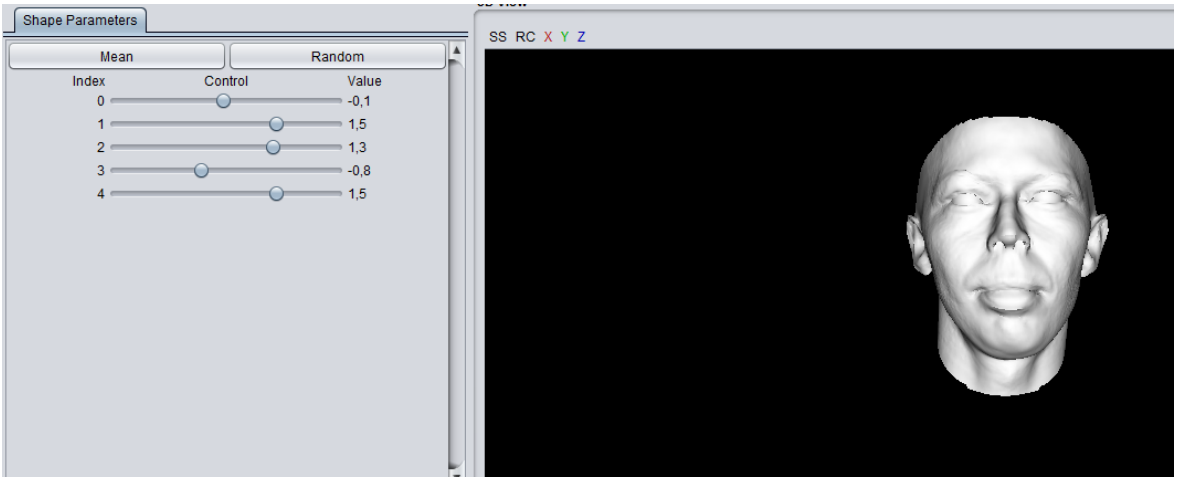

Figure 3. 32: *The third generated face.*

3.4 Application on femur

This application will proceed in several steps; we start by building a femur model using only one provided femur mesh as reference and the Gaussian Process modeling skills acquired so far. Next, we proceed by having an access to SICAS (Swiss Institute for Computer Assisted Surgery) Medical Image Repository (SMIR) [10] and downloading a set of femur meshes and corresponding landmarks that we will need for building a statistical shape model. For the final step we use the femur model we have built in the first step and fit it to the dataset prepared in the second step. The goal is to establish correspondence and build a statistical femur model from data.

3.4.1 Building a Femur model

To show the need of statistical shape modeling in many fields we chose the biomedical field where we build a femur model for a database downloaded from the Sicas Medical Image

Repository (SMIR) that contains 50 simulated femurs of 50 people to help explore the shape variations of a femur for medical purposes. As mentioned above, the goal in this step is to build a femur model. To do so, we proceed as follows: We load the provided reference femur mesh, shipped with the Scalismo package under datasets/femur.stl. which is displayed below:

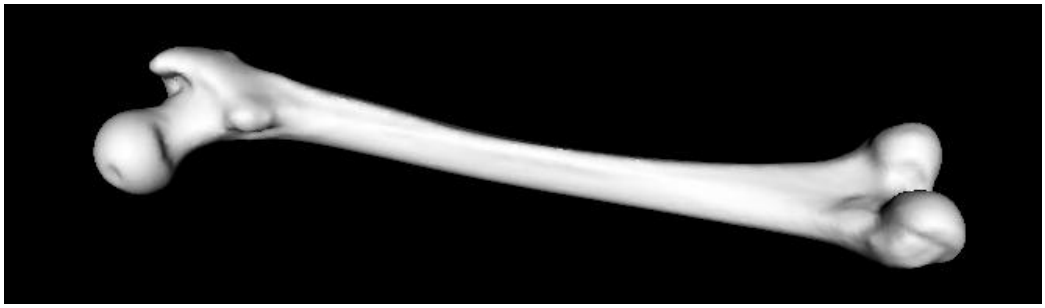

Figure 3. 33: *The reference mesh.*

We use the shape modeling concepts to build a Gaussian Process after testing many kernel functions until the combined kernels yields smooth deformations and combine it with the provided reference mesh to build a Statistical Mesh Model of the femur. The visualization is very crucial at this stage. We made sure to visualize the shapes resulting from our model and we verified that it exhibits the right amount of flexibility.

