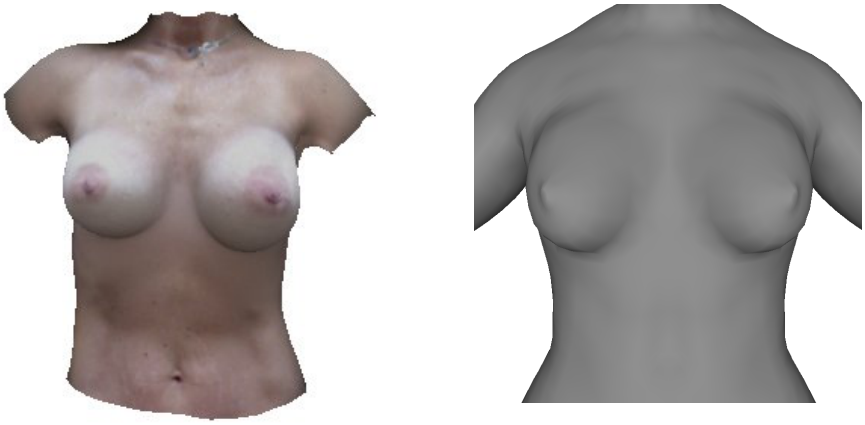
Inferring Human Body Parts and Correlations from Images, Point Clouds and Meshes



David Haldimann

Semester Thesis
May 2018

Endri Dibra
Prof. Dr. Markus Gross

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

cgl
computer graphics laboratory

Abstract

In this thesis, a mapping between parameters, defining the upper torso of a female, is provided, using linear and non-linear methods and compared among one another. Furthermore, a parametric model of the human body is created, based on meshes, generated by a free character modelling tool called MakeHuman and compared to a parametric model, based on real data. The parametric models are created using principle component analysis. Different representations of the data, used for principle component analysis, are tested and evaluated.

Zusammenfassung

In dieser Arbeit wird eine Relation zwischen Parametern, die den weiblichen Oberkörper definieren, beschrieben, indem lineare und nicht-lineare Methoden verwendet und untereinander verglichen werden. Zusätzlich wird, anhand von Gittern, ein parametrisches Modell des menschlichen Körpers kreiert. Dafür wird eine kostenlose Charakter modellierungs Software verwendet, namens MakeHuman. Dieses Modell wird mit einem Modell, basierend auf echten Daten, verglichen. Um die parametrischen Modelle zu kreieren, wird Hauptkomponentenanalyse angewendet. Verschiedene Representationen der Daten, die für den Prozess der Hauptkomponentenanalyse benötigt werden, werden getestet und evaluiert.

Contents

List of Figures

List of Tables

1

Introduction

The price for breast enhancement surgery was estimated to be $3718 in 2017 according to the American Society of Plastic Surgery[1]. Next to the financial aspects, there are also risks connected to undergoing surgery as well as not knowing exactly what the result will look like. Before commiting to this kind of operation, it should be possible to generate a preview of the outcome from a few images. This thesis aims to design a method that is able to predict a 3D model of the outcome by learning a mapping between parametric models. Additionally, the idea is explored, if it is possible to generate a parametric model from a character modelling software.

1.1. Parametric Model

A previous implementation by Biland [Bil17] was used to create parametric models. A parametric model can describe all data that went into the model with its parameters. For example, the physical appearance of a person can be roughly described by their height, skin tone and hair color, where these three are the parameters of this parametric model. This is, of course, only an approximation as the accuracy of the description of the person would increase with more parameters. It is also possible, that one parameter influences multiple features. In the previous example, when the height of a person is raised, the length of the arms are also proportionally increased.

[1] https://www.plasticsurgery.org/cosmetic-procedures/breast-augmentation/cost

1.2. Mapping

A mapping between sets associates each element in the first set with one or more elements of the second set. An example for a simple mapping could be the numbers one to twenty-six as the first set and the letters of the alphabet as the second. In the case of two parametric models, the goal is to find a mapping that describes the relationship between the parameters of the first and second model.
This mapping can either be linear or non-linear. The difference between a linearity and a non-linearity can be described with a simple example. The time it takes to drive $10km$ in a car at $10\frac{km}{h}$ is $1h$. If the distance to drive is doubled to $20km$, so will the time it takes. That is, because distance to drive and time it requires are in a linear relationship. On the other hand, given that the braking distance while travelling with $10\frac{km}{h}$ is $1m$, the braking distance while travelling with $20\frac{km}{h}$ is $4m$. This is due to the fact, that speed and braking distance are related in a quadratic, non-linear, manner.

1.3. Applications

The applications for this method could be plenty. While the most obvious use for this implementation would be plastic surgery - creating 3D models from images of possible breast enhancements, it could also be used in oncology, to show affected patients what their body might look like after a mastectomy and even further, after a possible reconstruction. This approach could also be used in an adapted version for commercial use, such as tailor fitting clothes by using an image of the customer.

2

Related Work

In the field of modelling the human body, early on, a lot of work was done concerning the face. Jeng et al. [JLH+98] proposed a geometric face model to localize the face in an image and detect facial features efficiently. The reason for the need of a functioning model for faces may have been the upcoming importance of facial recognition.
Parametric models have also been applied to reshaping the human body in images using semantic parameters. [ZFL+10] Their approach included reshaping a morphable 3D model and applying the reshaping effects using an image warping approach. Other work was done by Allen et al. [ACP03] on parameterizing full body scans and using the parameters in a variety of applications. This method could be used to find the relations between parameters and features which could then, in turn, be modified. For example, the weight could be increased and the resulting mesh would increase the shape of the mesh in a meaningful way.
Work about specifically modelling the breast has been done by Galle et al. [GGC10]. They proposed a parameter space based on MRI scans to describe the shape of a human breast using principle component analysis. Ciechomski et al. [dHCCG+12] proposed a web approach to create 3D models from images, where participants uploaded images and selected keypoints. These 3D models can later be accessed by surgeons to visualize different types and sizes of implants.

3

Methods

In this chapter, the methods needed to create a parametric model and a mapping between parametric models are introduced and explained. Firstly, the data gathering and preprocessing are outlined. Multiple variants of the parametric model are discussed. Different learning approaches for the mapping are reviewed. Lastly, a character editor is presented that is used to generate data for a parametric model.

3.1. Data Acquisition and Preprocessing

It would be necessary to have a large data set of images of women before and after breast enhancement surgery, where the patients pose topless. It is very unlikely though, that such a database exists, due to the fact that having breast surgery is a very personal topic and people generally don't enjoy posing naked. Therefore, the images used were downloaded from a website[1] that offered to simulate various plastic surgical procedures including breast enhancement. For each user, a 3D model of their torso was displayed side by side with different enhancements varying in size. An screenshot of the interface can be seen in figure 3.1.

Each model was made up of a sequence of 24 images displaying the torso from different angles. This dataset fit the requirements nicely, as images are available for *before* and *after*, except the *after* is generated and based on their model. Additionally, each after image sequence had a short label, usually describing how much silicone was added, that was also saved for further evaluation. In total 2'937 examples were retrieved and preprocessed. This dataset included images from 748 subjects of which each one was comprised of one *before* and at least one *after* image sequence.

