Graz University of Technology

Institute for

Computer Graphics and Vision

Master Thesis

A 3D Reference Model for Presentation of 2D/3D Image Based Forensic Data

---

Johannes Höller

Graz, Austria, May 2011

Prof. Dr. Horst Bischof, Graz University of Technology, *reviewer*

Dr. Martin Urschler, Graz University of Technology, advisor

Abstract

In the context of Clinical Forensic Imaging our task is to create a tool which allows visualizing volumetric data from MRI/CT and conventional 2D image data in the same reference system. The data is difficult to interpret without a reference therefore we register the data to a reference model. First, we investigate different reference models. Second, we presented two registration processes; one for volume to 3D model registration and one for image to 3D model registration. For volume to 3D model registration we calculate an initial transformation based on corresponding markers between the reference model and the volume data. The initial transformation is refined with the Iterative Closest Point (ICP) algorithm. For image to 3D model registration we use the POSIT algorithm which uses corresponding markers between the reference model and the image data. As result we calculate texture coordinates to apply the image as a texture on the reference model. The experiments have shown that due to the ICP algorithm, the registration has a tolerance to measurement errors of the marker locations. On the other hand the image to 3D model registration process uses the POSIT algorithm; measurement errors of the marker locations directly influence the result. The appearance of the reference model has to be adjusted manually according to the data. The posture of the reference model can be adapted automatically for volume to mesh registration but nevertheless some adjustments have to be done manually. For image to 3D model registration the entire posture has to be adapted manually.

Keywords. medical image analysis, rigid image registration, pose estimation, reference model, ICP, POSIT, MakeHuman

Acknowledgments

I want to thank my advisor Dr. Martin Urschler for his patience, enthusiasm and continuous support during this thesis. I am also grateful to Prof. Horst Bischof who sparked my interest towards computer vision during his lectures. I want to thank the Ludwig Boltzmann Institute of Clinical-Forensic Imaging (LBI-CFI) for the opportunity to work on the project and the trust in me by sending me to a conference. Finally I want to thank my parents and my partner for their support throughout my undergraduate studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

The clinically well-established radiological methods Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) give us information that can also be used for forensic questions which is the aim of Clinical Forensic Imaging.

Clinical Forensic Imaging has taken on the task to develop a method for acquiring data of internal injuries in the living as basis for forensic expertise at court. Currently only the external inspection of the (living) body by means of images is used for the gathering of evidence. But the radiological methods allow the clinical forensic medicine to provide an additional and objective basis for examining and evaluating internal injuries in the living. Therefore these methods provide an increased legal certainty in the juridical processing.

The forensic evaluation of living cases following i.e. domestic and sexual violence, strangulation, child maltreatment, traffic related and other incidents has considerably gained in importance during the past years. The reasons, above others, are in general an increased sensibility towards acts of violence. Forensic imaging aims at the reconstruction of the sequence of events, the interpretation of severity of injuries and the life-threatening quality of an act. In combination with MRI and CT studies the external and internal inspection of the living cases present a unitary image of the injuries and provide a significant contribution to legal security.

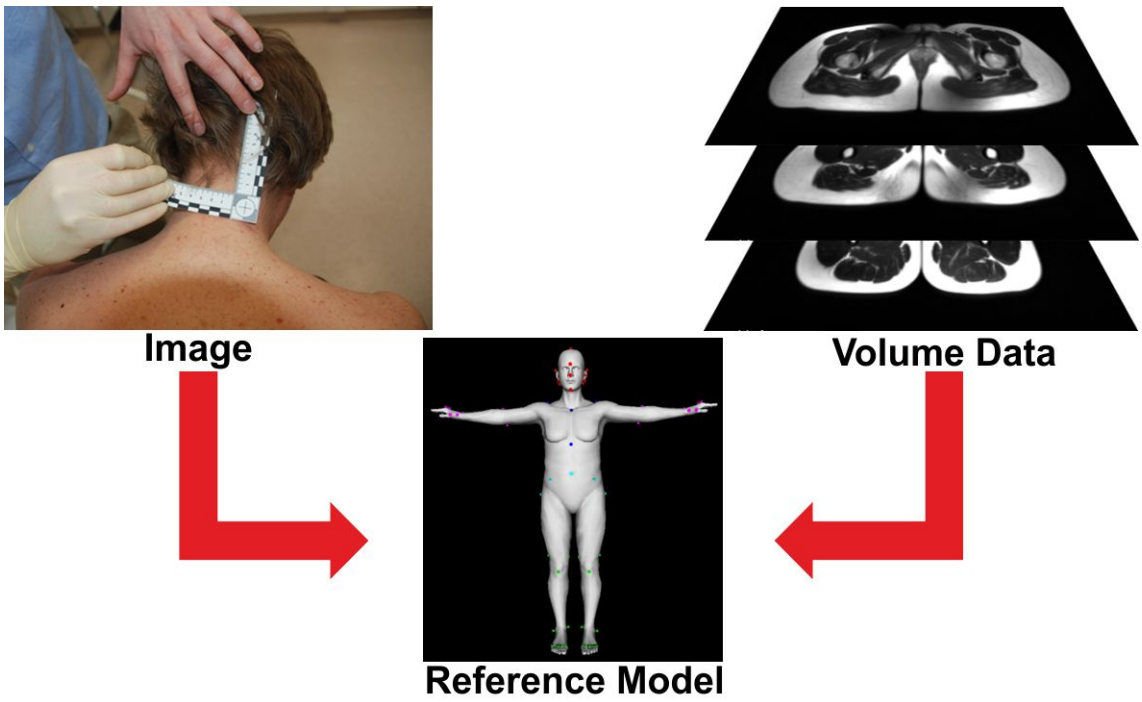


Figure 1.1: Schematic outline of the aim of this work. Image data and volume data is registered to the same reference model.

1.1 Motivation

The main goal of the Ludwig Boltzmann Institute of Clinical-Forensic Imaging (LBI-CFI)[i] is to lay the fundamentals for implementing clinical forensic imaging into the forensic routine examination of living persons. Typically MRI and CT are used in medicine therefore mostly medical personal uses them. On the contrary within juridical processes mainly non-physicians are involved. Data from MRI and CT are difficult to interpret but this often applies to images as well if there is no reference. In order to improve the application of clinical forensic imaging in juridical processes the data has to be prepared. Our goal is to create visualizations from data of clinical forensic imaging which are easily comprehensible for non-physicians. We also lay the fundamentals for a system to compare injury locations from 3D and 2D data.

[i] http://cfi.lbg.ac.at

1.2 Medical Imaging

There is a large variety of imaging modalities in clinical practice for example Single Photon Emission (SPECT), Positron Emission Tomography (PET) or Ultrasound (US). In this work we concentrate on Magnetic Resonance Imaging (MRI) and Computed Tomography (CT). Both modalities give us 3D-Voxel data with spacing information. With CT we have to consider the radiation dose because of ionizing radiation. Without a medical indication the usage of CT for living persons cannot be justified. MRI on the other side is a non-invasive technique. Since we focus on living persons MRI is the method of choice. Figure 1.2 shows example MRI images.

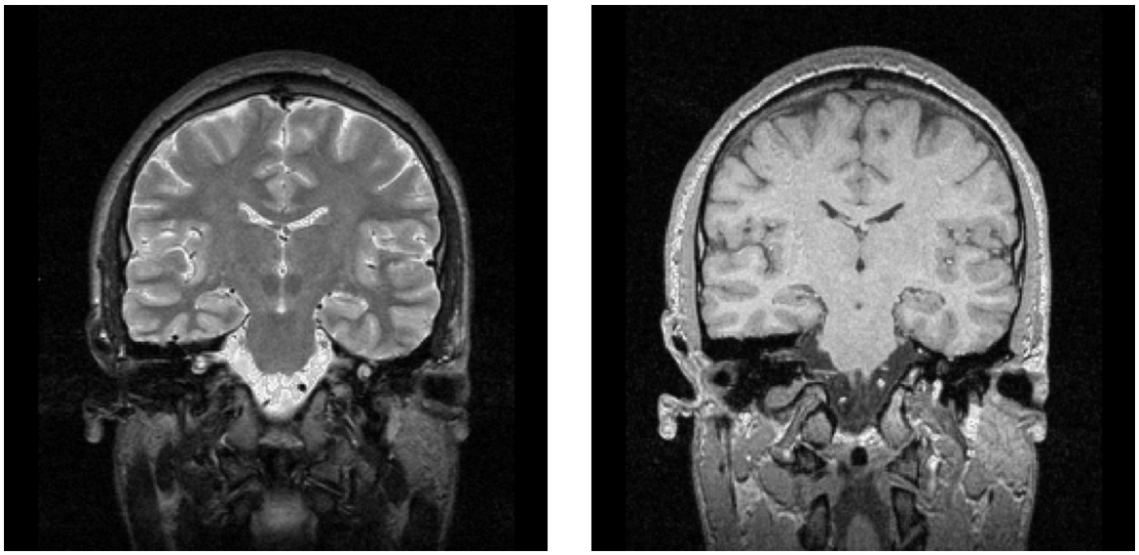


Figure 1.2: Brain MRI Images

1.3 Aims of the Thesis

In this thesis our goal is to develop a tool which allows visualization of evidence from image data from various sources. Furthermore internal evidence from MRI and CT needs to be displayed in the same reference model. Figure 1.1 illustrates the work within this thesis. To meet the demands we need to be able to register 2D-Images and 3D-Volumetric data to the reference model. The first application of the tool is to create visualizations which are used in court. Because of the presentation of the evidence on a reference model it is easier to discern the information. The second application is the usage for correlation

studies. Within correlation studies internal and external evidence is compared. The tool allows this comparison with respect to the location of the evidence in the reference model. The two applications have different requirements of accuracy. Accuracy is the difference between the real position and the position on the model. For visualizations a divergence of a few centimeters is tolerable while for correlation studies the divergence has to be within a few millimeters.

1.4 Structure of this thesis

In Chapter 2 we present corresponding state-of-the-art methods. Chapter 3 describes the selection process of finding a suitable model which we can use as reference model. In continuation we take a closer look at the selected model. The next Chapter 4 is engaged in the task of volume to mesh registration. Firstly it is necessary to create a 3D-Mesh from the volumetric data since the reference model is a 3D-Mesh. The initial registration is performed based on markers and is afterwards refined. We also describe a method to adapt the posture of the reference model. In the following Chapter 5 we register images to the reference mesh. The goal of Chapter 5 is to bring the image as texture on the reference model in the same position as shown in the image. Therefore we have to estimate the position of the person with respect to the camera. In Chapter 6 we performed experiments with the methods which are presented in Chapter 4 and Chapter 5. Finally Chapter 7 provides the conclusion of this thesis and discusses future work.

Chapter 2

Related Work

Contents

Our goal is to register 2D-Images and 3D-Volumetric data to the reference model therefore we have to investigate registration methods. Nowadays image registration is a wide area of research, a full review is beyond the scope of this thesis. Nevertheless there have been some important surveys on this topic in the recent years. A more general paper [4] addresses image registration techniques in computer vision, image processing, medical imaging and remote sensing. Especially for registration methods in computer vision a survey can be found in [31].

On the topic of medical image registration there are some standard books. One of them is the compendium for medical image registration [13]. The books [9] and [2] include chapters about medical image registration but are focused on image registration in a more general way.

In order to classify the variety of registrations methods we first make the distinction between linear and nonlinear methods. Secondly we divide them further in feature- and intensity based methods. Intensity based methods use the image directly whereas feature based methods require a preprocessing step to extract points of interest (POI).

2.1 A Definition of Image Registration

Generally image registration is the process of aligning two images. There is a large number of registration methods with various merits and limitations. Basically we use image registration for the following reasons:

- To combine information from multiple modalities. The two images might be acquired with the same sensors (intramodality) or with different sensors (intermodality).
- To monitor changes in size, shape or image intensity over time.
- To relate preoperative images and surgical plans to the physical reality of the patient in the operating room during image-guided surgery or in the treatment stage during radiotherapy.
- To relate an individual anatomy to a standardized atlas.

Now we give a more formal definition of image registration based on [19].

We name the two images which have to be aligned the moving image $I_M(\boldsymbol{x})$ and the fixed image $I_F(\boldsymbol{x})$. The spatial coordinates $\boldsymbol{x}$ belong to a domain Ω of a certain dimension $d \in \mathbb{N}$.

$$I_F(\boldsymbol{x}) : \Omega_F \to \mathbb{R}, \qquad \Omega_F \subset \mathbb{R}^d \tag{2.1}$$

$$I_M(\boldsymbol{x}) : \Omega_M \to \mathbb{R}, \qquad \Omega_M \subset \mathbb{R}^d \tag{2.2}$$

The goal of the registration process is to find a spatial transformation $\varphi(\boldsymbol{x}_F) : \Omega_F \to \Omega_M$. This spatial transformation $\boldsymbol{\varphi}(\boldsymbol{x}_F)$ needs to minimize the dissimilarity between the fixed image $I_F(\boldsymbol{x})$ and the transformed moving image $I_M(\boldsymbol{\varphi}(\boldsymbol{x}))$. With a distance measure D we can formulate the registration process as minimization problem such as finding the transformation $\boldsymbol{\varphi}(\boldsymbol{x})$ which minimizes the distance measure D between the fixed image $I_F(\boldsymbol{x})$ and the transformed moving image $I_M(\boldsymbol{\varphi}(\boldsymbol{x}))$.