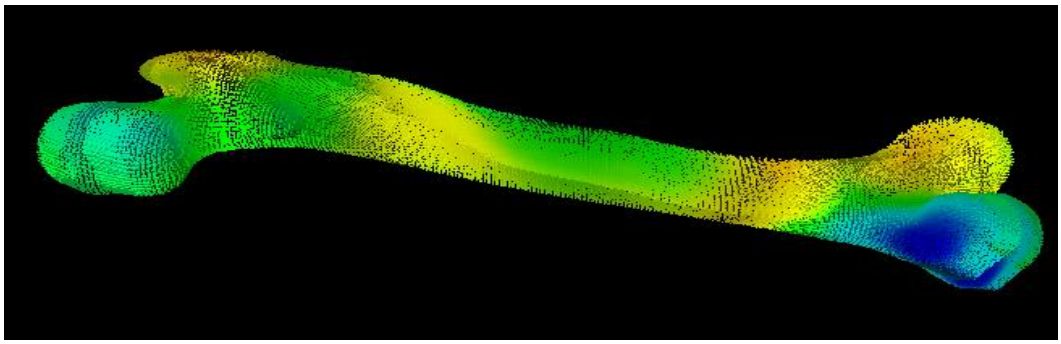

Figure 3. 34*: The resulting deformation field.*

In the above figure we disabled the visibility property using the GUI interface to display the resultant deformation fields obtained.

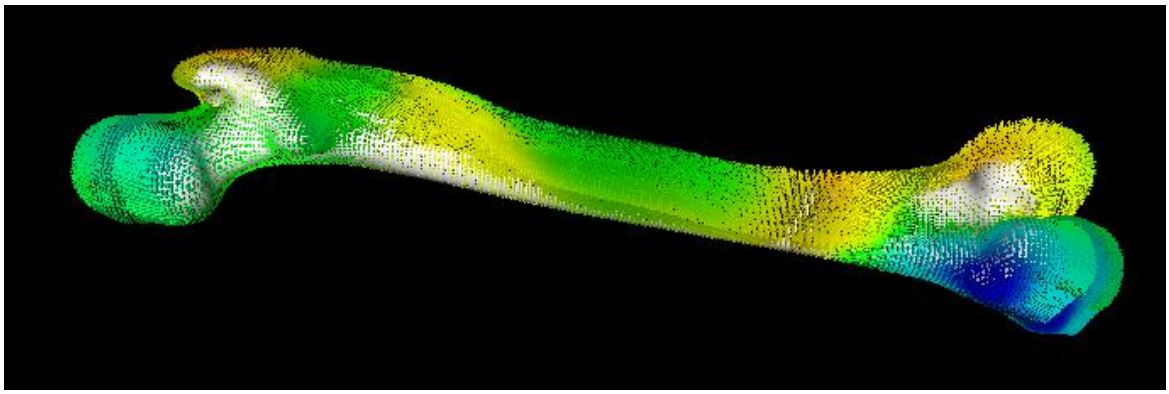

Figure 3. 35: *Visualization of the deformation field from our reference.*

We then wrapped around the deformation vectors over the reference to build our Gaussian model.

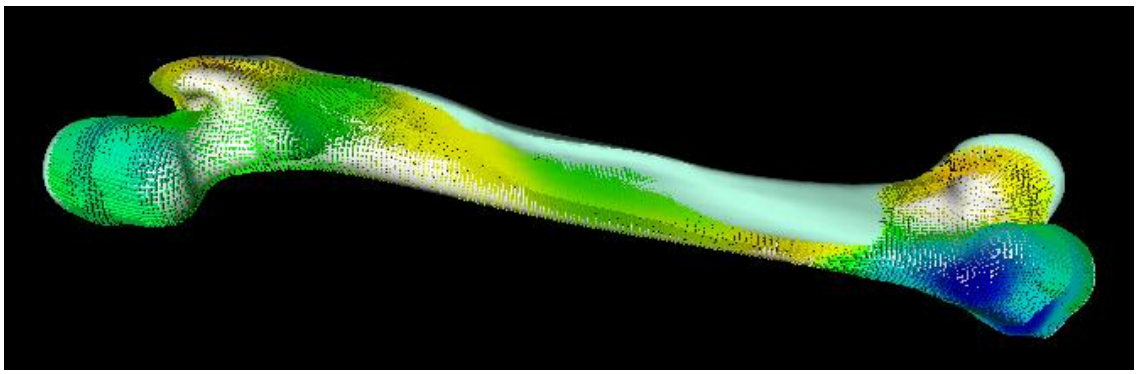

Figure 3. 36: *Wrapping our model over the deformation field.*

After disabling the visibility option on our reference mesh and the deformation vectors we are left with our model as shown below:

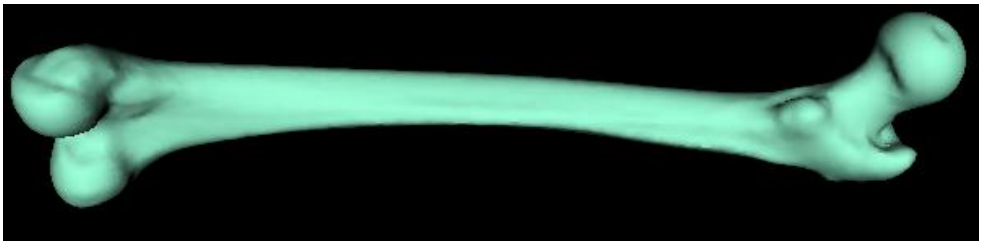

Figure 3. 37: *The visualization of our model alone.*

Now that we have satisfied with the generated Statistical Mesh Model, we stored it to a file to be used in the following steps