[1] https://my.crisalix.com/

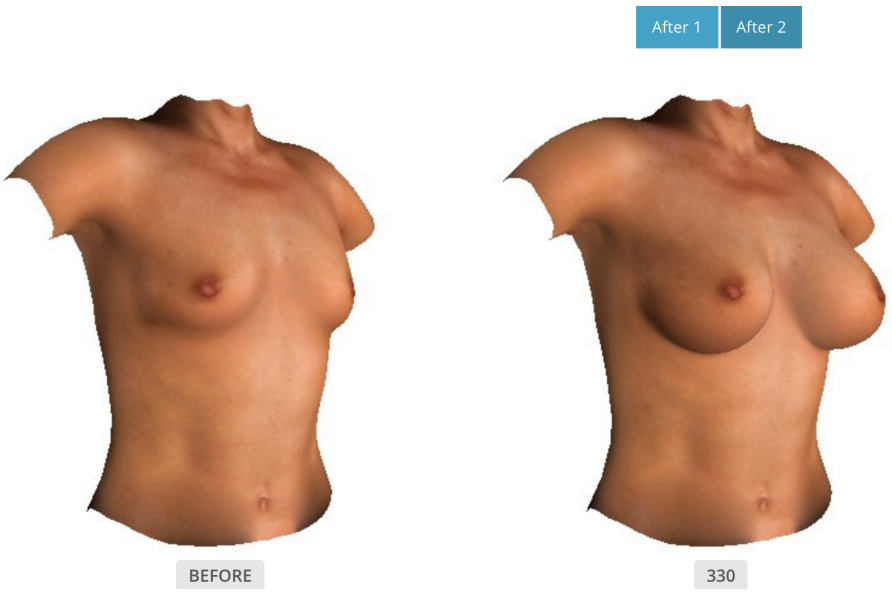


Figure 3.1.: *Example screenshot of the visualization on the website of Crisalix.*

In a next step, these image sequences needed to be transformed into point clouds. This was done, using a general-purpose Structure-from-Motion (SfM) [SF16] and Multi-View Stereo (MVS) [SZPF16] pipeline called COLMAP. This generated point clouds, spanning from 5'000 to 15'000 points. Some of the images needed to be discarded, due to the fact that SfM created a point cloud with less than 1'000 points or the point clouds had holes, such that certain areas had no points and were not defined at all. The remaining point clouds were cleaned using a C++ implementation by Biland [Bil17] that removed white points around the borders.

The mapping required to have one set of point clouds of *before* examples and the corresponding[2] *after* examples. Therefore, the data was split into sets of *before* and *after* point clouds. As all of the *after* point clouds were based on various breast enhancements, all the corresponding labels were processed and analyzed. This resulted in a table of terms sorted by the frequency of the term. The 9 most frequent terms can be found in table 3.1.

Term	**R:**	**L:**	**LSC**	**73**	**cc**	**LSA**	**350**	**Custom**	**300**
Frequency	205	205	100	85	54	73	73	70	60

Table 3.1.: *The 9 most frequent terms can be seen in the upper row. The corresponding frequency is stated below.*

The first few terms are not informative, as they don't describe the breast themselves. The terms "LSC" and "73" are descriptions of a certain implant [3] and "cc" is a measure of volume. The first meaningful term is "350", which means that an implant with a volume of 350ml was inserted. Therefore, only the *after* examples that were labelled "350" were included, to create a better mapping. After removing some of the point clouds that were not reconstructed well enough,

[2]Corresponding meaning, based on the same subject.
[3]http://www.simplybreastimplants.com/breast_implant_sizes/sebbin_firm_high_profile.html

this resulted in 57 examples in the *before* and 57 in the *after* set. All of these point clouds were further processed in a MATLAB implementation by Biland [Bil17] to generate mesh files using a method called nonrigid iterative closest point (NRICP) algorithm. Given a point cloud and a base mesh, NRICP iteratively deforms the base mesh to match the points in the point cloud. The parameters used in this thesis can be found in the appendix in table A.1 and the template mesh used was the Audrey mesh. This is one of the meshes that was also used by Biland.

3.1.1. Alignment

The meshes are aligned in the procedure of the MATLAB implementation by Biland [Bil17]. Due to the alignment being done according to handpicked points, there are certain inaccuracies that can occur and lead to slight rotations of the meshes. To prevent the mapping from learning rotations, the *before* and *after* should be aligned pairwise. This was achieved by using an implementation of Horn's method [Hor87]. Given two sets of vertices, Horn's method computes the translation, rotation and, if desired, scale change from one set to the other. As it is expected, that for the same subject, exclusively points defining the breasts should vary from *before* to *after*, only a subset of vertices should be used. The points used in the alignment can be seen in figure 3.2.

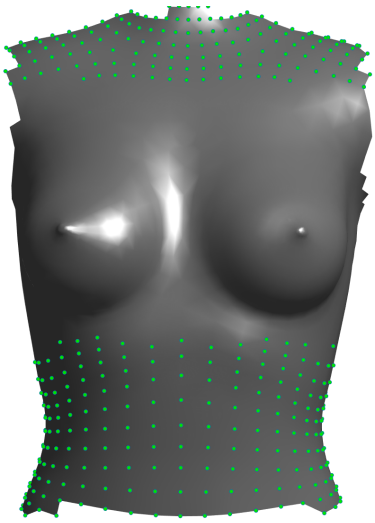


Figure 3.2.: *The mesh is depicted as the grey surface. The green points are used for the alignment.*

3.2. Parametric Model from Meshes

In the next section, it is described how a parametric model is obtained using principle component analysis (PCA). This first method is the standard method, that is used in this thesis. In the following subsections, two different variations are described.

3.2.1. Points

In this subsection, the general procedure to create a parametric model based on points, is described. Given n meshes $m_i \in \mathbb{R}^{k\times 3}$, where k describes the amount of vertices m has, each mesh needs to be transformed to be of shape $\mathbb{R}^{1\times 3k}$. Then, all transformed meshes are stacked into a matrix $M \in \mathbb{R}^{n\times 3k}$. As the differences over each column isn't significant, the mean $\bar{m}$ of the matrix M is subtracted from each row of M.

$$\mathbf{A} := \begin{bmatrix} m'_1 - \bar{m} \\ m'_2 - \bar{m} \\ \vdots \\ m'_n - \bar{m} \end{bmatrix} \in \mathbb{R}^{n\times 3k} \tag{3.1}$$

Next, PCA is run with matrix A as the input. PCA is able to reduce the dimensionality of the data, while retaining most of the information from the initial data set. This is achieved, by finding orthogonal basis vectors, where the first basis vector is responsible for the largest variance in the data. The second basis vector needs to be orthogonal to the first and is responsible for the second largest variance of the data. This holds for each following basis vector. These basis vectors in PCA are also known as principle components.

The output of the PCA function is:

$$\mathbf{coeff} := \begin{bmatrix} c_1 & c_2 & \cdots & c_{n-1} \end{bmatrix} \in \mathbb{R}^{3k\times n-1} \tag{3.2}$$

$$\mathbf{score} := \begin{bmatrix} s_1 & s_2 & \cdots & s_{n-1} \end{bmatrix} \in \mathbb{R}^{n\times n-1} \tag{3.3}$$

where coefficient c_i is the i-th principle component and score s_i is the i-th parameter vector corresponding to the i-th input. Therefore, the input data can be reproduced by computing $\mathbf{A} = \mathbf{score} \times \mathbf{coeff}^{\mathbf{T}}$. Instead of using all $n-1$ coefficients, it is possible to only use the first $\mathbf{q}$ coefficients resulting in an approximation of the space. The parameters $\mathbf{p_{q,new}}$ for a new input mesh m_{new} can be easily computed for $\mathbf{q}$ coefficients, by first reshaping the mesh to be of the form $\mathbb{R}^{3k\times 1}$ and subtracting the mean $\bar{m}$. This is done the same way as above, adding an additional step to transpose. Then, the pseudoinverse of $\mathbf{coeff_{(q)}}$ is multiplied from the left to compute the corresponding parameters $\mathbf{p_{q,new}}$:

$$\mathbf{p_{q,new}} = \mathbf{coeff}^{+}_{\mathbf{(q)}} \cdot (m'_{new} - \bar{m})^T \text{ where } \mathbf{coeff_{(q)}} := \begin{bmatrix} c_1 & c_2 & \cdots & c_q \end{bmatrix} \in \mathbb{R}^{3k\times q} \tag{3.4}$$

The number of principle components used to approximate the space was reduced to 30 in this thesis. In this example, the parametric model is based on the vertices of the mesh. In the next subsections, two other variants are explored and explained.