In general there does not exist a direct solution for registration problems therefore iterative optimization schemes are used.

2.2 Rigid and Similarity Transformation

The basic transformations are translation T and rotation R. Each of them has three degrees of freedom in 3D-Space. Rigid transformations consist only of translation T and

rotation R. A similarity transformation has additionally to translation T and rotation R a scale parameter s which gives an additional degree of freedom in case of isotropic scaling. In total a similarity transform has seven degrees of freedom, three for translation T, three for rotation R and one for scaling s.

2.2.1 Feature-Based Methods

Feature-Based methods use specific structures which have to be extracted from the data. These structures can be anatomical or artificial markers. We divide feature-based registration methods in two main categories: the point-based methods and the surface-based methods.

The Procrustes alignment [13] is a point-based method which is able to estimate the rigid or affine transformation parameters between two points sets based on corresponding points pairs.

A common method for surface-based methods is the Iterative Closest Point (ICP) algorithm [3]. Correspondences between the two point sets representing surfaces are unknown a-priori which makes registration a difficult task. Two point sets are aligned by first identifying for each point of one point set the closest point in the other points set and take this as a correspondence. Second, we calculate a transformation based on the found correspondences. We apply the transformation to the first point set before the next iteration starts. The method stops when a distance threshold is reached. A drawback of the ICP algorithm is the fact that the algorithm generally reaches a local minimum. In [17] they present a ICP algorithm which conducts an exhaustive search to find a global minimum. A more detailed description of ICP can be found in Chapter 4 Section 4.4. In [26] they use rigid registration to register a CAD mesh model of a part with a CT volumetric model of assembly including that part. They extract the 3D-Mesh of the volumetric data and finally apply the ICP algorithm. They do not use the whole surface but extract edge features with 2D edge and 2D corner detection algorithms. The method is faster than typical ICP but less robust if the initial position of the source mesh is far from the target.

2.2.2 Intensity Based Methods

Intensity based methods do not extract features but use the intensities of the whole data. Drawback of those methods is often the computational effort since they work on the whole data. Due to this the solution is typically obtained by an iterative optimization process. In this work we do not use intensity based methods since we deal with surface based data.

2.3 Nonlinear Registration

In many applications a rigid or affine transformation is not sufficient therefore a nonlinear transformation is required i.e. the registration of abdominal images which is changed due to heart-beat or breathing.

In [8] they use a general-purpose nonlinear optimization to register 2D and 3D-Point sets instead of the ICP algorithm. The method can be enhanced by incorporating robust estimation without loss in speed which is an advantage over ICP.

There are intensity based methods as well but the optimization process is more complex due to the large number of unknowns. An example is optical flow estimation formulated in a variational setting.

2.4 Models

As a matter of fact it is often necessary to fit data to a model. The model may be static but a modifiable model is preferable. The principal component analysis (PCA) is a statistical approach to create a model.

PCA is a mathematical procedure in which a set of observations of possibly correlated variables is converted into a set of values of uncorrelated variables called principal components. By changing the principal components and back transformation we can generate new data. The principal components are typically ordered descending by the corresponding variance. In many cases PCA is used to reduce the dimensionality by omitting principal components with a variance below a threshold. The resulting error of the original and the reduced back transformed data depends on the omitted variance of the principal components. In [27] more detailed information on PCA can be found. In the context of models PCA is used to create statistical models. In this context observations are the points of aligned meshes. By altering the principal components and back transformation new meshes can be created.

In [25] they use the ICP scheme to non-rigidly fit a morphable template mesh to a point cloud. The fitting is a minimization by iteratively estimating the non-rigid shape and the rigid posture of the template mesh. The morphable shape is modeled as a single PCA model or as multiple PCA models for individual regions.

2.5 Pose Estimation

The pose estimation is the problem of finding the pose of the 3D-Object with respect to a camera in which the 2D-Image was taken. We assume that the 2D-Image is an image of the 3D-Object. There are numerous solutions for the pose estimation problem, [11] gives an overview on them.

Basically there are three main categories:

1. The analytic or geometric methods where the pose transformation is calculated from a set of equations which relate the 3D-Coordinates of the points with their 2D-Image coordinates. The POSIT algorithm from [5] belongs to this category.

2. Genetic algorithm methods are robust especially when the images are not perfectly calibrated. The pose represents the genetic representation and the projection error is used as the fitness function. In [29] they use a genetic algorithm to estimate the pose. A drawback of genetic methods is the increased runtime.

3. Learning based methods require a database of the object in different known poses during the learning phase of the system. A new image is matched against the system to estimate the pose. In [20] they use shape context as a descriptor of the images and Histograms of Oriented Gradients HOG are used in [22] to estimate the posture.

If more information is available other methods to estimate shape and posture can be used like structured light, photometric stereo or multiple cameras. In [1] they show a way to estimate shape and posture from multi-camera silhouettes. Other methods can track a human figure over a video sequence. With known initial configuration of the human body they track a human figure in [15]. In [23] walking pedestrians are tracked which constrains the posture on walking. Pfinder from [30] is able to track a human figure without estimating posture.

In [12] they optimize posture and body shape of a statistical 3D-Model based on silhouette, edge distance and shading. They initialize the method with manually selected joint positions. This approach is very laborious and computationally demanding and therefore not suited for our problem.

The majority of the methods assume certain knowledge of the posture therefore only the pose of the object is estimated. In [16] the posture is estimated because of known locations of points on the head and on body joints. Also the lengths of the body segments

have to be known. A drawback of this method is the large number of required feature points. At least six feature points on the head are required so this method is not applicable for images where the head is not visible.

Chapter 3

Reference model

Contents

In this Chapter we describe the process of finding a suitable reference model. The reference model is an integral part of this work therefore it is important which model we select. In the beginning we come up with several potential options for the reference models. The first option was to take a rigid human 3D-Model. This approach has some serious drawbacks, for instance all people do not look the same. Therefore a fixed 3D-Model has very little expressiveness and there is a major difference between 3D-Model and real world data. The next idea was to take a set of 3D-Models to meet the variability of human body shape. This seemed practicable but the use of a set of 3D-Models implies that every patient is measured in the same posture as the 3D-Model because otherwise the difference between 3D-Model and real world data increases again. Furthermore, a large number of 3D-Models are needed to cope with variations in posture and body shape. Finally we decided that a configurable model where both appearance and posture are modifiable with parameters is needed.

A more practical requirement for the model is its availability. The best model is of no use if we cannot obtain it.

Section 3.1 describes the reference model selection process in detail. In the following Section 3.2 we describe the chosen 3D-Model.

3.1 Finding a model

After research work we found two possible candidates: the SCAPE method (Shape Completion and Animation for People) from [7] and the MakeHuman Project[i]. These two are very different from each other; SCAPE is completely based on 3D-Scans whereas MakeHuman is based on an artificially created mesh which is modified to create universal characters.

3.1.1 SCAPE (Shape Completion and Animation for People)

The SCAPE method is a data-driven method for building a human shape model that accounts for different body shapes, posture and non-rigid deformations due to articulation. The method consists of a pose deformation model and a body shape deformation model which together builds the SCAPE model.

At first a set of scans of a single person in different poses is needed (an example set can be seen in Figure 3.1). With those scans the pose deformation model is learned. This pose deformation model derives the non-rigid surface deformation as a function of the pose of the articulated skeleton. Hence a pose is parameterized by a set of rigid body part rotations.

After the pose deformation model is learned, scans of different people in different poses are used to learn the body shape model. The body shape deformation model encodes the variability due to body shape across different individuals. A low-dimensional statistical model of body shape deformations is learned using principal component analysis (PCA). An example of the first principal components from the body shape model can be seen in Figure 3.2.

The two models are combined to produce 3D-Surface models with realistic muscle deformation for different people in different poses, when neither appears in the training set. The SCAPE model is focused on representing muscle deformations resulting from

[i] `http://www.makehuman.org`

[iii] `https://graphics.soe.ucsc.edu/private/data/SCAPE/`

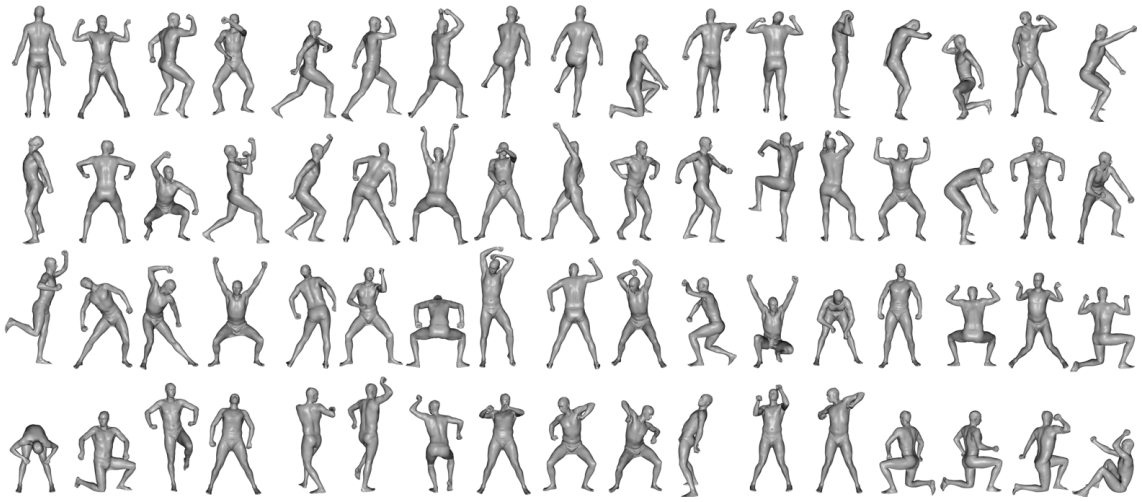


Figure 3.1: Example set of 3D-Scans used to learn the pose model. The image is taken from the SCAPE body shape database webpage[iii].

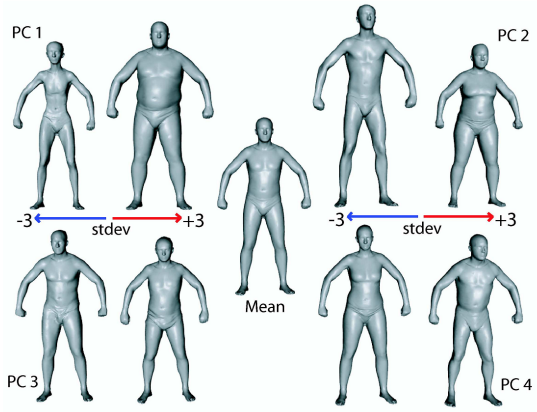


Figure 3.2: The image shows the first principal components in the space of body shape deformations. Image from [7].

articulated body motion. Deformations resulting from other factors are not encoded. One such factor is deformation resulting from pure muscle activity.

As the name SCAPE implies the purpose of the method is shape completion. This means generating a complete surface mesh from a limited set of markers specifying the target shape. An application of shape completion is partial view completion and motion capture animation. Additional information can be found at the project webpage[iv].

SCAPE is completely data driven and therefore requires scans of people. It is possible

[iv] `http://robotics.stanford.edu/~drago/Projects/scape/scape.html`

to purchase a database of scans like the CAESAR database[v] to learn a SCAPE model. The CAESAR database contains anthropometric variability of men and women with ages 18-65. People were chosen to ensure samples for various weights, ethnic groups, gender, geographic regions and socio-economic status. But in order to get a model for people younger than 18 we need scans from them as well.

3.1.2 MakeHuman

The intention of the MakeHuman project was to provide a tool for game developers and animators to create a human mesh model so they do not need to create human characters from scratch. The principal aim of the MakeHuman project is to develop an open source application capable of realistically modelling a very wide variety of human anatomical forms in the full range of natural human poses from a single universal mesh. With MakeHuman it is possible to generate a realistic character in a few minutes.

The humanoid mesh of MakeHuman can be parametrically manipulated and deformed in order to represent alternative anatomical characteristics. Also the posture of the mesh can be parametrically manipulated with respect to a common structural skeleton. The software is designed for real-time manipulation and real-time visualizations, therefore it needs to be efficient. Because of that the aim was to create a simplified and optimized mesh on which deformation can be realistically applied while maintaining a low polygon count.

With the current version MakeHuman 1.0 Alpha5[vi] it is possible to create different characters but only in the reference posture. MakeHuman 1.0 Alpha5 has not the ability to change the posture of the human mesh. Therefore we use MakeHuman 0.91 Release Candidate which was published in December 2007 and incorporates all required functionalities. In Figure 3.3 we can see a screenshot of MakeHuman 0.91 RC. The main differences of the current version MakeHuman 1.0 Alpha5 in comparison to MakeHuman 0.91 RC are the new GUI and a new mesh with improved topology but without ability to change the posture of the human mesh. Furthermore MakeHuman 1.0 uses a C/OpenGL core application and the user functionality is implemented in Python whereas MakeHuman 0.91 RC is completely written in C++/OpenGL.

MakeHuman is released under an Open Source Licence (GPL3.0) and available for Windows, Mac OS X and Linux. More detailed information can be found on the project

[v] `http://store.sae.org/caesar/`

[vi] `http://sites.google.com/site/makehumandocs/road-map`, State of 1.10.2010

webpage [vii].