3.4.2Aligning the data

Once the data is downloaded, we rigidly align the provided surfaces to the reference femur mesh provided in the first step of the project. To do so, we proceeded as follows: We Locate the landmarks associated with the reference mesh in our data set as the following figure shows:

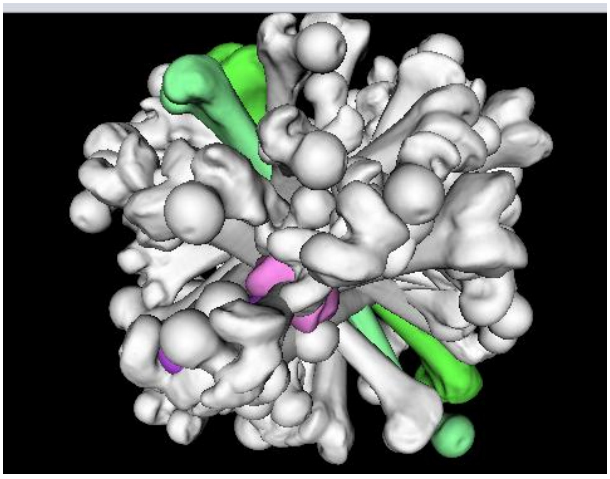

Figure 3. 38*: Loading our dataset.*

Next, we rigidly align the provided femur meshes to the given reference mesh using the landmark alignment method seen previously and the landmarks provided with the meshes, see the next figure:

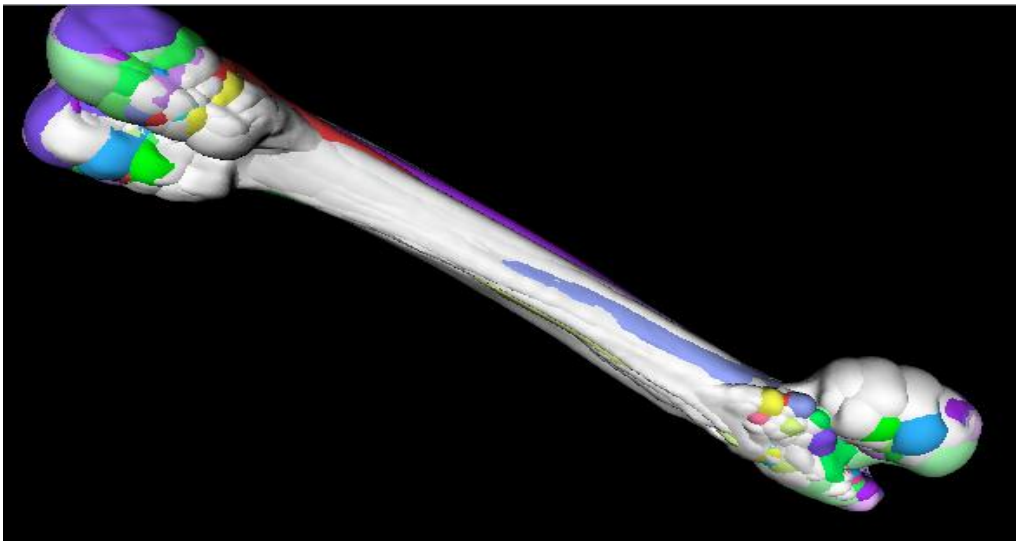

Figure 3. 39: *The result of the alignment.*

The Important thing is to visualize the aligned meshes and to assess that we have obtained a correct alignment. Clearly in the previous figure the femurs are not in correspondence.

Therefore, in order to build a statistical shape model from this data, we needed to fit them to the model built in the first step using Gaussian Process and Kernels.

3.4.3 Fitting the data to our model

We used the Iterative Closest Point (ICP) model fitting method to fit each femur in the database to the model we created in the beginning:

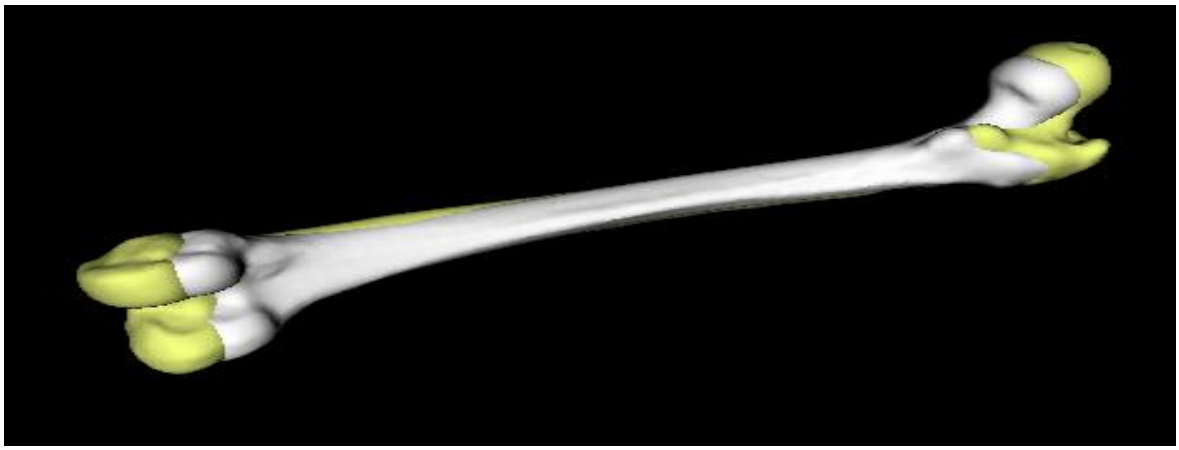

Figure 3. 40: *The Miss-matched femurs.*

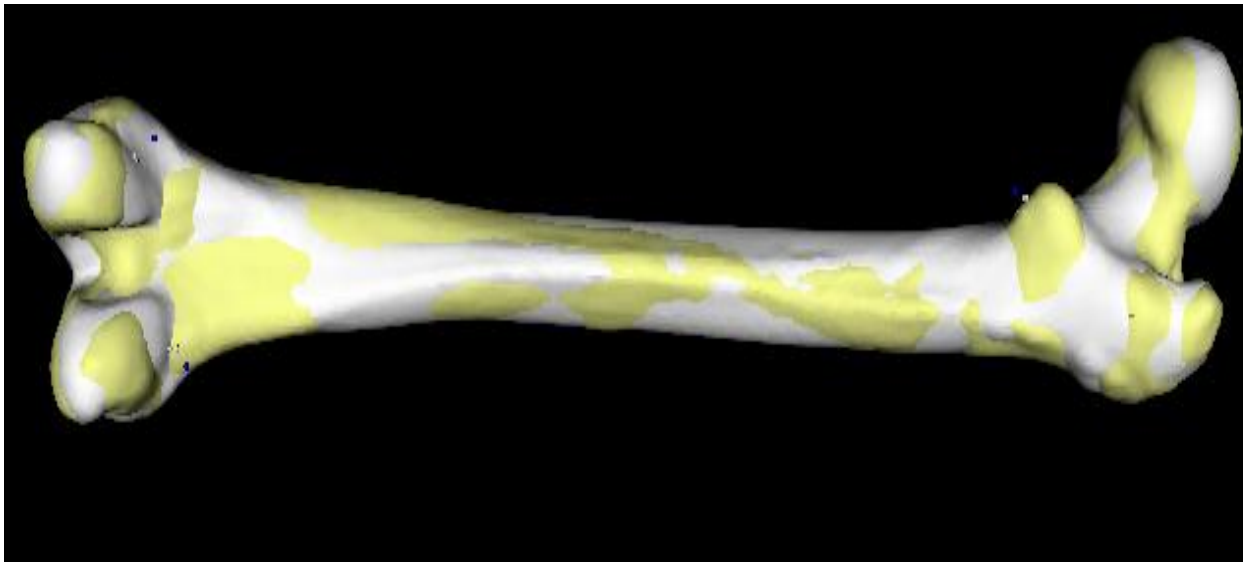

Figure 3. 41: *The result of the fitting.*

We applied the fitting algorithm on each one separately and saved the resulting femur after the fitting in a file. The reason we treated our data separately was because it needs a large memory to do it, after applying it to all the femurs we displayed all the treated femurs to ensure that the result is a good one, which is shown below:

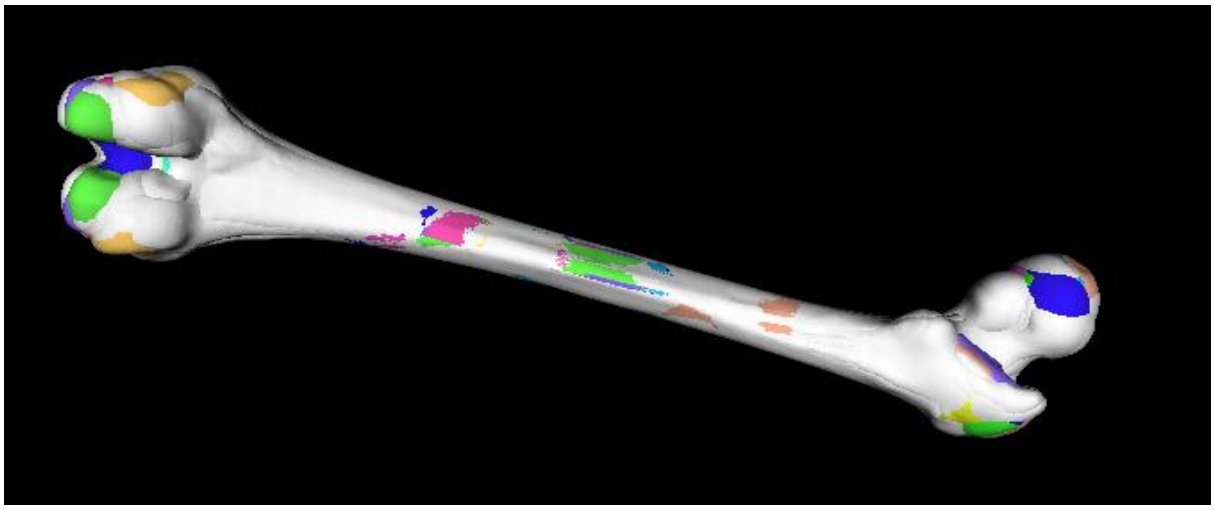

Figure 3. 42: *Displaying the 50 fitted femurs.*

Now that we have our dataset aligned and in correspondence we built a statistical model that represent our data and the possible shape variations as shown next:

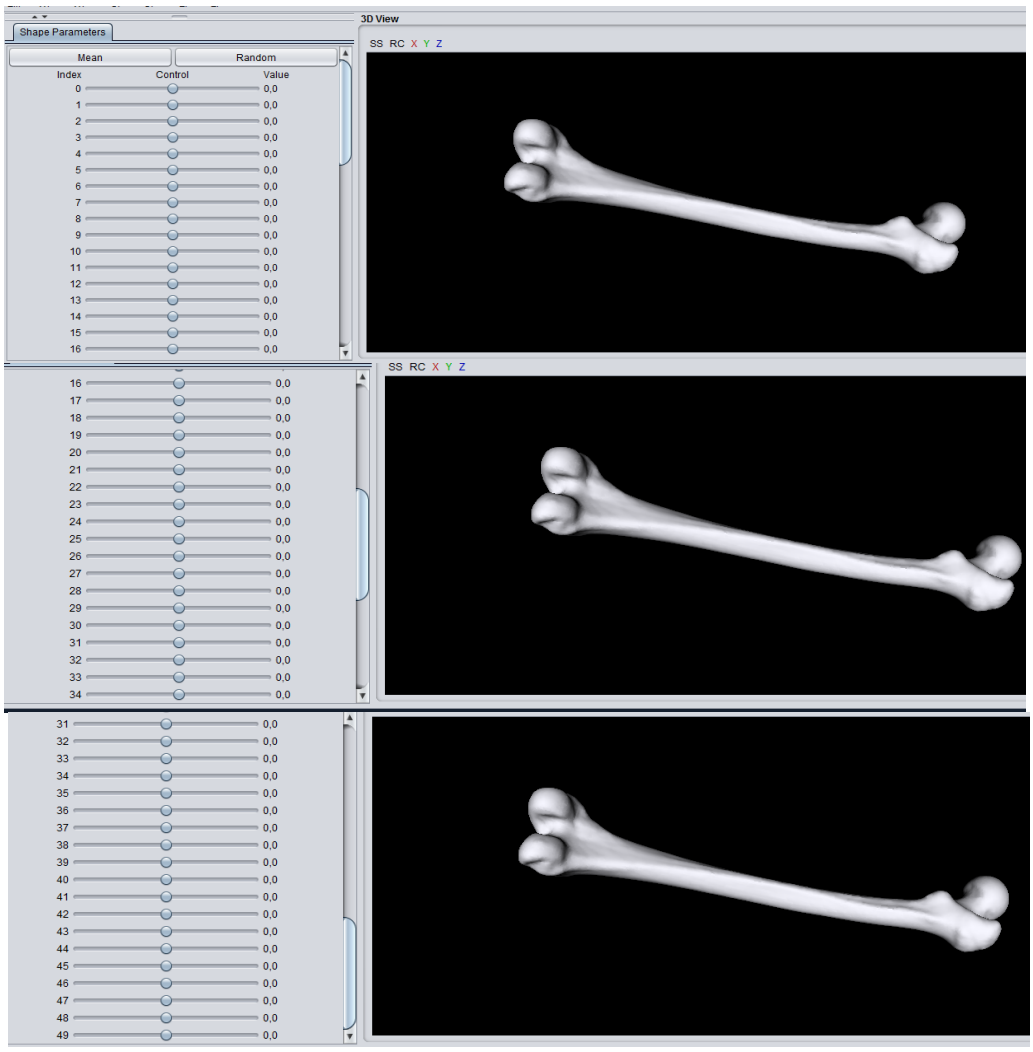

Figure 3. 43: *The femur built model.*

As our dataset contains 50 elements we have 50 parameters since we've used PCA method to build it. By changing the coefficients we could generate new shapes (samples) as the following figures illustrate:

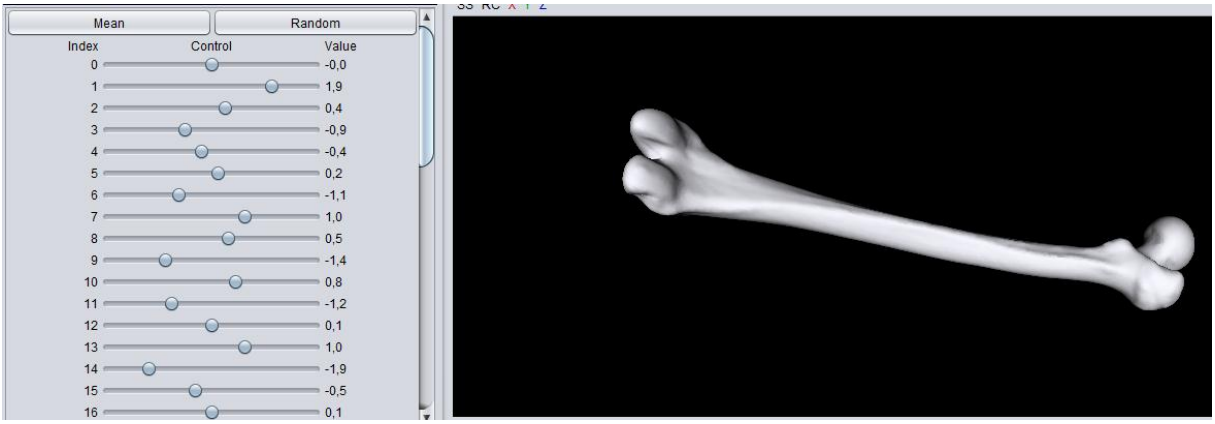

Figure 3. 44: *The first generated femur.*

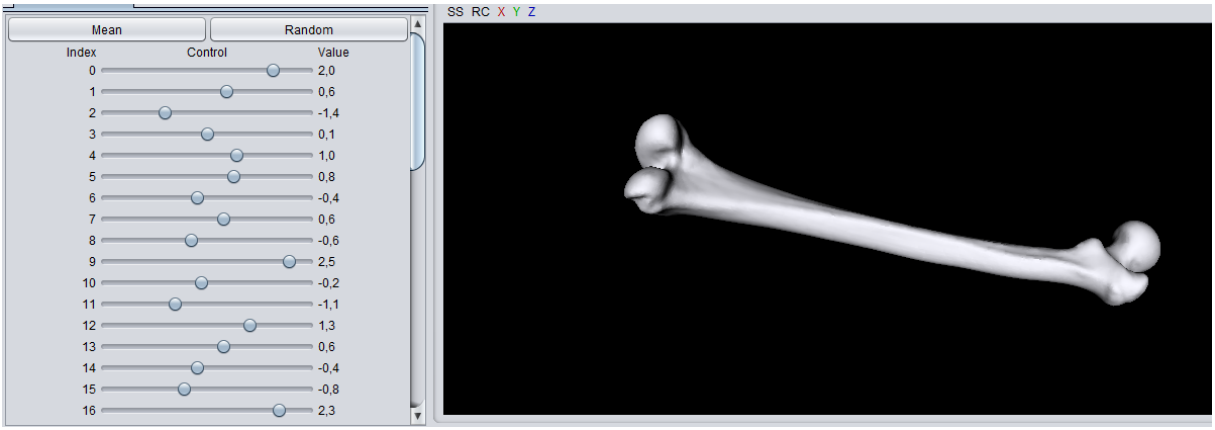

Figure 3. 45: *The second generated femur.*

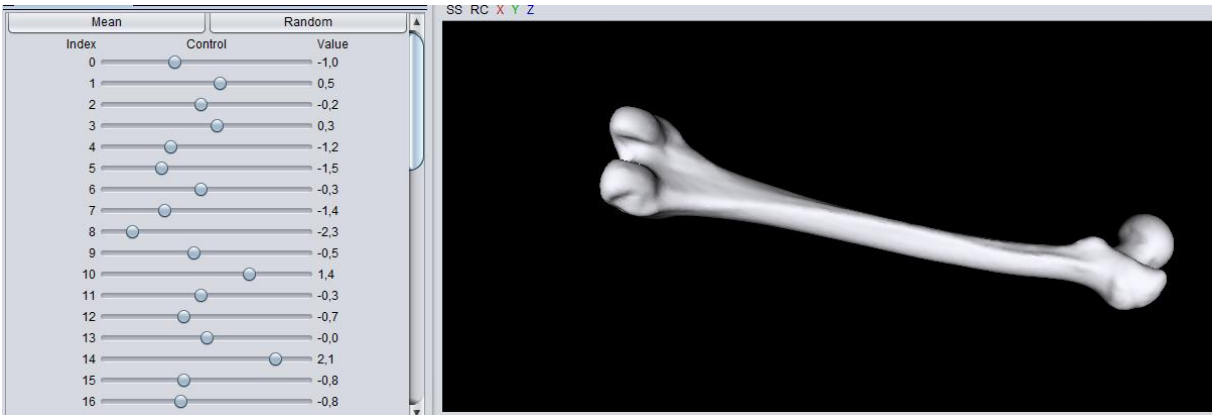

Figure 3. 46: *The Third generated femur.*

3.5 Summary

This chapter presents an experimental part, we have tested and applied the concepts learned in theory by using some predefined functions in the platform inside our program to build two different shape models: face and femur; For building femur model part it was our contribution to the SICAS (Swiss Institute for Computer Assisted Surgery) Medical Image Repository [9] team.