3.2.2. Face Deformations

Instead of defining a mesh by its vertices, it is possible to describe the mesh by the deformations of the faces. This is done by defining one source mesh and computing how each face is deformed in relation to the source mesh. One way to quantify a deformation of a face, was described by Sumner [SP04], where the idea was to transfer triangle deformations between similar meshes. First, a new vertex needs to be computed, such that an affine transformation can be determined. The fourth vertex is defined as follows:

$$\mathbf{v_4} = \mathbf{v_1} + (\mathbf{v_2} - \mathbf{v_1}) \times (\mathbf{v_3} - \mathbf{v_1}) / \sqrt{|(\mathbf{v_2} - \mathbf{v_1}) \times (\mathbf{v_3} - \mathbf{v_1})|} \tag{3.5}$$

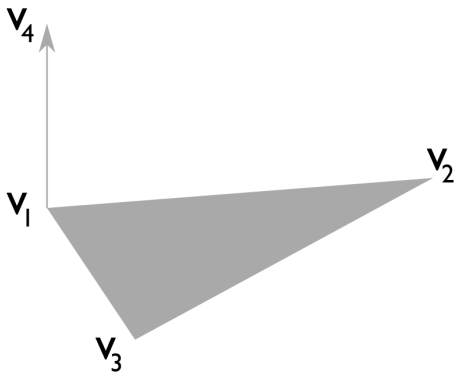


Figure 3.3.: *A vertex is added to the triangle that lies in the direction of the normal.*

The cross product is scaled by the reciprocal of the square root of its length, to keep the distance proportional to the length of the triangle edges. See figure 3.3 for a visualization of adding a vertex on the normal. Given both triangles with one additional vertex, the following equations can be posed:

$$\mathbf{Q}\mathbf{v_i} + \mathbf{d} = \tilde{\mathbf{v}}_\mathbf{i}, \mathbf{i} \in \mathbf{1} \ldots \mathbf{4} \tag{3.6}$$

where $\mathbf{Q} \in \mathbb{R}^{3\times3}$ and translation vector $\mathbf{d}$ describe the affine transformation. By subtracting the first equation from the following three equations and rewriting the resulting system in matrix form, the problem can be defined as $\mathbf{QV} = \tilde{\mathbf{V}}$ where

$$\mathbf{V} = \begin{bmatrix} \mathbf{v_2} - \mathbf{v_1} & \mathbf{v_2} - \mathbf{v_1} & \mathbf{v_2} - \mathbf{v_1} \end{bmatrix} \tag{3.7}$$

$$\tilde{\mathbf{V}} = \begin{bmatrix} \mathbf{v_2} - \mathbf{v_1} & \mathbf{v_2} - \mathbf{v_1} & \mathbf{v_2} - \mathbf{v_1} \end{bmatrix} \tag{3.8}$$

The closed form expression for $\mathbf{Q}$ is defined as

$$\mathbf{Q} = \tilde{\mathbf{V}}\mathbf{V}^{-1} \tag{3.9}$$

To fully describe a mesh, this $\mathbf{Q}$ matrix needs to be computed for each face of the mesh. Finally, the mesh is represented by $\mathbf{D} \in \mathbb{R}^{3\times3h}$ where $\mathbf{h}$ is the number of triangles the mesh has. To be able to run PCA, each mesh needs to be reshaped to be of form $\mathbf{D'} \in \mathbb{R}^{1\times9h}$ and the rest of the process is the same as above in section 3.2.1.

3.2.3. Point Normals

This method is very much similar to the one described in section 3.2.1. In addition to the vertices of the mesh, one vertex per face is computed as described in equation 3.5 and added to the list of vertices of the mesh. Therefore, the mesh matrix will be of form $\mathbb{R}^{k+h\times 3}$, where $\mathbf{k}$ is the number of original vertices and $\mathbf{h}$ is the number of triangles of the mesh.

3.3. Mapping

The following segment describes how a mapping can be computed in a linear or a non-linear way. The data available is the parameters returned by both the parametric models for the *before* and *after* examples. The matrices of the data are

$$\mathbf{P_{before}} = \begin{bmatrix} \mathbf{p_{b,1}} \\ \mathbf{p_{b,2}} \\ \vdots \\ \mathbf{p_{b,n}} \end{bmatrix} \in \mathbb{R}^{n\times n-1}, \mathbf{P_{after}} = \begin{bmatrix} \mathbf{p_{a,1}} \\ \mathbf{p_{a,2}} \\ \vdots \\ \mathbf{p_{a,n}} \end{bmatrix} \in \mathbb{R}^{n\times n-1} \tag{3.10}$$

where each parameter pair $(\mathbf{p_{b,i}}, \mathbf{p_{a,i}})\ \forall i$ is related, as the *after* was generated from the *before*.

3.3.1. Linear Method

The first method used, was the linear system solver by MATLAB (also known as backslash solver) that solved the equation

$$\mathbf{P_{before}} \cdot \mathbf{M_{linear}} = \mathbf{P_{after}} \text{ where } M_{linear} \in \mathbb{R}^{n-1\times n-1}. \tag{3.11}$$

It is clear, that an explicit linear transformation could be found for each parameter pair, but the goal is to have one mapping that works for each parameter pair. This is also known as linear regression. One basic example in linear regression is the task of fitting a line to a point cloud. As all the points of the point cloud do not lie on one line, it isn't possible to define a straight line. Therefore, a line is sought after, such that all points are as close as possible on average. This is depicted in figure 3.4.

3.3.2. Non-Linear Variants

This section deals with non-linear methods to solve the regression problem described above. All the methods used are part of the scikit-learn library [PVG+11]. It is common in a machine learning setting to split data into two sets. One set is used to train the model and the other is used to evaluate how well the method does on data, that was not part of the first set. Both data sets should be drawn from the same distribution. This is necessary because the model shouldn't be penalized for performing badly on problems it didn't encounter in the training phase.

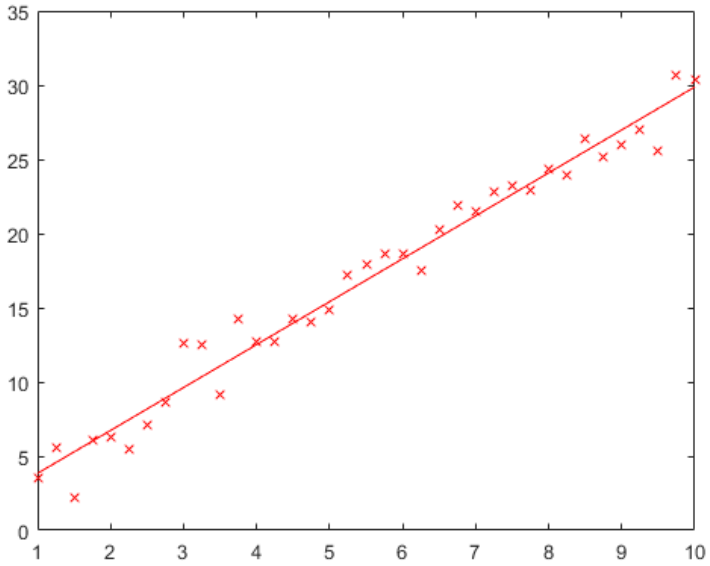


Figure 3.4.: *Fitting a line to a point cloud using linear least-squares.*

During the training phase, it is standard to run grid search. Grid search varies the parameters of the model to find the best set of parameters for a specific task. It needs to be considered, that grid search can lead to overfitting. This occurs when the error on the training set is reduced at the cost of increasing the test error. This can be avoided by adding regularization terms or by running grid search along with cross-validation.