Modifications of the mesh are performed by deforming the mesh rather than altering its topology. The possibility to manipulate the base mesh of the program arises from so called body and pose modifiers. Every modifier defines a certain manipulation like size for body modifiers or bend of an elbow for pose modifiers. A considerable number of mesh deformation targets have been created by artists to provide a large number of realistic starting points to model particular ethnic, gender, age and body mass figures.

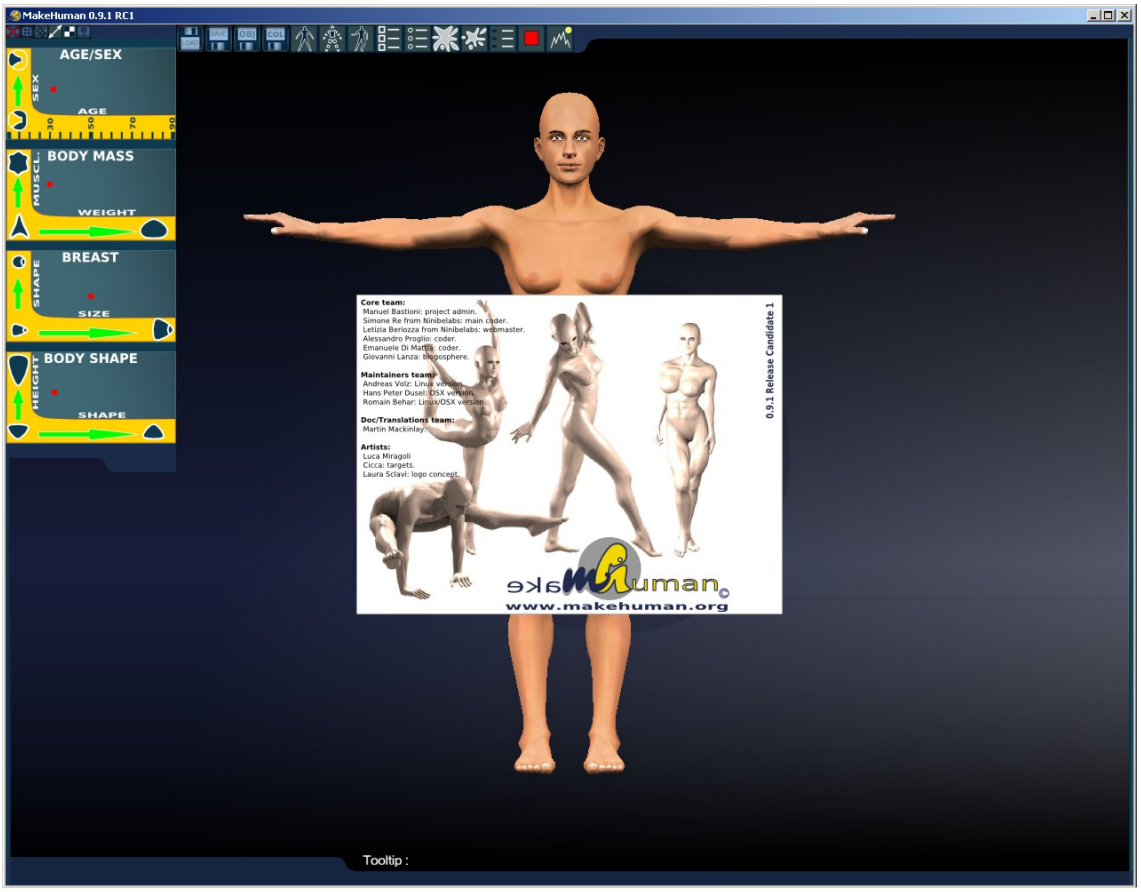


Figure 3.3: Screenshot of MakeHuman 0.91 RC after startup.

3.1.3 SCAPE vs. MakeHuman

The main differences between SCAPE and MakeHuman are:

- SCAPE is based on real-life data whereas MakeHuman uses an artificial mesh.

[vii] `http://makehuman.blogspot.com/`

- The expressiveness of the SCAPE model depends on the used 3D-Scans and for MakeHuman on its modifiers.

In the end we chose MakeHuman because right from the start it is possible to create meshes with a wide variety of body shapes and poses. The included modifiers allow us to create meshes of different people between ages 10y and 90y in different poses. In order to have the same variety with the SCAPE model a large database of 3D-Scans is necessary. For example in [12] they used scans from over 1000 men and 1000 women to learn their SCAPE model. It costs a lot of resources to acquire such a database either by buying or scanning.

3.2 MakeHuman 0.91 RC

The MakeHuman 0.91 RC program consists of three main parts: the GUI mhgui-0.2, the Animorph package animorph-0.3 and the modifier data. We do not discuss the GUI any further since we do not use it so we are more interested in the functionality of the Animorph package and in the modifier data.

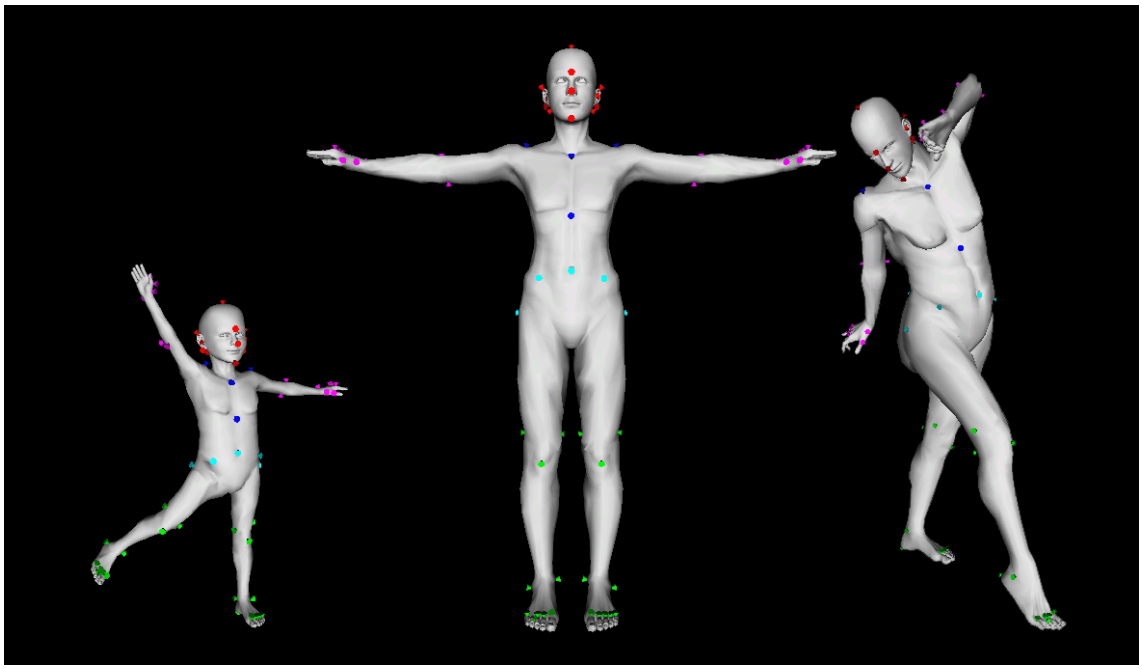


Figure 3.4: Illustration of the possibilities of the Animorph package. The markers described in Section 3.8 are shown.

3.2.1 Animorph package

The Animorph package contains the whole functionality to manipulate and maintain a mesh. No other special libraries are needed, just the standard C++ libraries. We use the Animorph package as it is, just with minor changes mostly due to operation system incompatibilities.

3.3 MakeHuman Data

The real power to create and manipulate a mesh arises from a set of different categories of modifier data files which are:

- **base** files containing initial information about the mesh
- **pose modifier** files giving information on how to perform changes of a specific joint
- **body modifier** files providing data to deform the mesh according to global body shape parameters
- **body detail modifier** files holding information on how to deform specific body parts
- **body settings** files store data of the applied pose or body / body detail modifiers

The whole data directory of the package consists of almost 3000 files. We describe a selection of the modifiers and files for each of the categories from above in the Appendix Section C.1.

3.4 Pose modifier

Pose modifier define how the mesh is affected by altering a joint parameter. There are 68 pose modifiers available. In Section C.3 of the Appendix we can see a body setting of an example posture of MakeHuman. We will show how a pose modifier works on the example of modifier group *200_left_upper_leg*. As the name suggests *200_left_upper_leg* is the hip joint. The hip joint has more than one possible rotation. As one can try in a self-experiment humans are able to lift a leg sideways as well as to the front. A pose modifier group possesses a pose modifier for each rotation. In our example *200_left_upper_leg* these are:

- *ROT_ADJUST1*, twist the leg
- *ROT_ADJUST2*, little sideways adjustment
- *ROT_BASE1*, lift leg sideways
- *ROT_BASE2*, lift leg to the front
- *ROT_BASE3*, lift leg a bit sideway and a bit to the front

Each of the pose modifiers can contain several PoseRotations and PoseTranslations (See Figure 3.6).

ROT_BASE3 from our example possesses six files:

1. *00_Y_LIMB_LUPPERLEG.rot*
2. *00_Y_LIMB_LUPPERLEG.rot.info*
3. *00_Y_LIMB_LUPPERLEG.target*
4. *00_Y_LIMB_LUPPERLEG.target.info*
5. *01_X_LIMB_LUPPERLEG.rot*
6. *01_X_LIMB_LUPPERLEG.rot.info*

That means we have two PoseRotations and one PoseTranslation for this pose modifier. An example of ROT_BASE3 with a rotation angle of 70° can be seen in Figure 3.5.

3.5 Body modifier

Body and body detail modifier consist only of PoseTranslations which denotes they contain only translation. We distinguish between two types of body shape modifiers. A body modifier works on the whole mesh at once whereas a body detail modifier influences only parts of the mesh. The latter is described in Section 3.5.1.

Now we are going to show how the body modifiers for the whole mesh are selected. We call a selected set of body modifiers a body setting. Section C.2 of the Appendix shows a body setting of an example character of MakeHuman. Within MakeHuman eight parameters are used to define the global body shape. It is possible to choose the body modifiers without the scheme of the eight parameters provided by MakeHuman but this scheme proved to give good result. Here are the eight parameters:

Figure 3.5: Figure of the base mesh with applied pose modifier *ROT_BASE3* of pose modifier group *200_left_upper_leg* with rotation angle of 70°.

1. **Age** 10-90
2. **Sex** male/female
3. **Muscle** skinny/big muscle
4. **Weight** thin/fat
5. **Breast Size** small/big
6. **Breast Shape** cone/sphere
7. **Body Shape** peer shape / V shape
8. **Body Size** small/tall

Based on these eight parameters a set of body modifiers is selected. The selection of the body modifiers from the eight parameters is done in the following way:

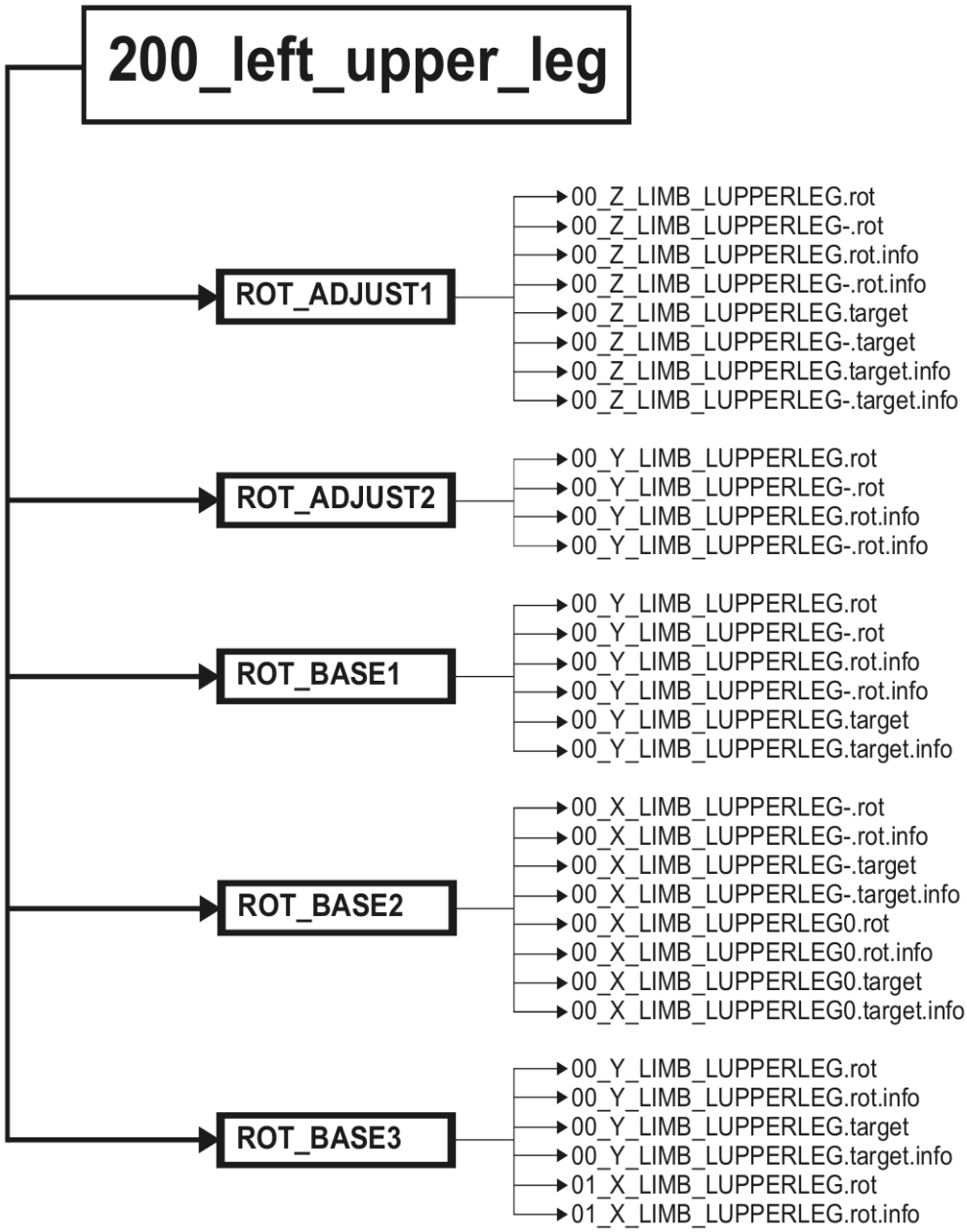


Figure 3.6: Hierarchical schemata of pose modifier *200_left_upper_leg* with all rotation subgroups and their corresponding PoseRotations and PoseTranslations.