Conclusion and Future work

Statistical shape models have become a widely used tool in computer vision and medical image analysis where they are of considerable interest when studying shape variations in anatomical shapes.

This project introduced some mathematical concepts and applied some algorithms to model and analyze statistical shape models through the following steps:

- Locating the shape samples in a 3D triangular meshes format as a training set to build the shape model required.
- The task of aligning the samples to a specified reference.
- The task of establishing correspondence between shapes and images is known as the registration problem and it is one of the fundamental problems of computer vision.
- Once the shapes have been brought into correspondence, a statistical shape model can be built.

The obtained results are satisfactory for both experiments since we have built both models the face model and the femur model successfully but the issue that can be considered as an obstacle is the availability and the access to the dataset that must be given as triangular meshes (3D) which is a very complex process that requires a fully equipped laboratory with optical scanners and well adjust lighting projectors and other materials which make it very hard and expensive. For this reason we only experimented on the face and the femur. Where the face training set was obtained from the Basel University that provided the Basel face model whereas the femur‘s was from the SMIR (SICAS Medical Image Repository).

Our shape modeling program can be used to model other objects, for instance other organs like the heart, liver or kidney. But it is unclear how it can be applied to objects for which the concept of correspondence does not apply so easily, like vascular trees or the bronchial tree of the lungs Moreover, statistical shape models also provide a very useful means for surgical reconstruction of missing, malformed or other pathological anatomical structures. Patient-specific, yet objective surgical reconstruction proposals based on such models have been established as a planning criterion for complex surgical interventions, like individual implant or transplant design.

Lists of Figures

Appendix A

The Cholesky decomposition or Cholesky factorization is a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose .

The Cholesky decomposition of a Hermitian positive-definite matrix **A** is a decomposition of the form $\mathbf{A} = \mathbf{L}\mathbf{L}^*$, Where **L** is a lower triangular matrix with real and positive diagonal entries, and **L** denotes the conjugate transpose of **L**. Every Hermitian positive-definite matrix (and thus also every real-valued symmetric positive-definite matrix) has a unique Cholesky decomposition.

If the matrix **A** is Hermitian and positive semi-definite, then it still has a decomposition of the form **A** = **LL*** if the diagonal entries of **L** are allowed to be zero.

When **A** has real entries, **L** has real entries as well and the factorization may be written as

$\mathbf{A} = \mathbf{L}\mathbf{L}^T$

The Cholesky decomposition is unique when **A** is positive definite; there is only one lower triangular matrix **L** with strictly positive diagonal entries such that **A** = **LL***. However, the decomposition need not be unique when **A** is positive semi definite.

The converse holds trivially: if **A** can be written as $\mathbf{L}\mathbf{L}^*$ for some invertible **L**, lower triangular or otherwise, then **A** is Hermitian and positive definite.[8]

Appendix B [8]

In statistics, Procrustes analysis is a form of statistical shape analysis used to analyse the distribution of a set of shapes. The name Procrustes (Greek: Προκρούστης) refers to a bandit from Greek mythology who made his victims fit his bed either by stretching their limbs or cutting them off.The shape of an object can be considered as a member of an equivalence class formed by removing the translational, rotational and uniform scaling components.

Translation

For example, translational components can be removed from an object by translating the object so that the mean of all the object's points (i.e. its centroid) lies at the origin.

Mathematically: take k points in two dimensions, say $((x_1 , y_1), (x_2 , y_2), \dots \dots, (x_k , y_k)$

The mean of these points is $(\bar{x}, \bar{y})$where

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_k}{k} \quad , \quad \bar{y} = \frac{y_1 + y_2 + \cdots + y_k}{k} \qquad \text{(B, 1)}$$

Now translate these points so that their mean is translated to the origin

$(x, y) \rightarrow (x - \bar{x}, y - \bar{y})$ giving the point $(x_1 - \bar{x}, y_1 - \bar{y}), \ldots$.

Uniform scaling

Likewise, the scale component can be removed by scaling the object so that the root mean square distance (*RMSD*) from the points to the translated origin is 1. This RMSD is a statistical measure of the object's scale or size:

$$s = \sqrt{\frac{(x_1 - \bar{x})^2 + (y_1 - \bar{y})^2 + \cdots}{k}} \qquad \text{(B.2)}$$

The scale becomes 1 when the point coordinates are divided by the object's initial scale:

$((x_1 - \bar{x})/s \, , (y_1 - \bar{y})/s)$.(B,3)

Notice that other methods for defining and removing the scale are sometimes used in the literature.