Decision Tree

One category of machine learning approaches is based on so called decision trees (DT). A decision tree uses a tree-like approach, where nodes of the tree are defined to split the data set into subsets corresponding to groups. As an example, the goal is to have a model that can predict the weight of a person, based on height, daily activity and daily calorie intake as features. Using a decision tree, the value of height could split the data set, as taller people tend to be heavier than small people. This is done over all features until the complete data set is divided into smaller subsets. Leafs make up the end of a decision tree. Each leaf has a certain value connected to it, in the case mentioned before it would be a certain weight range or value.
When the weight of a new person would need to be predicted, one would follow the tree from top to bottom until a leaf was met. The corresponding weight value of the leaf would be equal to the prediction the model returns. An example of a decision tree can be seen in figure 3.5.

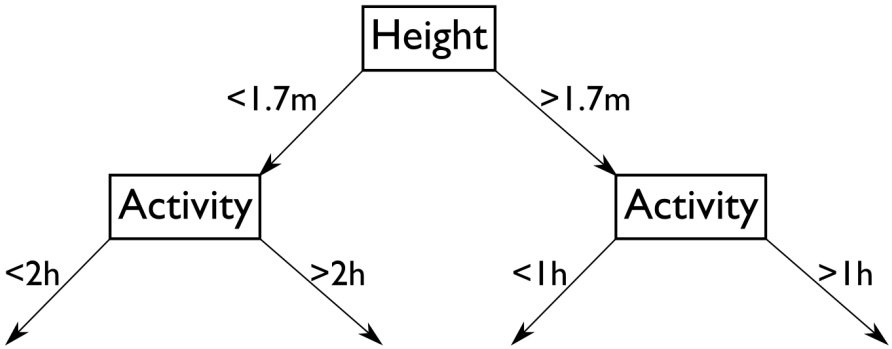


Figure 3.5.: *Example of a decision tree following example of section 3.3.2*

Random Forest

The random forest (RF) regressor is also a tree-like approach, similar to the decision tree. The main difference is, that the decision tree considers all features when splitting and thus can be prone to overfitting. Random forest chooses features at random and builds decision trees based on those features. This is done for multiple trees until all subtrees are combined.

Multilayer Perceptron

The multilayer perceptron (MLP) is a type of neural network. It uses a feedforward approach to propagate information through the network. MLP has at least three layers, where the first layer is the input layer and the last layer is the output layer. Each layer is made up of neurons, that receive information from the previous layer, apply a multiplication with a weight and an addition of a bias term followed by an activation function and pass the result on to the next layer. These activations can either be linear or non-linear. It doesn't make sense to have multiple intermediate layers with linear activation functions, as any combination of linear functions can be described by one linear function.
Each layer is made up of multiple neurons, each with their own weights and biases, and an activation function. All weights and biases are initialized at random. In the training phase, a technique known as backpropagation is applied. Given an example, input and outcome, the input data is passed through all the layers. The resulting output is compared to the actual outcome. Given the difference between the two values, weights and biases are modified to reduce the difference, starting from the output layer, moving back through the hidden layers, up to the input layer. A basic scheme of a MLP is shown in figure 3.6.

3.4. Parametric Model from Editor

In all of the previous sections, the parametric model was based off of real world data. In this section, the same procedure is done as in section 3.2.1, except that the meshes used for PCA

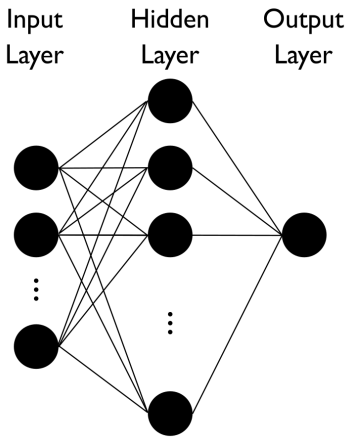


Figure 3.6.: *Scheme of a multilayer perceptron for regression.*

were generated by software called MakeHuman [4]. MakeHuman is a free and open source character modelling tool. It starts out with a gender neutral base mesh that can be modelled by varying sliders. These sliders apply linear interpolations to the base mesh between the extreme positions for a specific semantic parameter.
There are over 40 semantic parameters in MakeHuman and the semantic parameters are categorized by what areas of the body they modify. The graphical user interface of MakeHuman can be seen in figure 3.7. Using the scripting feature in MakeHuman, multiple meshes could be generated, exported and later imported into MATLAB.

3.5. Fitting parametric models to point clouds

Given a parametric model, it is possible to create a large diversity of possible outputs by varying the parameters. But finding the specific parameters for a new point cloud is a non-linear optimization problem. Therefore, a C++ implementation by Biland [Bil17], using an open source C++ library for optimization problems called the Ceres solver [AMO], was slightly modified and used. This implementation allows to fit a parametric model to a point cloud given a few keypoints of the point cloud. The slight modification enables a list of indices to be used as candidates for the correspondences required.
In a first step, the keypoints are used to find the parameters of translation, rotation and scaling. Afterwards, a select number of points are chosen as correspondences for the model and, using a nearest neighbour approach, the closest points in the point cloud are found. This implementation tries to minimize the distance between all the correspondences iteratively, by varying the parameters of the model. Additionally, constraints are applied to the values the parameters are allowed to take. This prevents one parameter from becoming too large and also possibly forces the model to use multiple parameters rather than to rely on a single parameter. After a certain number of iterations has been reached, a certain threshold in accuracy is passed or the change in parameters is vanishingly small, the resulting mesh and parameters are returned.

[4] http://www.makehumancommunity.org/

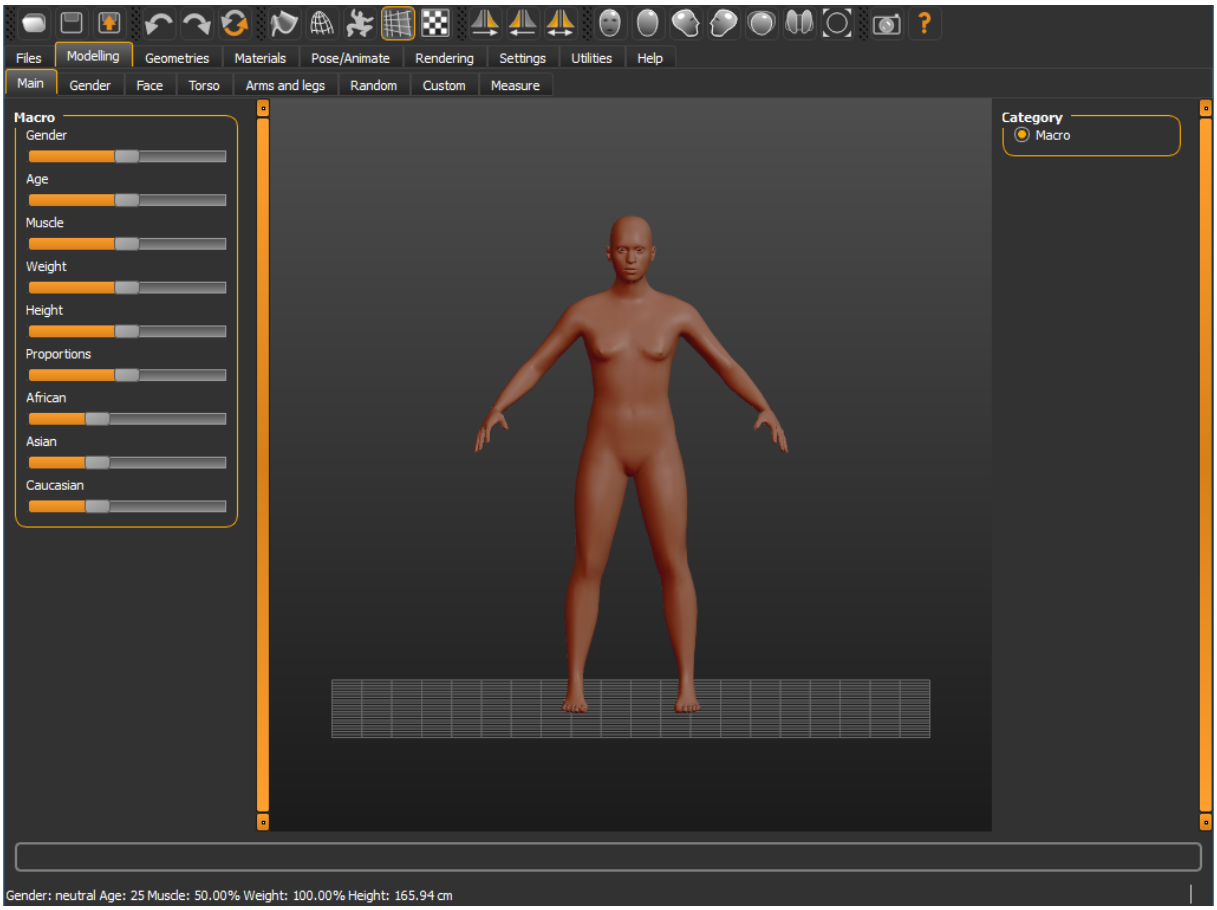


Figure 3.7.: *Graphical user interface of MakeHuman. On the left some of the semantic parameters can be seen.*