1. **Step 1** Age and Sex parameters specify which body modifiers from the age category are selected.
2. **Step 2** Based on the selected modifiers from step 1 and with parameters Muscle and Weight the body mass and the body shape modifiers were selected.
3. **Step 3** With the selection from the former steps and with parameters Breast Shape and Size the body modifiers for Breast are selected.
4. **Step 4** The Body Shape and the Body Size body modifiers are selected only because of the Body Shape and Size parameters.

For example we like to create the following model: 30year female, big muscle, thin, small cone shape breast and a tall V shape body. This will lead us to the following body setting:

ages/female_30.target,1
muscleSize/female_30_skinny_muscle.target,1
breast/female_30_skinny_muscle_sphere_little.target,1
shapes/longilinear_vshape.target,1

The mesh is created in a straight forward procedure, we use the four Pose-Translations to alter the mesh. But what if we want to create this model: 40year male, normal muscle, average weight, normal cone shape breast and a medium sized a bit V shaped body. We do not have the exact body modifiers for this case like before. Therefore we use a linear combination of the available modifiers, for instance to create a 40year old man we combine body modifiers 30year male and 50year male.
The BodySettings of the 40year old man look like this:

ages/male_30.target,0.5
ages/male_50.target,0.5
muscleSize/male_30_big_muscle.target,0.122944
muscleSize/male_30_big_nomuscle.target,0.135214
muscleSize/male_30_skinny_muscle.target,0.132878
muscleSize/male_30_skinny_nomuscle.target,0.145537
muscleSize/male_50_big_muscle.target,0.145298
muscleSize/male_50_big_nomuscle.target,0.159798

muscleSize/male_50_skinny_muscle.target,0.157038
muscleSize/male_50_skinny_nomuscle.target,0.171998
shapes/brevilinear_peershape.target,0.0133096
shapes/brevilinear_vshape.target,0.400855
shapes/longilinear_peershape.target,0.0840619
shapes/longilinear_vshape.target,0.526334

We can see that not only the age modifiers are combined but the other modifiers as well. The value of each body modifier is calculated because of the distance from the given parameter to the body modifier. For instance to create the 40year old man we combine weight male_30.target and male_50.target each with 0.5. Body modifiers for sex can be combined as well since not every man has broad shoulders and not every woman has slim waists as the body modifiers state. Because of the use of linear combinations we have a wide range of possible models.

3.5.1 Body details modifier

Body details modifiers influence only parts of the mesh. For instance it is possible to alter the length of the upper arms separately from the rest of the mesh. The body details modifiers are grouped by the part they modify for example all body detail modifier for the nose are in the nose group.

3.6 How to alter the model

Now that we have seen how the parts look like we put them altogether to work. At first we need to set up a mesh. To do this the instance of the Animorph mesh needs to load the available base files, pose modifiers body modifiers and body detail modifiers. After that a basic mesh without any applied modifiers is available.

To apply modifiers to the mesh there are two important functions, doMorph for body and body detail modifiers and doPose for pose modifiers. Both functions require the name of a modifier and a value.

3.6.1 doMorph

The doMorph function applies body and body detail modifiers to the mesh. The function requires a value which is a number that must be within the limits $[0, 1]$; we call this number

morph value.

Based on the information of the PoseTranslation file associated with the modifier we can alter the mesh. Each vertex v specified in the file is transformed with the belonging translation vector t and the morph value m_{new} according to the following equation:

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} * [m_{new} - m_{current}] \tag{3.1}$$

It is necessary to use the difference $(m_{new} - m_{current})$ because if we have applied the same modifier before, we just need to apply the difference between the current morph value $m_{current}$ and the new morph value m_{new}. Body and body detail modifiers can be applied only when the model is in the initial posture, because all body and body detail modifiers have been designed for the initial posture.

3.6.2 doPose

The doPose function applies pose modifiers to the mesh. The function requires the name of the pose modifier and a value which is the angle of the rotation in degrees.

Since for a certain pose modifier multiple PoseRotations can be defined, we choose at first the appropriate PoseRotations according to the angle and its corresponding PoseTranslations. Every single PoseRotation is specified within certain limits and we select the PoseRotations where the angle is within those limits.

The corresponding PoseTranslation is applied to deform the mesh before applying the PoseRotation. This is done in a similar way as in Eq.(3.1) only with minor changes due to the fact that doPose uses an angle instead of a morph value.

To apply a PoseRotation we need to calculate a rotation matrix $\boldsymbol{R}$ for each vertex v. To calculate the rotation matrix $\boldsymbol{R}$ we need the affected axis and an angle. We get the affected axis from the PoseRotation. The rotation angle is obtained by multiplying the angle given to the function and the angle belonging to the vertex which we get from the PoseRotation. We get the rotation center C from the PoseRotation file as well. This leads us to the following equation:

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \boldsymbol{R} * \left(\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} - \begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix} \right) + \begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix} \tag{3.2}$$

The order in which we must apply pose modifiers is important since matrix multiplication is not commutative. The number in the name of each pose modifier group give us the order by sorting them ascending. The reason for this order is the definition of the pose modifiers. Each pose modifier was created based on the initial posture. If we use the pose modifier of the lower arm before using the pose modifier of the hand, the defined coordinate system of the hand joint is transformed with the pose modifier of the lower arm, while the data in the files of the pose modifier of the hand is not. Due to that we do not get a correct mesh. In practice there are functions like setPose to prevent this. This function setPose resets the mesh to the initial posture and applies all used pose modifiers in the right order.

3.7 Practical considerations

There are a few things one has to keep in mind when working with the Animorph package. Body modifiers can only be applied when the model is in the initial posture. The mesh saves its vertex data in multiple variables. It is necessary to keep in mind that some functions like setPose reset the mesh to initial posture by overwriting the main vertex data variable with a saved copy without posture. This is not a problem if we have ensured that this copy contains the vertex data of the mesh with applied body and body detail modifiers. Otherwise we just get the initial mesh with posture. To copy the vertex data there are functions for different modes like poseMode, animationMode and bodyDetailsMode.

3.8 Marker

We need markers for the exact localization of indications. Markers or reference points are structures of the human body which are radiologically and outwardly visible and verifiable.

For radiological examinations special markers are available which are sticked to the patient before the examination. In the volumetric data the position of those markers give us the exact location. Another possibility is that a radiologist identifies the marker location. For images visual markers are sticked to the patient but with the same aim; to allow us exact localization.

On the reference model we define the markers beforehand. In Figure 3.4 we can see the reference model with different appearance and posture and with applied markers. A complete list of the markers we use can be found in Section B.1 of the Appendix.

Chapter 4

Volume to Mesh Registration

Contents

In this chapter we show how to register volumetric data from CT or MR to the reference model from the previous Chapter 3. At first we extract the surface of the volumetric data and create a 3D-Mesh from it. This is necessary since the reference model is a 3D-Mesh. The registration process is based on the surface. Furthermore only rigid transformations are allowed, more specifically, we allow only rotation, translation and scaling i.e. a similarity transformation.

Before we are able to apply registration techniques we need to generate a 3D-Mesh from the volumetric data. At first we use region growing to extract the surface within the volume. After that we use the Marching Cubes algorithm (Section 4.1) to create a 3D-Mesh. We call this 3D-Mesh of the volumetric data volume mesh.

From the Marching Cubes algorithm we get a highly detailed volume mesh in comparison to the reference model. Due to that we can simplify the volume mesh to speedup computations. We use a tool called *tridecimator* which is built with VCG Lib[i]. Detailed information on mesh simplification can be found in Section 4.2.

[i] `http://vcg.sourceforge.net/index.php/Main_Page`

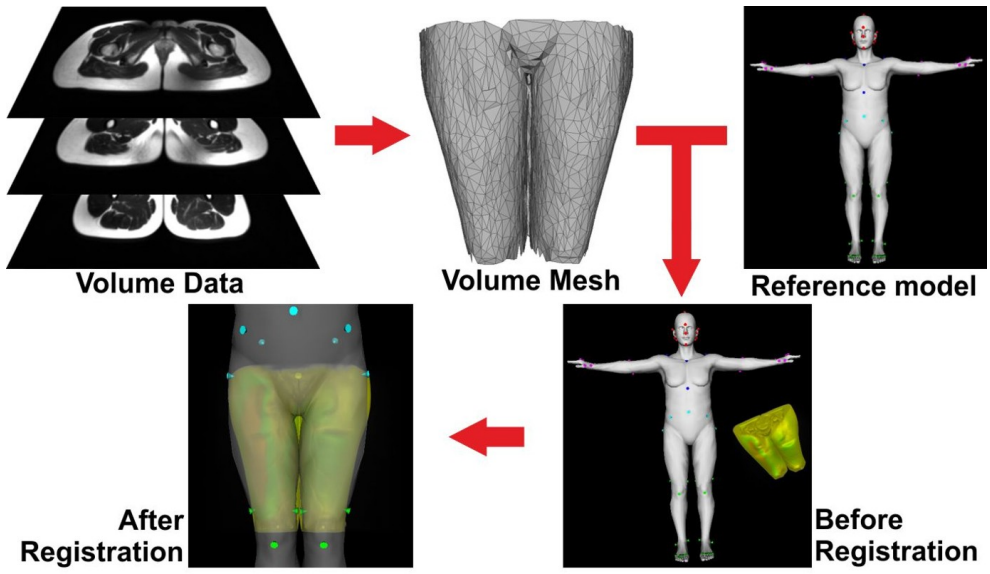


Figure 4.1: Pipeline of the volume to mesh registration process. At first we create a mesh from the volume data which we register on the reference model.

For instance from volumetric data which contains the body from knee to hip with resolutions $x = 1.11607$, $y = 1.11607$, $z = 6$ we get a mesh with 95234 vertices and 188122 faces. The same body area of the reference model consists of around 749 vertices and 1423 faces. After simplification the volume mesh consists of 1940 vertices and 2997 faces.

In a first step we calculate an initial transformation based on the correspondences between the markers of the volume and the markers of the reference model. The marker data of the volume needs to be provided in a text file. The initial transformation is calculated with the Insight Segmentation and Registration Toolkit (ITK)[ii] which uses the method described in Section 4.3.

This initial transformation is refined with the Iterative Closest Point (ICP) algorithm from Section 4.4. The ICP algorithm is more accurate than the registration based on markers since ICP uses all points of the volume mesh and the reference model to calculate a transformation whereas the initial transformation is computed based on a small number of markers. For example the initial transformation for the knee to hip volumetric data is calculated based on 4 markers and the refined transformation used all 1940 vertices of the volume mesh. We do not use ICP from the beginning because of the problem of local minima. In order to get out of this local minimum the error has to rise again and ICP is not able of letting the error rise again. In general the volume could be placed at random in

[ii] `http://www.itk.org`

the reference coordinate system with some scale. The goal of the registration with markers is to compute the scale and to place the volume on the correct position on the reference model. After this first transformation the risk of the ICP algorithm of getting trapped in a local minimum is low.

Since we cannot enforce that all volumetric data is acquired in the same posture we have to adapt the reference model. We adapt the model with the techniques which are described in Section 4.5.

After the volume to mesh registration process the volume mesh and the reference model are in the best alignment. It may be necessary to execute ICP registration and pose adaption more than once since after pose adaption ICP may find a better transformation.

4.1 Marching Cubes

The Marching Cubes algorithm from [18] is able to create a 3D-Triangle mesh from volumetric data which consists of multiple slices. Each slice is an image of density values. Lattice points in slice k are adjacent to lattice points in slice $k+1$. The Marching Cubes algorithm processes the volume by building cubes between four adjacent points in slice k and four adjacent points in slice $k+1$. Figure 4.2 illustrates how a cube is built.

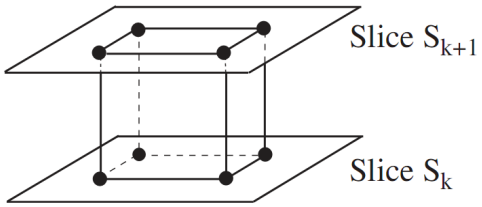


Figure 4.2: The cube is formed between slice k and slice $k+1$. Image from [21].