Rotation

Removing the rotational component is more complex, as a standard reference orientation is not always available. Consider two objects composed of the same number of points with scale and translation removed. Let the points of these be $((x_1, y_1), \ldots)$,$((w_1, z_1), \ldots)$. One of these objects can be used to provide a reference orientation. Fix the reference object and rotate the other around the origin, until you find an optimum angle of rotation θ such that the sum of the squared distances (*SSD*) between the corresponding points is minimized (an example of least squares technique).A rotation by angle θ gives

$$(u_1, v_1) = (\cos\theta w_1 \, - \sin\theta z_1 \, , \sin\theta w_1 + \cos\theta z_1) \qquad \text{(B,4)}$$

Where (u, v) are the coordinates of a rotated point. Taking the derivative of

$(u_1 - x_1)^2 + (v_1 - y_1)^2 + \cdots$ With respect to θ and solving for θ when the derivative is zero gives

$$\theta = \tan^{-1}\left(\frac{\sum_{i=1}^{k}(w_i y_i - z_i x_i)}{\sum_{i=1}^{k}(w_i x_i + z_i y_i)}\right) \qquad \text{(B, 5)}$$

When the object is three-dimensional, the optimum rotation is represented by a 3-by-3 rotation matrix *R*, rather than a simple angle, and in this case singular value decomposition can be used to find the optimum value for *R*

Shape comparison

The difference between the shape of two objects can be evaluated only after "superimposing" the two objects by translating, scaling and optimally rotating them as explained above. The square root of the above mentioned SSD between corresponding points can be used as a statistical measure of this difference in shape:

$$d = \sqrt{(u_1 - x_1)^2 + (v_1 - y_1)^2 + \cdots} \qquad \text{(B, 6)}$$

This measure is often called **Procrustes distance**. Notice that other more complex definitions of Procrustes distance and other measures of "shape difference" are sometimes used in the literature.

Superimposing a set of shapes

We showed how to superimpose two shapes. The same method can be applied to superimpose a set of three or more shapes, as far as the above mentioned reference orientation is used for all of them. However, Generalized Procrustes analysis provides a better method to achieve this goal.

Singular Value Decomposition (**SVD**) [9]

In linear algebra, the singular value decomposition (SVD) is a factorization of a real or complex matrix. It is the generalization of the Eigen-decomposition of a positive semi-definite normal matrix (for example, a symmetric matrix with positive eigenvalues) to any m matrix via an extension of polar decomposition. It has many useful applications in signal processing and statistics.

Formally, the singular value decomposition of a real or complex matrix is a factorization of the form, where is a real or complex unitary matrix, is a rectangular diagonal matrix with non-negative real numbers on the diagonal, and is a real or complex unitary matrix. The diagonal entries φ_i of Σ are known as the singular values of M. The columns of U and the columns of V are called the left-singular vectors and right-singular vectors of M, respectively.

The singular value decomposition can be computed using the following observations:

- The left-singular vectors of M are a set of orthonormal eigenvectors of MM^*.
- The right-singular vectors of M are a set of orthonormal eigenvectors of M^*M.
- The non-zero singular values of M (found on the diagonal entries of Σ) are the square roots of the non-zero eigenvalues of both M^*M and MM^*.

References and Bibliography

[1] https://github.com/unibas-gravis/scalismo.

[2] Marcel Lüthi - University of Basel " Statistical Image And Shape Models " ,2014.

[3] A. Papoulis,"Probability, Random Variables and Stochastic Processes,"McGraw-Hill, 1965.

[4] http://www.cmlab.csie.ntu.edu.tw/~cyy/learning/papers/PCA_ASM.pdf.

[5] Steinke, Florian, and Bernhard Schölkopf.'Kernels, regularization and differential equations'. *Pattern Recognition*, vol. 41, 2008.

[6]https://cims.nyu.edu/~kiryl/Calculus/Section_4.5Optimization%20Problems/Optimization_Problems.pdf.

[7]Tam, Gary K.L., et al. 'Registration of 3D point clouds and meshes: a survey from rigid to nonrigid'. *IEEE Transactions on visualization and computer graphics*, 2013.

[8] https://en.wikipedia.org .

[10] https://www.smir.ch/.

Chapter 1

Statistical Shape Modeling

This chapter introduces the theoretical details of the different concepts that play a key role in the algorithms used to build a statistical shape models.

Chapter 2

Theoretical Concept of Building Statistical Shape Models

This chapter presents in details the fundamental methods that are a for building the statistical shape models.

Chapter 3

Experimental Part of Building Statistical Shape Model:
Face and Femur

This chapter covers the implemented programs of building statistical face and femur model with different obtained results.