4

Results

In the next section, two error metrics are briefly described, followed by evaluations of each experiment. At the beginning of each section a small description of the problem setting is given, followed by the results and discussion.

4.1. Error Metrics

For this thesis two error metrics were considered:

- **Point Distance:** Given two meshes with equal topology, meaning both meshes have the same amount of vertices and layout, the error computed is based on the distance between corresponding points. In general, the mean distance is computed over all point correspondences.
- **Face orientations:** The idea behind this method is based on the $\mathbf{Q}$ matrix mentioned in section 3.2.2. Given two topologically equal meshes, the $\mathbf{Q}$ matrix is computed for all corresponding triangles. As it is of interest to investigate the difference in breast shapes, this error tries to quantify the variation of the $\mathbf{Q}$ matrices.

One drawback of the point distance method occurs, when the two meshes are not aligned properly. This can be solved by applying the same algorithm mentioned in section 3.1.1 before computing the error. One advantage of this method is the unit of the error, as it could be converted into a unit of distance, given that the distance between two points is known in real world lengths. This error can also be computed, when two meshes are not topologically equal, by first finding the correspondences between vertices.
One advantage of the face orientation method is the invariance of translation and global rotation. This means, that moving or rotating the mesh doesn't affect the error. On the other hand, the values of this error don't have a real world meaning and it is a lot harder to apply this method

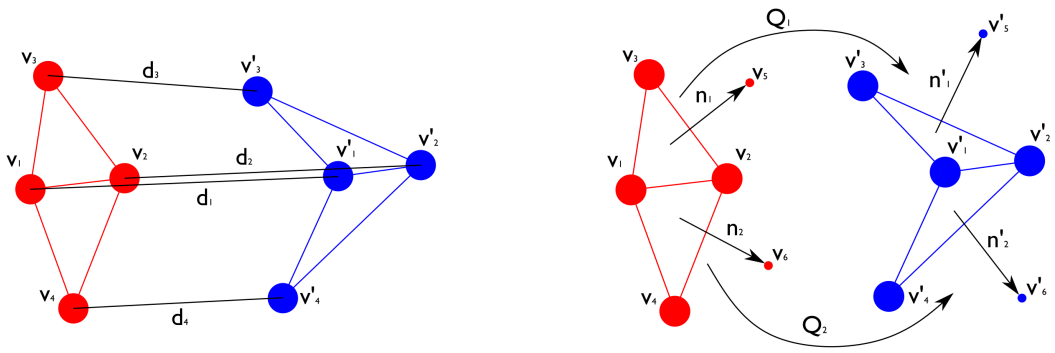


Figure 4.1.: *The left image depicts distances between corresponding points of the mesh. The right image shows a representation of the information used to compute the* **Q** *matrix.*

to topologically different meshes, as finding correspondences between triangles is a lot more challenging.
The methods are depicted in figure 4.1. Both methods have their advantages and disadvantages, but due to the fact that alignment isn't a problem, the error metric used in this thesis is the point distance method. Additionally, based on the MakeHuman model, the distance between the belly button and throat is approximated to be $40.6cm$. This is taken as an assuption for both meshes, such that all distances can be computed in centimetres.

4.2. Evaluation

In the following section, the evaluation methods that were used are explained. Optimally, when evaluating a prediction, some representation of the true outcome, often called the "ground truth", is given. This is useful to quantify how accurate the prediction was.

4.2.1. Mesh Point Error

For this method the comparison is done as explained in section 4.1. Before evaluating the point distances between a prediction and the ground truth, the meshes are aligned, as mentioned in section 3.1.1. Additionally, the mean point distance error is computed by dividing by the number of vertices in the mesh and also by multiplying with the approximated centimetre length of one unit in mesh space. This allows to transform all values into centimetres. Given a prediction of a mesh $M^{(p)}$ and the ground truth $M^{(g)}$ the problem can be stated as:

$$error = s_{cm} \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{3} (M_{i,j}^{(p)} - M_{i,j}^{(g)})^2} \tag{4.1}$$

where n is the number of vertices of the mesh, s_{cm} is the approximated length in centimetres equivalent to one unit distance in mesh space.

4.2.2. Error Heat Map

The idea of this method, is to generate a visualization of the error, indicating where the prediction failed to represent the ground truth the most. This can be accomplished by computing the error described in section 4.1, but instead of adding up the errors per mesh, the errors are stored separately for each vertex. This error per vertex is accumulated over multiple samples and the mean is computed. Given the k-th prediction of a mesh $M^{(p_k)}$ and the corresponding ground truth $M^{(g_k)}$ the problem can be described as:

$$e'_i = \frac{s_{cm}}{m} \sum_{k=1}^{m} \sqrt{\sum_{j=1}^{3} (M_{i,j}^{(p_k)} - M_{i,j}^{(g_k)})^2} \; \forall i = 1 \dots n \text{ with } e = \frac{e'}{max(e')} \tag{4.2}$$

where m is the number of samples, s_{cm} is the approximated length in centimetres equivalent to one unit distance in mesh space, e' is a vector of the accumulated errors over all samples and e is a vector of the errors scaled inversely by the largest entry in e'. This is done to visualize the error on a range, such that, for example, the error can be color coded, where dark red is equivalent to a large error and blue is a small error.

4.3. Input Data for PCA

In this experiment, three different variations of data inputs are tested for PCA, as described in section 3.2.1 and the following two subsections. Based on the resulting parameters of each model, a linear mapping is created and compared based on the error of the predictions.

4.3.1. Problem Setting

Given a data set comprised of 57 pairs of meshes, where one mesh is based on the *before* point cloud and the other is based on the *after* point cloud, both sets are split into equally-sized sets of 42 training pairs and 15 testing pairs. Each variation is trained on the training set and afterwards evaluated on the test set. The evaluation is done by computing a linear mapping over the parameters of each model. Then, the mean mesh point error over all 15 testing pairs is computed and an error heat map is generated.

4.3.2. Results

In the table 4.1, the time column is the amount of time each method needed to compute the parametric model. The training mean error is computed as mentioned in 4.2.1 for each case. The mean is computed over all cases. The same is done for the test mean error. The errors for the error heat maps were computed on the test set. In each subfigure of figure 4.2 the values that are listed above are the respective $\mathbf{max}(\mathbf{e}')$, as explained in 4.2.2. Additionally, one vertex on the border doesn't have a colored point, as the error wasn't being computed correctly for it.