A cube consists of eight points. We assign one to a point if its voxel value v equals or exceeds the value of the surface we are constructing and zero otherwise. Points with a one are inside or on the surface whereas points with a zero are outside the surface. We assume that the surface intersects edges where one point is outside and the other one is inside the surface. There are eight points with two different states, this means that there are $2^8 = 256$ possible ways a surface could intersect the cube. By using symmetries of the cubes it is possible to reduce from 256 to 14 cases. Figure 4.3 shows the triangulation of the 14 cases. We create an index of each case based on the assigned number of the points. Figure 4.4 illustrates how the case numbers are obtained. This case number is used as a

pointer to a table which give us the triangles for a given cube configuration. The vertices of the triangles are the surface intersections. The surface intersections are estimated with subvertex accuracy by linear interpolation. In the final step of the Marching Cubes algorithm for each vertex the unit normal is calculated. By doing this for each cube we obtain a 3D-Mesh of the volume.

In summary, Marching Cubes creates a surface from a three-dimensional set of data as follows:

1. Scan two slices and create a cube from four neighbours on one slice and four neighbours on the next slice.
2. Calculate an index for the cube by comparing the eight density values at the cube vertices with the surface constant.
3. Using the index, look up the list of edges from a precomputed table.
4. Using the densities at each edge vertex, find the surface edge intersection via linear interpolation.
5. Calculate a unit normal at each cube vertex using central differences. Interpolate the normal to each triangle vertex.
6. Output the triangle vertices and vertex normals.

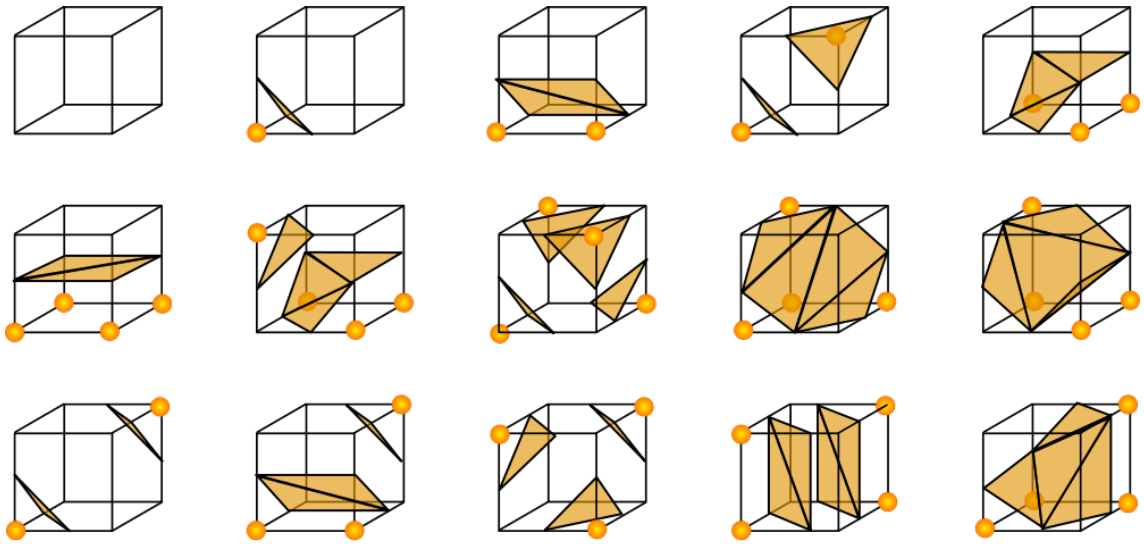


Figure 4.3: Triangulation of the possible configurations with respect to reflections and symmetrical rotations. Image from[iii]

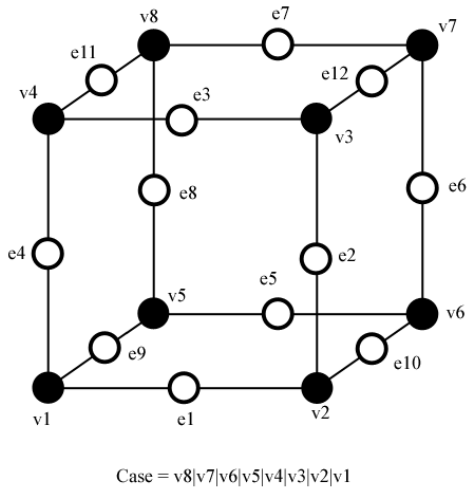


Figure 4.4: Numbering scheme of the cube. Image from[iv]

[iii] http://de.wikipedia.org/wiki/Marching_Cubes
[iv] http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

4.2 Mesh simplification using quadric error metrics

The method is described in [10]. The algorithm is based on iterative contraction of vertex pairs. This can be seen as a generalization of edge contraction.

The basic technique is pair contraction which is written as $(v_1, v_2) \rightarrow \overline{v}$ this denotes the movement of vertices v_1 and v_2 to the new position $\overline{v}$, the connection of all their incident edges to v_1 and the deletion of the vertex v_2. Figure 4.5 illustrates pair contraction and shows the two possibilities.

Beginning with an initial model M_n pair contractions are applied until the simplification goal is satisfied and the final model M_g is produced.

Algorithms based only on edge contraction cannot join disconnected components but pair contraction is able to merge individual components.

Possible candidate pairs for pair contraction are selected if either (v_1, v_2) is an edge or distance $\|v_1, v_2\| < t$ where t is a threshold parameter. The higher the threshold parameter t becomes the more likely it is that separated portions of the model can be connected. The candidate pairs need to be checked during the iterative process.

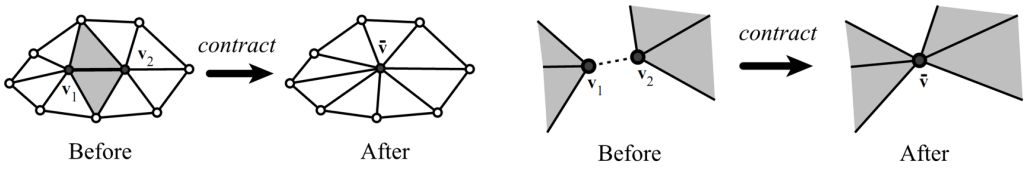
To select a pair for pair contraction it is necessary to calculate the cost of the contraction. The cost is defined for each vertex with a 4×4 matrix $\boldsymbol{Q}$. The error at vertex $\boldsymbol{v} = [v_x, v_y, v_z, 1]^T$ is defined as a quadratic form $\Delta(\boldsymbol{v}) = \boldsymbol{v}^T \boldsymbol{Q} \boldsymbol{v}$.

Summary of the steps of the method:

1. Compute the $\boldsymbol{Q}$ matrices for all the initial vertices.
2. Select all candidate pairs.
3. Compute the optimal contraction target $\overline{v}$ for each valid pair (v_1, v_2). The error $\overline{v}^T \boldsymbol{Q_1} + \boldsymbol{Q_2} \overline{v}$ of this target vertex becomes the cost of contracting that pair.
4. All pairs are sorted ascending according to their cost.
5. Iteratively remove the pair (v_1, v_2) of least cost by contracting the pair and update the cost of all valid pairs.

4.3 Marker based Registration

We use the method described in [14] to recover the transformation between two different Cartesian coordinate systems with given measurements of corresponding points. This photogrammetric problem is also referred to as the problem of absolute orientation.



(a) Edge contraction (b) Non-edge contraction

Figure 4.5: Illustration of pair contraction. In a) an edge is contracted to a single point. In b) non-edge pairs are contracted hence unconnected sections of the model are joined. (Images from [10])

The transformation between two Cartesian coordinate systems is modelled as a similarity transformation, thus the transformation can be decomposed into a rotation, a translation and a scale. We have got seven degrees of freedom three for translation, three for rotation and one for scaling. Three corresponding points provide nine constraints (three for each correspondence) more than enough to determine the seven unknowns. Since measurements are in general not exact, greater accuracy can be achieved by using more than three points. Therefore, no longer the transformation which maps the points from one coordinate system in the other system exactly is searched for, but the transformation which minimizes the sum of squares of residual errors. The advantage of the closed form solution to the least squares problem of absolute orientations is that it provides the best possible transformation in a single step. Because of that no initial guess is necessary as for iterative methods.

The input of the method are correspondences between points $\boldsymbol{r}_{l,i}$ from one coordinate system and points $\boldsymbol{r}_{r,i}$ from the other coordinate system.

The translation is just the difference of the right centroid $\overline{\boldsymbol{r}}_r$ and the scaled and rotated left centroid $\overline{\boldsymbol{r}}_l$.

$$\overline{\boldsymbol{r}}_l = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{r}_{l,i}, \quad \overline{\boldsymbol{r}}_r = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{r}_{r,i} \tag{4.1}$$

$$\boldsymbol{r_0} = \overline{\boldsymbol{r}}_r - sR\left(\overline{\boldsymbol{r}}_l\right) \tag{4.2}$$

The scale is obtained based on the sums of the squares of the measurement vectors

relative to their centroid by

$$s = \left(\sum_{i=1}^{n} \left\| \boldsymbol{r'}_{r,i} \right\|^2 / \sum_{i=1}^{n} \left\| \boldsymbol{r'}_{l,i} \right\|^2 \right)^{1/2} \tag{4.3}$$

with

$$\boldsymbol{r'}_{r,i} = \boldsymbol{r}_{r,i} - \overline{\boldsymbol{r}}_r, \quad \boldsymbol{r'}_{l,i} = \boldsymbol{r}_{l,i} - \overline{\boldsymbol{r}}_l \tag{4.4}$$

Eq.(4.3) is derived based on a symmetrical expression for the error term. This allows us to determine scale without the need to know the rotation.

Because of the derivation of scale it is necessary to find a rotation which maximizes

$$\sum_{i=1}^{n} \boldsymbol{r'}_{r,i} \cdot R\left(\boldsymbol{r'}_{l,i}\right) \tag{4.5}$$

Quaternions are used to find a solution for Eq.(4.5). Unit quaternion notation has some advantages. For instance it is easier to enforce a quaternion to have unit magnitude than ensure that a matrix is orthonormal. A quaternion is basically a vector with four components which can be interpreted as scalar with an ordinary vector or as complex number with three different imaginary parts. A symbol with a circle above it denotes a quaternion.

$$\mathring{q} = q_0 + iq_x + jq_y + kq_z \tag{4.6}$$

Further information on quaternions are available in [14].

Eq.(4.5) can be rewritten as:

$$\sum_{i=1}^{n} \left(\mathring{q} \mathring{r}'_{l,i} \mathring{q}^* \right) \cdot \mathring{r}'_{r,i} = \sum_{i=1}^{n} \left(\mathring{q} \mathring{r}'_{l,i} \right) \cdot \left(\mathring{r}'_{r,i} \mathring{q} \right) = \mathring{q}^T \boldsymbol{N} \mathring{q} \tag{4.7}$$

where $\mathring{q}^*$ is the conjugate quaternion of $\mathring{q}$.

The unit quaternion that maximizes

$$\mathring{q}^T \boldsymbol{N} \mathring{q} \tag{4.8}$$

is the eigenvector corresponding to the most positive eigenvalue of the matrix $\boldsymbol{N}$.

Matrix $\boldsymbol{N}$ is composed of elements of a 3×3 matrix $\boldsymbol{M}$ whose elements are sums of products of coordinates measured in the right and left coordinate systems.

$$\boldsymbol{M} = \sum_{i=1}^{n} \boldsymbol{r'}_{r,i} \boldsymbol{r'}_{l,i}^{T} = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix} \tag{4.9}$$

The 4×4 matrix $\boldsymbol{N}$ can be created as following:

$$\boldsymbol{N} = \begin{bmatrix} (S_{xx} + S_{yy} + S_{zz}) & S_{yz} - S_{zy} & S_{zx} - S_{xz} & S_{xy} - S_{yx} \\ S_{yz} - S_{zy} & (S_{xx} - S_{yy} - S_{zz}) & S_{xy} - S_{yx} & S_{zx} + S_{xz} \\ S_{zx} - S_{xz} & S_{xy} + S_{yx} & (-S_{xx} + S_{yy} - S_{zz}) & S_{yz} + S_{zy} \\ S_{xy} - S_{yx} & S_{zx} + S_{xz} & S_{yz} + S_{zy} & (-S_{xx} - S_{yy} + S_{zz}) \end{bmatrix} \tag{4.10}$$

The eigenvalues of N can be found as solution of the fourth order polynomial λ which is obtained by

$$\det\left(\boldsymbol{N} - \lambda I\right) = 0 \tag{4.11}$$

The corresponding eigenvector $\mathring{e}_m$ to the most positive eigenvalue λ_m is obtained by

$$\left[\boldsymbol{N} - \lambda_m I\right] \mathring{e}_m = 0 \tag{4.12}$$

.

Summary of the algorithm:

1. Find the centroids $\overline{\boldsymbol{r}}_l$ and $\overline{\boldsymbol{r}}_r$ of the two sets of measurements of the left and right coordinate system.
2. Subtract the centroids of the measurements such that from now on all measurements are relative to the centroid.
3. Calculate matrix $\boldsymbol{M}$ with the measurements as in Eq.(4.9).
4. Build matrix $\boldsymbol{N}$ with the elements of matrix $\boldsymbol{M}$ as shown in Eq.(4.10).
5. Compute the eigenvalues of $\boldsymbol{N}$ with Eq.(4.11) and pick the largest positive eigenvalue λ_m.
6. Obtain $\mathring{e}_m$ as the eigenvector corresponding to λ_m with Eq.(4.12). The quaternion representing the rotation is the unit vector in the same direction.
7. Now the scale is calculated with Eq.(4.3).
8. With given scale and rotation we can compute the translation with Eq.(4.2).

4.4 Iterative Closest Point

The iterative closest point (ICP) algorithm is a method to align 2D or 3D-Point sets. In [3] the name ICP was introduced and the basic algorithm was presented. As input we have a data point set P with points p_i and a model point set M with points m_i. In general M can be any surface if it is possible to calculate the distance between the surface and a point.

Our goal is to find the parameters of the transformation T in order to reduce the error between P and M. We collect the parameters for transformation T in a vector a. For 3D-Registration we define

$$T_{3D}(\boldsymbol{a}; x) = T(\alpha, \beta, \gamma, t_x, t_y, t_z; x) = [R(\alpha, \beta, \gamma)]\, x + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \quad for\ x \in \mathbb{R}^2 \tag{4.13}$$

where $R(\alpha, \beta, \gamma)$ is the rotation matrix built from α, β and γ.

The alignment is measured with the error function $\epsilon^2(|x|)$ which is typically defined as the length of the vector

$$\epsilon^2(|x|) = \|x\|_2 \tag{4.14}$$

The correspondences are denoted by a function $\phi(i)$ which selects for each data point p_i the corresponding model point $m_{\phi(i)}$. We establish correspondences between P and M by looking for each point p_i for the point $m_{\phi(i)}$ which minimizes the error function. For the error function in Eq.(4.15)

$$\phi(i) = \arg\min_j \|m_j - p_i\|_2 \tag{4.15}$$

The overall error $E(\boldsymbol{a})$ between data P and model M is defined as

$$E(\boldsymbol{a}) = \sum_i \min_j \epsilon^2(|m_j - T(\boldsymbol{a}; p_i)|) \tag{4.16}$$

In the basic form the algorithm consists of two steps. At the beginning we need an initial estimate of $\boldsymbol{a}_0$. In each iteration a new estimate of the transformation parameters $\boldsymbol{a}_k$ is calculated which reduces the error $E(\boldsymbol{a})$ between model and data point set.

1. **Step 1** Compute correspondences $\phi(i)$:

$$\phi(i) = \arg\min_j \epsilon^2 \left(|m_j - T(\boldsymbol{a}_k; p_i)|\right) \tag{4.17}$$

2. **Step 2** Update transformation, $\boldsymbol{a}$:

$$\boldsymbol{a}_{k+1} = \arg\min_{\boldsymbol{a}} \sum_i \epsilon^2 \left(|m_j - T(\boldsymbol{a}_k; p_i)|\right) \tag{4.18}$$

   For many common cases $\boldsymbol{a}_{k+1}$ can be calculated in a closed form [13] (for example see Section 4.3).

Since the error $E(\boldsymbol{a})$ is reduced in every iteration convergence to a local minimum is guaranteed. We can use $\phi(i)$ for a termination criterion. If there is no change in $\phi(i)$ then $\boldsymbol{a}_{k+1}$ does not change as well hence no further change is possible.

The ICP algorithm guarantees to find a local minimum but no global minimum. To overcome the risk of getting caught in a local minimum we can take the best solution of several initial transformation parameter a_0. The basic ICP algorithm uses a least squares norm Eq.(4.14) which is not robust to outliers. But by integration of a robust norm in the ICP algorithm the closed solution of the transformation update step is lost.

In [8] they use nonlinear minimization in order to reduce the overall error $E(\boldsymbol{a})$ between data P and model M from Eq.(4.16) instead of the basic ICP algorithm. The advantage of the method they describe is the increased robustness due to the use of robust statistics without loss of speed. In [17] they present a ICP algorithm to find a global minimum which has greater expense because the algorithm conducts an exhaustive search.

4.5 Adaption of Model

We do not assume that a volume was acquired in a certain posture therefore we need to adapt the posture of the reference model to the posture of the volume. The adaption of the model is essential because without adaption the distance between the reference model and the volume would be not tolerable. Also we would face a problem with registration because the transformation obtained between different postures would not be usable.

The adaption is divided in two main classes. First, the adaption based on markers and secondly the adaption based on the whole mesh.

4.5.1 Adaption based on markers

After the volume is registered to the reference model with markers we estimate the posture in a first attempt. The marker based adaption is divided in three types which differ in the number of markers they need and in the type of computation.

- **Type 1** needs two markers, the affected axis and the name of the pose modifier.
  We assume that markers M_1 of the reference model and the volume are aligned. Then we calculate the vector between the marker positions of marker M_1 and marker M_2 for reference model and volume mesh which gives us two vectors v_{ref} and v_{vol}. See Figure 4.6 for an illustration. The two vectors are projected on xy, xz and yz planes and angles for both vectors are calculated with the arctan function (see Figure 4.7). Based on the given axis we compute the difference between the angles of the according projection. The difference gives us the angle which we apply to the specified pose modifier.

$$
\begin{aligned}
\boldsymbol{v}_{ref} &= \text{Reference model } \left(\overline{M_1 M_2}\right) \\
\boldsymbol{v}_{vol} &= \text{Volume } \left(\overline{M_1 M_2}\right) \\
\alpha &= \begin{cases}
\arctan\left(\frac{\boldsymbol{v}_{ref}(z)}{\boldsymbol{v}_{ref}(y)}\right) - \arctan\left(\frac{\boldsymbol{v}_{vol}(z)}{\boldsymbol{v}_{vol}(y)}\right) & \text{if } x\text{-axis} \\
\arctan\left(\frac{\boldsymbol{v}_{ref}(z)}{\boldsymbol{v}_{ref}(x)}\right) - \arctan\left(\frac{\boldsymbol{v}_{vol}(z)}{\boldsymbol{v}_{vol}(x)}\right) & \text{if } y\text{-axis} \\
\arctan\left(\frac{\boldsymbol{v}_{ref}(y)}{\boldsymbol{v}_{ref}(x)}\right) - \arctan\left(\frac{\boldsymbol{v}_{vol}(y)}{\boldsymbol{v}_{vol}(x)}\right) & \text{if } z\text{-axis}
\end{cases}
\end{aligned}
\tag{4.19}
$$

- **Type 2** needs three markers and the name of the pose modifier.
  We only measure the angle between the two vectors $v_1 = \overline{M_1 M_2}$ and $v_2 = \overline{M_1 M_3}$ for the markers of the reference model and for the markers of the volume. With the vectors v_1 and v_2 we calculate the angle α_{ref} for the reference model and an angle α_{vol} for the volume.

$$
\begin{aligned}
\boldsymbol{v}_1 &= \overline{M_1 M_2} \\
\boldsymbol{v}_2 &= \overline{M_1 M_3} \\
\alpha &= \arccos\left(\frac{\boldsymbol{v}_1 * \boldsymbol{v}_2}{|\boldsymbol{v}_1| * |\boldsymbol{v}_2|}\right)
\end{aligned}
\tag{4.20}
$$

  The difference between α_{ref} and α_{vol} is then applied to the pose modifier. Figure 4.8 shows an illustration of the elbow joint. Type 2 is simpler than Type 1 and can only

be used for certain joints where the possible angles are only positive or negative, for instance the elbow joint.

- **Type 3** needs one marker and the name of the pose modifier.
  We assume that pose modifiers with a higher number are already aligned. This is necessary since before we are able to adapt for example the knee joint we have to adapt the hip joint. At first we select equally distributed angles in the range of possible angles of the pose modifier. For each angle we change the posture of the reference model and calculate the distance between the marker M_{ref} of the reference model and the marker M_{vol} of the volume. Around the angle with the smallest distance we use a gradient descent method to obtain the best angle. The final angle is applied to the pose modifier. Figure 4.8 illustrates the process.

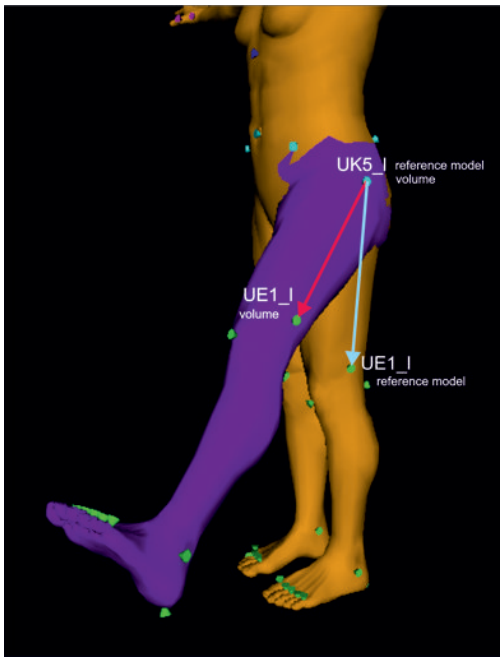


Figure 4.6: The figure shows the vectors we use to estimate the angle for Type 1 adaption. One vector is between two markers of the reference model and the second vector is between the same two markers but on the volume.

Type 1 and Type 2 are just approximations since markers are on the skin and joints are inside the body but we do calculations as if the joint was on the skin. Type 3 is not affected by this problem but since it depends on the accuracy of a single marker it still can be seen as approximation. Experiments have shown that these approximations are sufficient as initialization to ICP.

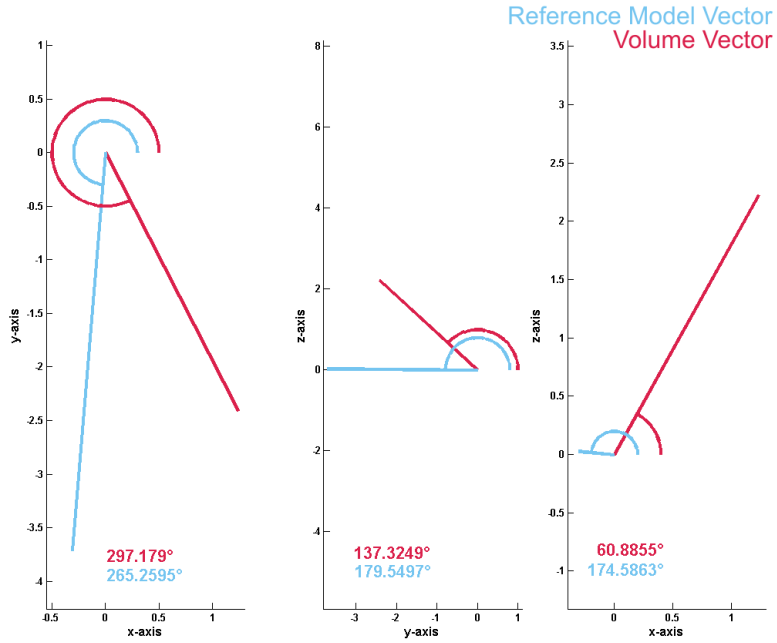


Figure 4.7: Type 1 projects both vectors on the xy, xz and yz plane and then calculates the value of the pose modifier according to the given axis. The figure shows the projections of the two vectors of Figure 4.6.

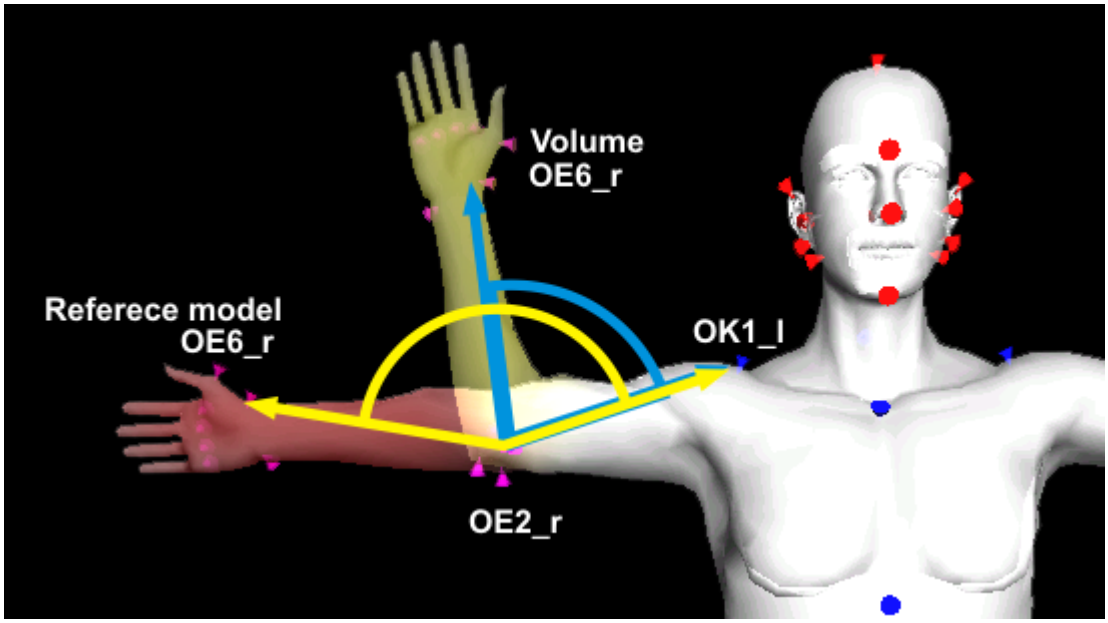


Figure 4.8: Here we see the vectors $\overline{OE2_rOK1_r}$ and $\overline{OE2_rOE6_r}$ for the reference model and for the volume. Type 2 uses the two vectors between three markers and calculates the angle between them. The difference between the angle of the reference model and the angle of the volume gives us the value for the pose modifier.