	Time (s)	Training Mean Error (cm)	Test Mean Error (cm)
Point	0.06	0.40	1.01
Deformation	6.81	0.49	1.53
Points and Normals	2.92	0.39	1.01

Table 4.1.: *Results in terms of mean mesh point error are listed for the training and for the test set in centimetres. Additionally, the time each function took to compute the parametric model is listed in seconds.*

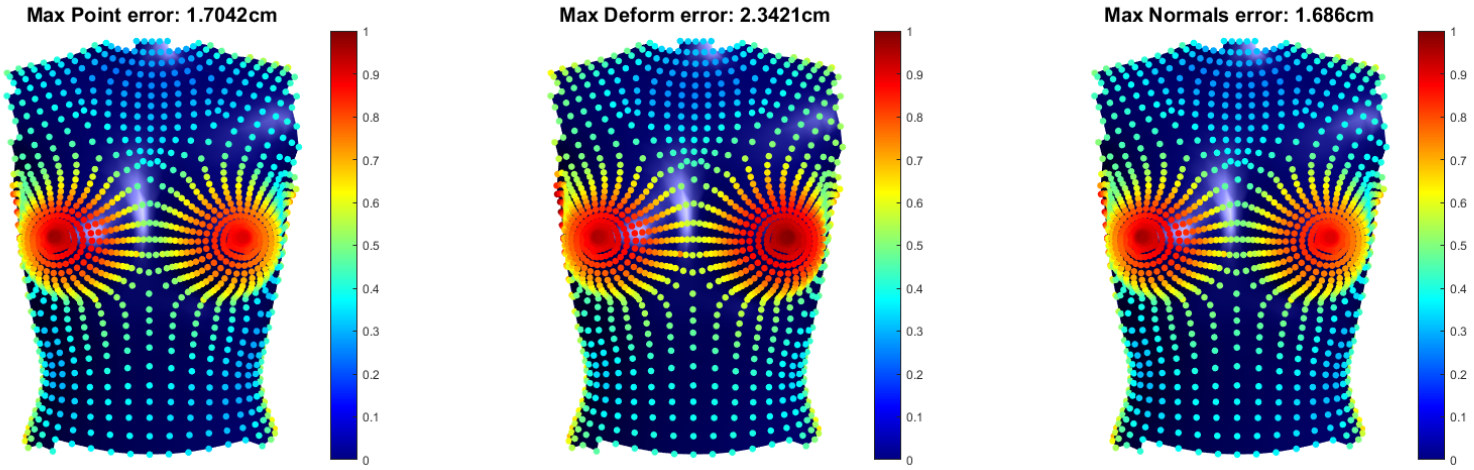


Figure 4.2.: *The left image shows the error heat map when using the points method. The middle one uses the deformation method. The right one uses the point normals method. The color of each point reflects the amount of the error. Red color indicates large error, blue small error.*

4.3.3. Discussion

The amount of time it takes to compute the parametric model, based on deformations, is much larger. This is due to the fact, that additional points on the normals of each triangle need to be computed, followed by computations of the Q matrix for each triangle. One can see, that the additional time to compute normals is also reflected in the point and normals method. Therefore, roughly $\mathbf{40}\%$ of the time is accounted for by the normal computation. The points method is by far the fastest method.
All three methods worsen going from training to test sets in terms of error, but that is expected. This is caused by the models having seen the training set before and are already familiar with those examples. Of the three methods, the point and normals method achieves the best results over test and training. The reason this occurs is due to the fact, that the normals carry information of the size of the triangles.
Figure 4.2 looks similar for all three cases. The error for the deformation method is higher than for the other two methods. This might be caused by the deformation method performing well in terms of triangle orientation, at the cost of some vertices being displaced. Around the right breast, the red area seems larger and darker than for the other two methods. The deformation method might have more difficulty placing the nipple correctly. Visually, no difference can be made out between the points method and the point normals method,

4.4. Learning Mapping

The next step was to investigate, if non-linear mappings could perform better than a linear mapping. Additionally, it is analyzed how the error behaves, when PCA is done on the complete set, instead of only the training set. The reason this is done, is because the mapping shouldn't be penalized or perform badly because of the parametric model. For each of the approaches, the best parameters need to be found first by running cross-validation grid search. Each of the learnt mappings can be applied to the test set and compared against the ground truth. The mean point error is computed to measure the error.

4.4.1. Problem Setting

The parameters for each of the 57 pair meshes are computed according to two parametric models. These will be referred to as the "complete" and "incomplete" model, where the "complete" model was based on the training and test data, whereas the "incomplete" model only received the training data. The non-linear approaches used, are a decision tree regressor, a random forest regressor, a multilayer perceptron with a rectified linear unit (ReLU) as its activation function and a multilayer perceptron with any other activation function except a ReLU.
For each approach the best parameters are evaluated by running 6-fold cross-validation grid search for either cases, where the error is not minimized in parameter space, but in mesh space. The method's varied parameters are listed in a table, that can be found in the appendix A.2. Finally, the training and test errors are computed based on the best parameters and a ratio between test and training error is calculated. Optimally, this ratio should be one, as the test error should

PCA Model	Method	Training Mean Error (cm)	Test Mean Error (cm)	Ratio Test/Train
Incomplete	**Linear**	0.40	1.01	2.52
	Random Forest	0.37	1.07	2.88
	Decision Tree	0.69	1.30	1.86
	MLP	0.84	1.02	1.21
	MLP ReLU	0.95	1.11	1.17
Complete	**Linear**	0.42	1.95	4.70
	Random Forest	0.42	1.07	2.55
	Decision Tree	0.68	1.19	1.74
	MLP	0.83	1.06	1.27
	MLP ReLU	0.88	1.06	1.20

Table 4.2.: *This tables shows the mean errors computed on the training and test set in centimetres. The first five rows show the results for the "incomplete" model, the last five for the "complete" model. The first two columns are the training and test error, followed by the ratio of test to training error.*

be equally low as the training error.

4.4.2. Results

The results for the different cases can be seen in 4.2. The table is divided into two sections, the first half for the results of the "incomplete" model and the second for the "complete" model. The first row of results are the mean errors for the training set, the second row for the test set. The error for a prediction is computed by comparing to the ground truth with the mesh point error, as explained in 4.2.1. The resulting error is equivalent to the mean over all errors. The last column is the ratio between test and training error.

4.4.3. Discussion

It is interesting to see, that the training error of the linear method and the random forest increases slightly going from the "incomplete" to the "complete" model. This must be caused by some examples in the test set being similar to some of the examples in the training set and forcing the parameters to adjust. The other three methods improve slightly in terms of training error due to the "complete" model. Independant of case, the test error is always larger than the training error. This is expected and was already discussed above.

The behaviour of the test error going from "incomplete" to "complete" is similar to the training error. The linear method worsens gravely, perhaps due to the "complete" model describing a wider space, making it harder for the linear method to find an appropriate mapping. All of the non-linear methods perform similarly in the "complete" case compared to the "incomplete" case in terms of mean test error.

Even though the best performing method in terms of error is the MLP, the MLP with ReLU has similar error values but has a slightly better test/train ratio. This could indicate that this method is more robust, meaning that the expected error for new data should be similar. It can be seen, that all non-linear methods outperform the linear method. This difference in performance could increase drastically, when the model gets more complex due to more data.

4.5. Parametric Models

For this experiment, it is explored how a parametric model based on an editor like MakeHuman performs against a model based on real data. As the topologies of the meshes used for the parametric models are different, the models are seperately evaluated and the resulting error heat maps are compared. Both models are tested, using the Ceres implementation by Biland [Bil17], mentioned in section 3.5. The resulting mesh is then compared to the NRICP solution.

4.5.1. Problem Setting

Firstly, a parametric model is created from MakeHuman. To achieve this, 8 semantic parameters of the editor are varied between 3 positions and exported into MATLAB. The semantic parameters and values used can be found in the appendix in table A.3. This results in 6'561 meshes, which are used to generate a parametric model. For the model based on real data, the parametric model generated from the *before* meshes, is used. The ground truth meshes for the real data are already available, as those point clouds were already processed earlier. The ground truth meshes for the MakeHuman model are generated by running NRICP with a modified MakeHuman mesh. The difference between the original mesh and the modified MakeHuman mesh can be seen in figure 4.3.