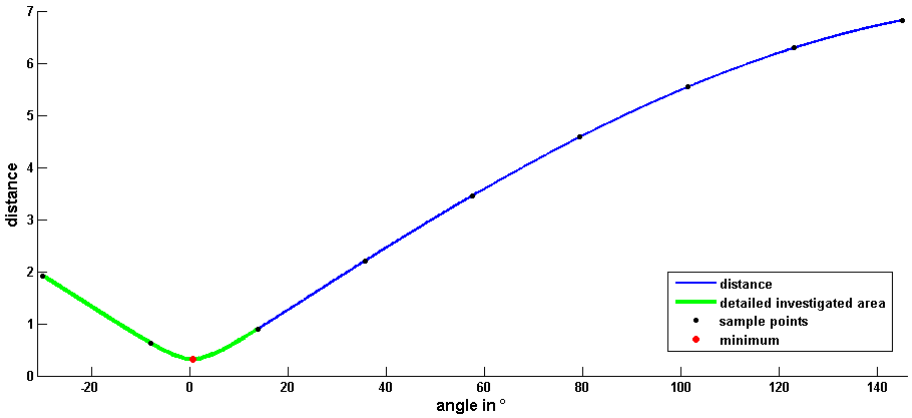


Figure 4.9: The Figure shows an example usage of Type 3. The distance is measured for marker *UE1_l* for pose modifier *200_left_upper_leg/ROT_BASE2*. The adaption with type 3 calculates at first the distance for a number of sample points. After that we search for the minimum distance around the sample point with the smallest distance. The corresponding angle of the minimum distance is the sought value for the pose modifier.

4.5.2 Adaption based on the whole mesh

Adaption of the whole mesh uses an iterative approach. We assume that the correct angle minimizes the distance between reference model and volume. The function needs only the name of a pose modifier as parameter. The function measures the distance ϵ_k between reference model and volume. After that we update the angle of the specified pose modifier with the step size s and measure the distance again. This gives us ϵ_{k+1}. Based on the difference between ϵ_k and ϵ_{k+1} we decide how to change the step size s. If $\epsilon_k - \epsilon_{k+1} > 0$ the distance is decreasing we can go on like before. If $\epsilon_k - \epsilon_{k+1} < 0$ the distance is increasing we change direction and reduce the step size s. Figure 4.10 shows the curve on which we are moving on.

A drawback of the adaption method is that it depends in which order we adapt the pose modifiers. A different order gives us a different posture. The problem arises from the fact that the pose modifiers of the reference model were define from outward to inward and we have to adapt from inward to outward.

4.6 Images of example data

In this Section we show images from all steps of the image to mesh registration process.

After the volume mesh was created from volumetric data it is placed in the reference

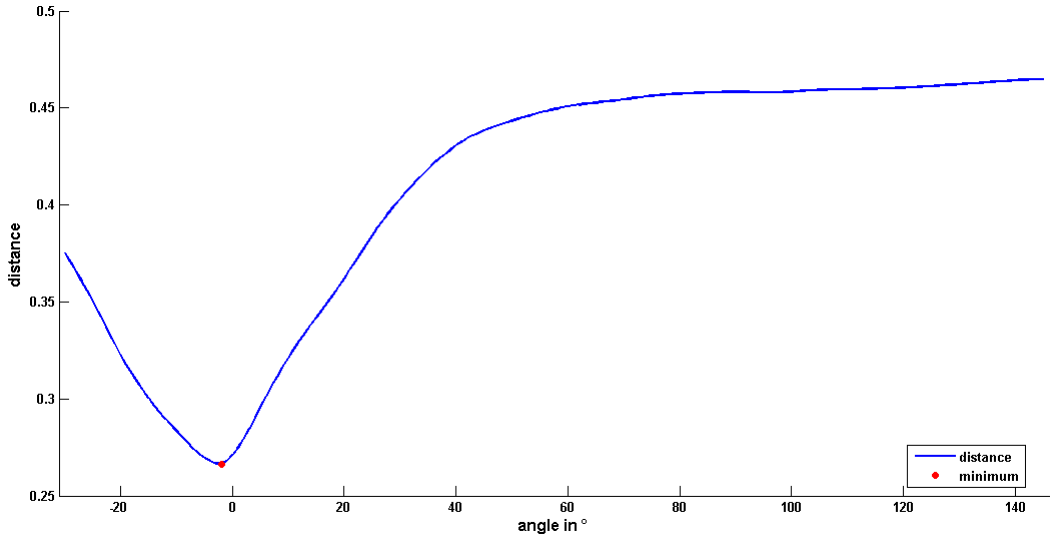


Figure 4.10: The figure shows the distance between the reference model and the volume as a function of the angle.

coordinate system. The first Figure 4.11 shows the initial situation. Position and scale of the volume mesh are arbitrary.

We begin the registration process with marker registration. Based on the markers we calculate a transformation to place the volume mesh with an appropriate scale on the reference model.

We refine the transformation with the ICP algorithm in order to get the best transformation. Figure 4.12 shows the registration with markers and ICP.

In the final step we adjust the posture of the reference model. With the adaption process we can reduce the distance between the volume and the reference model further. Figure 4.12 shows the difference before and after adaption.

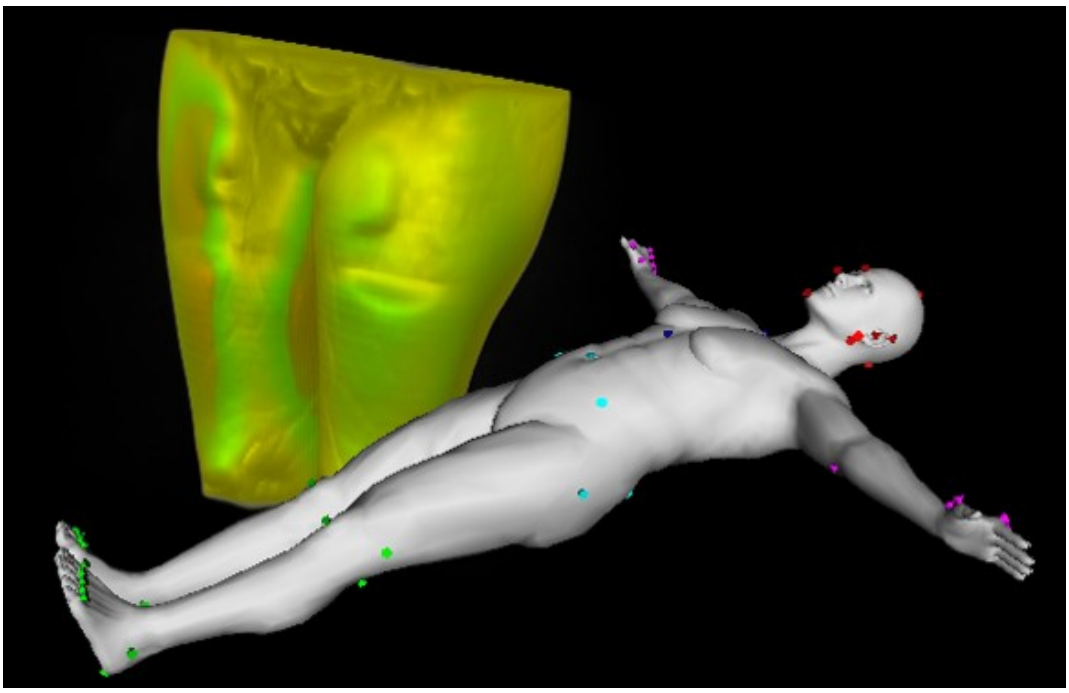
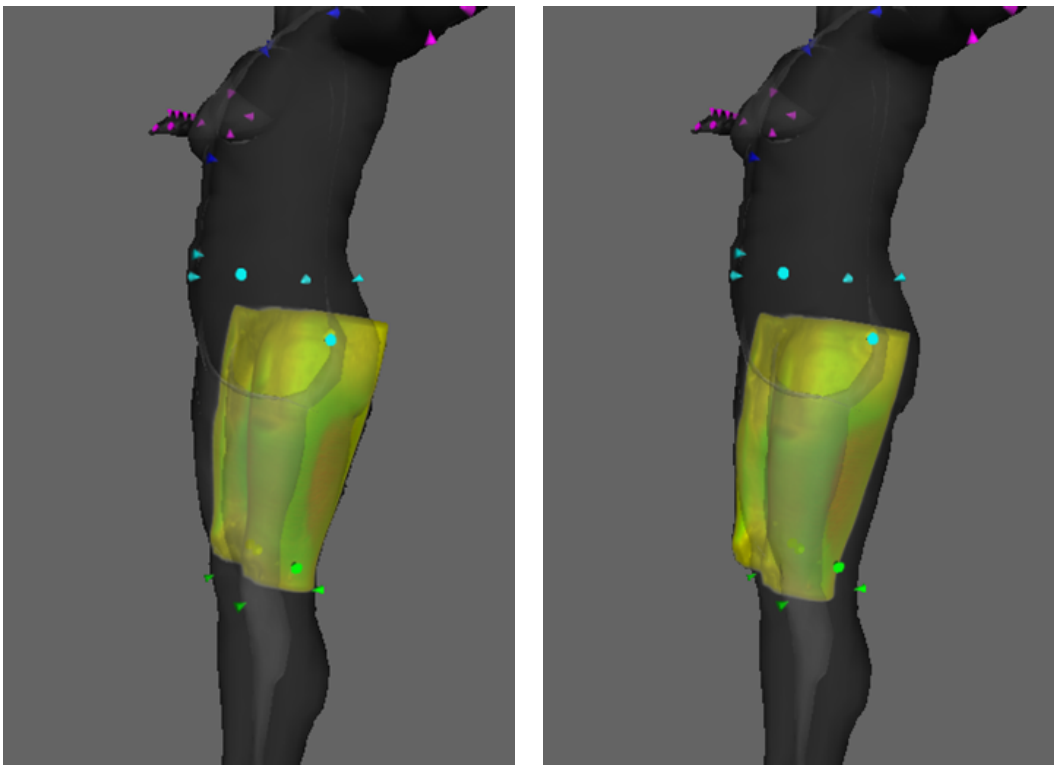


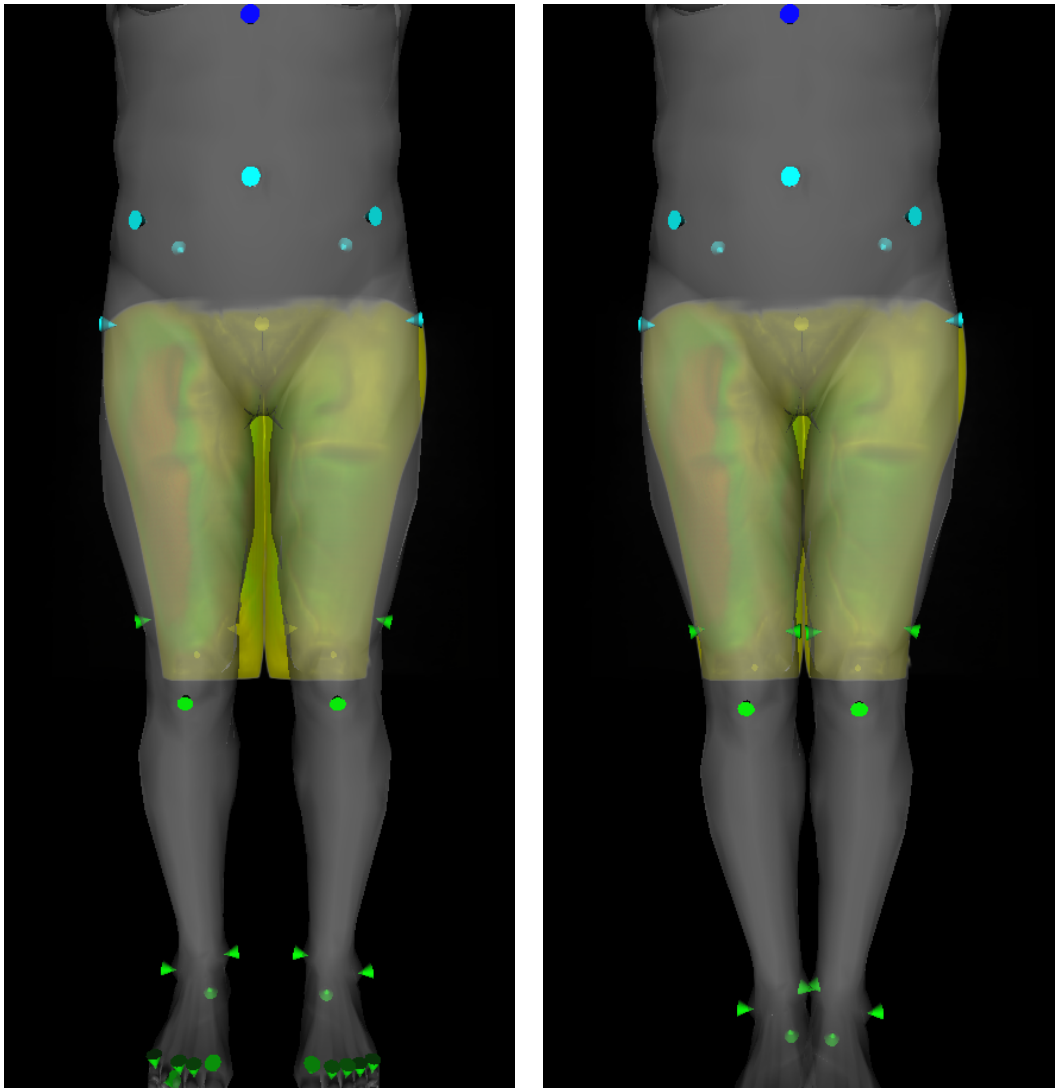
Figure 4.11: The figure shows the initial situation of volume to mesh registration. The mesh of the volume can be placed arbitrarily in the coordinate system of the reference model with any scale.