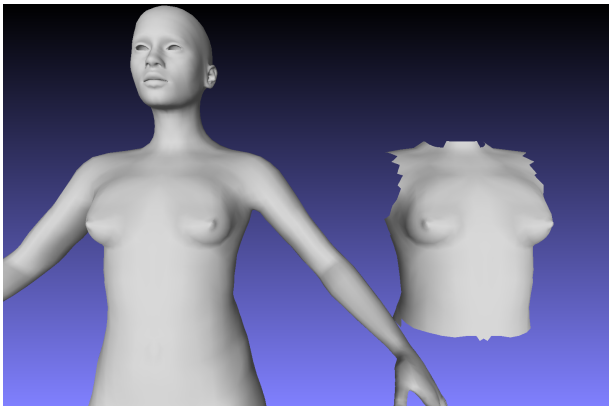


Figure 4.3.: *On the left, the original mesh of MakeHuman can be seen. The cut mesh is on the right.*

The Ceres implementation is run for both parameteric models, fitting both models to the test set of *before* point clouds. The resulting meshes are then compared to the ground truth, by generating an error heat map for both models. The parameters used in the implementation can be found in the appendix in table A.4.

4.5.2. Results

In figure 4.4, the error in distance per vertex is accumulated over all test set examples. The maximum vertex error is stated above the mesh for each model respectively, left for MakeHuman and right for real data. All errors were scaled inversely by the largest error, coloring the largest errors dark red and small errors dark blue. For the MakeHuman model the largest errors arise

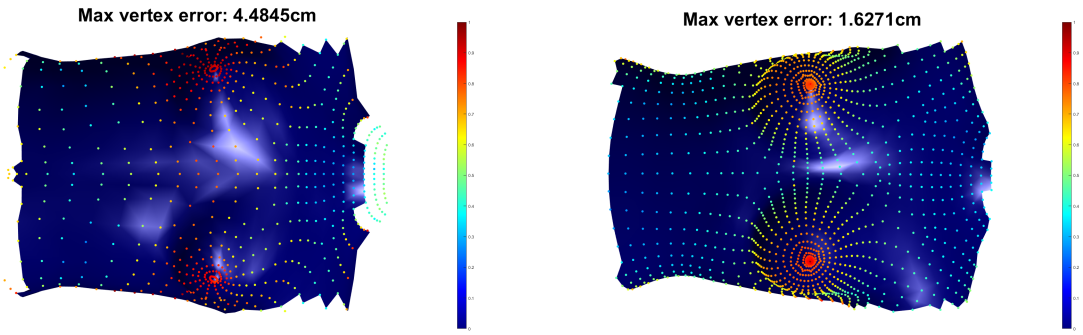


Figure 4.4.: *The left image shows the error heat map for the MakeHuman model. The right image depicts the error heat map for the model based on real data.*

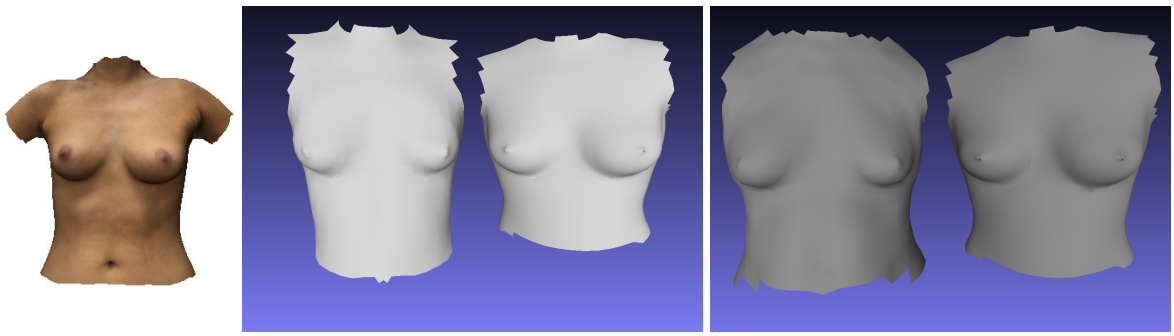


Figure 4.5.: *The left image shows the torso of the subject. The first two meshes (light grey) are the ground truth meshes, generated from NRICP. The two other meshes (dark grey) are the fits generated by the Ceres implementation. Of each pair, the left mesh represents the MakeHuman model and the right represents the real data model.*

around the nipple and the lower side of the breast. The real data model has the largest error around the nipple and the area to the outside of the breast.

4.5.3. Discussion

The maximum error for the MakeHuman model is roughly 2.5 times larger than for the real data model. Especially in the nipple area the error is quite large. It appears that this is caused by the difference to the ground truth. When comparing the shape of the MakeHuman ground truth to the input image in figure 4.5, it looks like the ground truth solution doesn't represent the input very well. This might be caused by the low resolution of the mesh, hindering the NRICP algorithm from capturing the breast shape features properly. On the other hand, the real data ground truth seems to portray the input image better.
The topologies of the meshes are very different. The real data mesh is a lot more refined around the breast and nipple. This leads to a naturally heigher weighting of the breast vertices for the real data model.

5

Conclusion

The first objective of this thesis was to find a mapping between *before* and *after* meshes and explore the possibility of non-linear mappings. It turned out to work quite well using a linear mapping, but the non-linear methods outperformed the linear method completely. Additionally, another model based on triangle deformations was considered and a modification of the point based model, where additional points were generated along the normals of the faces. The point normals method, surprisingly, performed the best, due to the fact that the additional points carried information about triangle surface size, which improves the model. The deformation model doesn't perform as well, as in most cases larger errors occured around the nipple. Perhaps due to the deformation formulation the nipple got slightly misplaced.
Lastly, a parametric model based on the MakeHuman editor was compared to a model based on real data. The real data model performed better, due to the fact that the mesh topology was more compatible for the task given. Additionally, that parameteric model was based on similar data. Considering these factors, the parameteric model, based on the editor, performed fairly well.

5.1. Outlook

As most of the tests and experiments were done on considerably small data sets, this is the main point, that could be improved to get more definite results. This could be achieved by using another implementation of SfM or finding better parameters for SfM, such that the reconstruction of the point clouds becomes more successful. Perhaps other data sets would become available, where 3D models are already given and could be used to create a parametric model.
Regarding the mappings, the *after* model was only based on a certain label. It would be interesting to explore the relationship between parameters from one model based on, for example, "300" enhancements and "350" enhancements. This could also allow to create intermediate

models by interpolating between models. Furthermore, the possibility of using tensorflow instead of scikit-learn could be investigated.
The parametric model generated from the MakeHuman meshes could also be further improved, by increasing the number of vertices per mesh. This would require a new implementation for PCA, as the current MATLAB implementation needs too much RAM to be able to load all the meshes and process them for PCA. Another possibility that needs to be explored, would consider to only use breast vertices as correspondences or weight those vertices heigher, due to the fact that the parametric model, generated from the MakeHuman meshes, modifies the complete torso, opposed to the model based on real data, where the most variation already lies in the breasts. This would lead to a better and fairer comparison.

A

Appendix

A.1. Tables

NRICP	**Rigid initialization**	false
	Use normals	false
	Epsilon	10^{-4}
	Lambda	1
	Bidirectional weighting	0
	Ignore boundary	0
	Normal weight	0
	Alphas	100, 10, 9, 8, 7, 6, 5, 4, 3, 2
Keypoint Weights	**Nipples**	20
	Sternum	0
	Navel	0
	Throat	0
	Armpits	0
	Breast insides	10
	Breast undersides	10
	Breast outsides	1
	Diagonal	0
	Border	0
	Align	1, 1, 1, 1
	Cut	1, 1, 1, 1

Table A.1.: *The parameters for the NRICP implementation used in this thesis.*

PCA	Method	Possible Parameters	Best
Inc.	**Random Forest**	'estimator__n_estimators':[5,10,20,30],	30
		'estimator__max_features': ('auto', 'sqrt','log2'),	auto
		'estimator__max_depth':[2,4,8,16]	8
	DT	'estimator__criterion':('mse','friedman_mse','mae'),	mse
		'estimator__splitter':('random','best'),	best
		'estimator__max_features':('auto', 'sqrt','log2'),	auto
		'estimator__max_depth':[2,4,8,16]	2
	MLP	'estimator__activation':('identity','tanh','logistic'),	identity
		'estimator__solver':('lbfgs','adam','sgd'),	sgd
		'estimator__hidden_layer_sizes': [(15,),(30,),(45,),(60,),(45,30,),(60,45,)],	(60,45,)
	MLPrelu	'estimator__activation':('relu'),	relu
		'estimator__solver':('lbfgs','adam','sgd'),	adam
		'estimator__hidden_layer_sizes': [(15,45,),(45,30,),(60,45,),(60,45,30),(60,45,15)],	(45,30,)
		early_stopping: true	true
Com.	**RF**	'estimator__n_estimators':[5,10,20,30],	30
		'estimator__max_features':('auto', 'sqrt','log2'),	auto
		'estimator__max_depth':[2,4,8,16]	4
	DT	'estimator__criterion':('mse','friedman_mse','mae'),	friedman_mse
		'estimator__splitter':('random','best'),	best
		'estimator__max_features':('auto', 'sqrt','log2'),	auto
		'estimator__max_depth':[2,4,8,16]	2
	MLP	'estimator__activation':('identity','tanh','logistic'),	tanh
		'estimator__solver':('lbfgs','adam','sgd'),	sgd
		'estimator__hidden_layer_sizes': [(15,),(30,),(45,),(60,),(45,30,),(60,45,)],	(60,45,)
	MLPrelu	'estimator__activation':('relu'),	relu
		'estimator__solver':('lbfgs','adam','sgd'),	sgd
		'estimator__hidden_layer_sizes': [(15,45,),(45,30,),(60,45,),(60,45,30),(60,45,15)],	(60,45,30,)
		early_stopping: true	true

Semantic Parameters	Min Value	Middle Value	Max Value
Body Proportion	0	0.5	1
Muscle	0.2	0.5	1
Weight	0.2	0.5	1
Age	0.5	0.75	1
Breast Firmness	0	0.5	1
Breast Size	0	0.5	1
Breast Trans Down	0	0	1
Breast Trans Up	1	0	0
Breast Volume Vert Down	0	0	1
Breast Volume Vert Up	1	0	0

Table A.3.: *A table of the values set for the semantic parameters of the MakeHuman editor. For the first 6 parameters the ranges went from 0 to 1. The last two parameters were made up of two parameters each. Therefore, to create 3 different variations, either one was set to 1 or both were set to 0.*

	Parameter	Value
Global	**MAX_NUM_PCS**	30
	NUM_PTS	300
	MAX_REP	100
	NUM_STD	2
Solver	**minimizer_progress_ to_stdout**	true
	num_threads	8
	use_nonmonotomic_ steps	true
	function_tolerance	10^{-6}
	mac_num_ iterations	500

Table A.4.: *The parameters are separated into global and solver specific parameters of the Ceres implementation. These are the set values in the implementation.*

Bibliography

[ACP03] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM transactions on graphics (TOG)*, 22(3):587–594, 2003.

[AMO] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[Bil17] S. Biland. Anatomically based body part modelling and fitting, 2017.

[dHCCG+12] Pablo de Heras Ciechomski, Mihai Constantinescu, Jaime Garcia, Radu Olariu, Irving Dindoyal, Serge Le Huu, and Mauricio Reyes. Development and implementation of a web-enabled 3d consultation tool for breast augmentation surgery based on 3d-image reconstruction of 2d pictures. *Journal of medical Internet research*, 14(1), 2012.

[GGC10] Giovanni Gallo, Giuseppe Claudio Guarnera, and Giuseppe Catanuto. Human breast shape analysis using pca. In *BIOSIGNALS*, pages 163–167. Citeseer, 2010.

[Hor87] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987.

[JLH+98] Shi-Hong Jeng, Hong Yuan Mark Liao, Chin Chuan Han, Ming Yang Chern, and Yao Tsorng Liu. Facial feature detection using geometrical face model: an efficient approach. *Pattern recognition*, 31(3):273–282, 1998.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[SF16] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[SP04] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 399–405. ACM, 2004.

[SZPF16] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

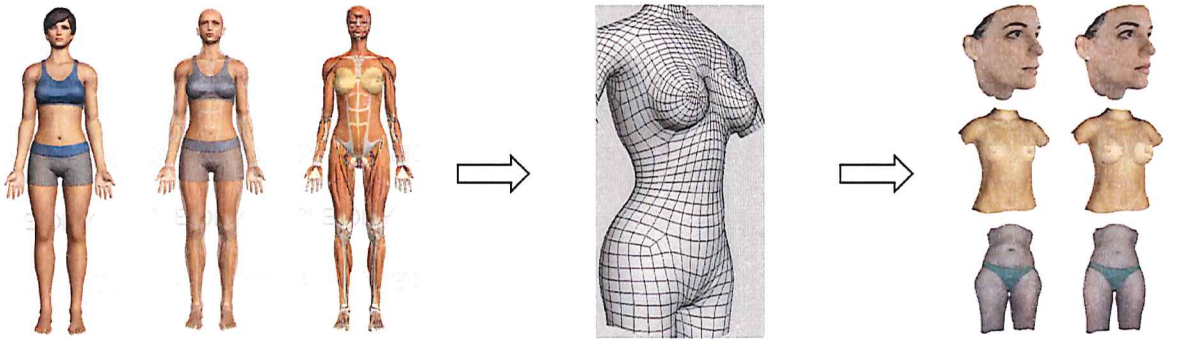
[ZFL+10] Shizhe Zhou, Hongbo Fu, Ligang Liu, Daniel Cohen-Or, and Xiaoguang Han. Parametric reshaping of human bodies in images. In *ACM Transactions on Graphics (TOG)*, volume 29, page 126. ACM, 2010.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

computer graphics laboratory

Semester Thesis

Inferring Human Body Parts and Correlations from Images, Pointclouds and Meshes



Anatomical Body Model | 3D Parametric Model | Applications

Introduction

Human body models have been crucial in the last decade for inference and synthesizing tasks. They find applications everywhere. A human body shape or pose is used in health monitoring, virtual cloth fitting, virtual avatars for virtual and augmented reality and medicine. Since a full body consists of its parts, in order to learn more expressive transformations, modelling of each part separately becomes a necessity. While there have been works applied to faces and hands, the upper torso has received little attention. Here, we focus on **inference**, **modelling** and **mapping** of the **the upper torso,** with applications in medicine.

Task Description

The semester thesis consists of the following steps :

- Get acquainted with the relevant literature in the field of body shape (upper torso) modelling/fitting and 3D reconstruction as well as with the code of a pervious student that consists of :
  - Retrieving pointclouds of various body parts from sprites utilizing SfM
  - Creating upper torso mesh and fit it to the retrieved pointclouds
  - Learning a parametric (PCA) model from the created dataset of fitted meshes
  - Fit model to new pointclouds
- Cluster the meshes based on characteristics/labels extracted during image extraction
- Learn mapping (between models) before and after procedure applied to the mesh
  - In mesh or/and image space (corresponding to a certain mesh)
- Manually annotate landmarks of captured real images (men/women) and fit meshes
  - From PCA model or Make human model

Remarks

A written report and an oral presentation conclude the thesis. The thesis will be overseen by Prof. Markus Gross and supervised by Endri Dibra (ETH).

Start: 27th of November 2017
End : 27th of May 2018

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

INFERRING HUMAN BODY PARTS AND CORRELATIONS FROM IMAGES, POINT CLOUDS AND MESHES

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
HALDIMANN	DAVID

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Opfikon, 27 May 2018	D. Haldimann

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.