(a) Registration with markers

(b) Registration with ICP

Figure 4.12: In a) we see the result of the registration process with markers. The volume is placed roughly on the reference model; note that the transformation in this case is computed based on only four markers. The image in b) shows the result of the ICP registration. The transformation is calculated based on all points of the volume, in this case nearly 2000.



(a) Before adaption of the posture

(b) After adaption of the posture

Figure 4.13: The figure shows the benefits of adjusting the posture of the reference model. In a) we see the result after ICP registration but before adapting the posture and b) shows the result of the adaption process. As we can see the distance between reference model and volume has decreased.

Chapter 5

Image to Mesh Registration

Contents

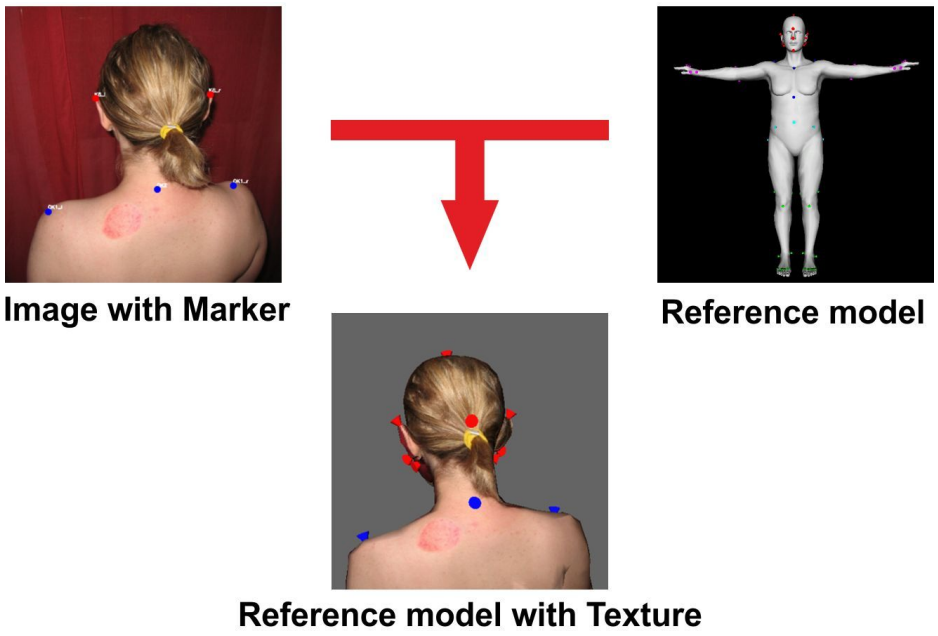


Figure 5.1: Overview of image to mesh registration. We register an image on the reference model based on corresponding markers.

In this Chapter we investigate registration of image with markers to the reference model from Chapter 3. Image to mesh registration is a sophisticated task because we need to bring a 2D-Image in accordance with a 3D-Mesh. There exist various concepts to reconstruct shape from images. For example shape from stereo or shape from motion. The methods require more than one image. We are restricted to use one image. Furthermore we keep the image acquisition process as simple as possible therefore we do not use a sophisticated setup. In general the images are taken by forensic experts but we also want to use images from other sources. For instance another source is a policeman who already took pictures of injuries which might not be available anymore at the time the forensic expert looks at the subject.

Because we also use existing images we cannot take advantage of camera calibration. The benefit of a calibrated setup has to be investigated in future work.

In Chapter 4 we were able to calculate transformations based on the points of the 3D-Volume and the 3D-Mesh. Now we have to deal with a 2D-Image instead of a 3D-Volume. The goal is to find a rotation and a translation that brings the 3D-Mesh in accordance with the 2D-Image. In Computer Vision this is called the 2D-3D pose estimation problem which consists of estimating the relative position and orientation of a 3D-Object according to a reference camera. Moreover we know that the image shows us an image of a human but we do not know the posture of the human. Therefore we have to recover the posture from the 2D-Image as well.

We have markers on our 3D-Mesh and specify the corresponding marker in the 2D-Image. Based on the corresponding marker we will firstly engage the 2D-3D pose estimation problem (Section 5.2) and secondly estimate the posture (Section 5.3). As final result we calculate texture coordinates (Section 5.4) to apply the 2D-Image as a texture on the 3D-Mesh in the same position as shown in the 2D-Image.

Since scaled orthographic projection (SOP) is important for the algorithms in Section 5.2 and Section 5.3 we give at first a brief description in Section 5.1.

5.1 Scaled orthographic projection (SOP)

Scaled orthographic projection (SOP) is a way of representing a 3D-Object in a 2D-Image. It is a type of parallel projection, where the view direction is orthogonal to the projection plane. The projected 3D-Points are scaled after parallel projection with scale factor s. The distance from the image plane has no influence on the size.

We can define a reference point M_0 which has the same image m_0 under SOP and

perspective projection. Based on the reference point M_0 the scale parameter s can be defined as the ratio between focal length f and Z_0, which is the distance between the center of projection O and the reference point M_0.

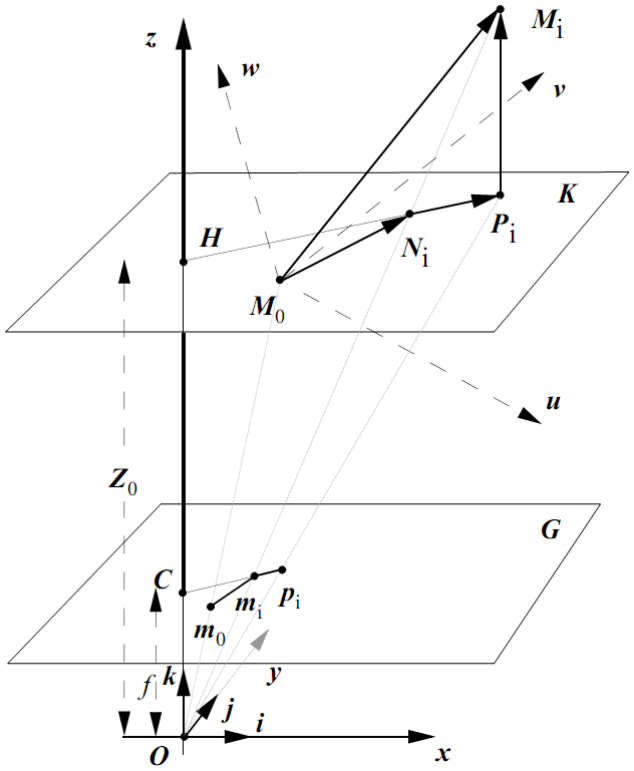


Figure 5.2: Perspective projection m_i and scaled orthographic projection p_i of point M_i (Image from [6]).

If we assume that the depths Z_i of different points M_i of the object coordinates do not differ in a significant way, we can use scaled orthographic projection (SOP) as an approximation of perspective projection. In other words there should be no strong perspective event in the image.

Scaled orthographic projection of a point M_i:

$$x'_i = fX_i/Z_0, \quad y'_i = fY_i/Z_0, \tag{5.1}$$

Perspective projection of a point M_i:

$$x_i = fX_i/Z_i, \quad y_i = fY_i/Z_i, \tag{5.2}$$

If all points Z_i equal Z_0 there would be no difference between SOP Eq. (5.1) and perspective projection Eq. (5.2).

In Figure 5.2 we can see the difference between SOP and perspective projection. The reference point M_0 has the same image m_0 under SOP and perspective projection whereas point M_i has image m_i under perspective projection and image p_i under SOP.

5.2 2D-3D pose estimation problem

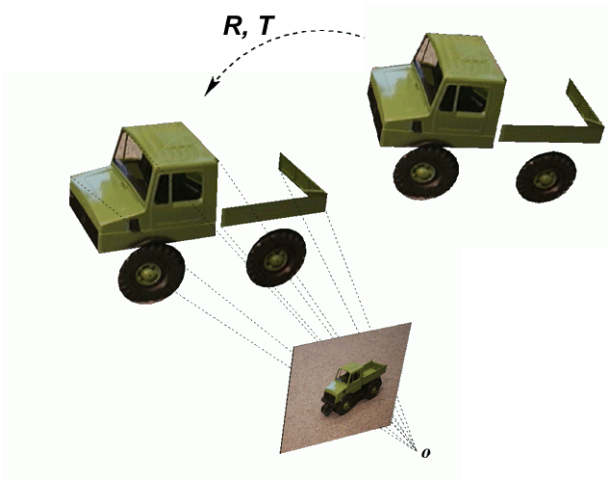


Figure 5.3: An illustration of the 2D-3D pose estimation problem. The goal is to obtain $\boldsymbol{R}$ and $\boldsymbol{T}$ with given 2D-Image and 3D-Model (Image from [24]).

In this Section we deal with the 2D-3D pose estimation problem. Figure 5.3 illustrates this problem. Basically we estimate the rotation R and the translation t of a 3D-Model so that the image of the reference camera equals the 2D-Image. Since we use markers we are able to use an algorithm to solve the 2D-3D pose estimation problem which is based on correspondences. We use an analytic formulation of the POSIT (Pose from Orthography and Scaling with Iteration) algorithm from [6]. This analytic formulation is presented in [5]. The main advantage of this analytic formulation is that due to the homogeneous form it does not need to locate the image of the origin M_0. In the following we describe the POSIT algorithm in more detail.

5.2.1 Notation and Problem Definition

We have markers $(M_0, M_1, \ldots, M_i, \ldots, M_n)$ on our 3D-Mesh which are located in the field of view of the camera and corresponding images of the markers $(m_0, m_1, \ldots, m_i, \ldots, m_n)$

in the 2D-Image. The object coordinate frame is $(M_0\boldsymbol{u}, M_0\boldsymbol{v}, M_0\boldsymbol{w})$. Coordinates of M_i (U_i, V_i, W_i) and coordinates of its image m_i (x_i, y_i) are known.

In the pinhole camera model we have a set of parameters, the center of projection O, the distance between O and the image plane f and axes O_x, O_y pointing along the rows and columns of the camera sensor and O_z pointing along the optical axis. The unit vectors for these three axes are called $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$. This gives us the camera coordinate system.

To obtain the coordinates of an object M_i in the camera coordinate system we need to multiply M_i with the rotation matrix $\boldsymbol{R}$ and add the translation vector $\boldsymbol{T}$. By using homogeneous coordinates we can compose $\boldsymbol{R}$ and $\boldsymbol{T}$ to the pose matrix $\boldsymbol{P}$.

$$P = \left[\begin{array}{ccc|c} & \boldsymbol{R} & & \boldsymbol{T} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \begin{array}{cccc} i_u & i_v & i_w & T_x \\ j_u & j_v & j_w & T_y \\ k_u & k_v & k_w & T_z \\ 0 & 0 & 0 & 1 \end{array} = \begin{array}{c} \boldsymbol{P_1^T} \\ \boldsymbol{P_2^T} \\ \boldsymbol{P_3^T} \\ \boldsymbol{P_4^T} \end{array} \tag{5.3}$$

The coordinates of M_i or vector $\boldsymbol{M_0M_i}$ in the camera coordinate system can now be calculated by simply multiplying M_i or vector $\boldsymbol{M_0M_i}$ with $\boldsymbol{P}$. This requires that M_i or vector $\boldsymbol{M_0M_i}$ is provided in homogeneous coordinates which is done by adding 1 as fourth coordinate to M_i or vector $\boldsymbol{M_0M_i}$.

According to Eq.5.3 we name the rows of $\boldsymbol{P}$, $\boldsymbol{P_1}$, $\boldsymbol{P_2}$, $\boldsymbol{P_3}$ and $\boldsymbol{P_4}$. $\boldsymbol{P_1}$ consists of the coordinates i_u, i_v, i_w, they are the first row of the rotation matrix $\boldsymbol{R}$ which is the x-axis of the camera coordinate system expressed in the object coordinate system. Coordinates j_u, j_v, j_w of $\boldsymbol{P_2}$ are the y-axis of the camera coordinate system expressed in object coordinate system $(M_0\boldsymbol{u}, M_0\boldsymbol{v}, M_0\boldsymbol{w})$. Vector $\boldsymbol{k}\,(k_u, k_v, k_w)$ made from the first three elements of $\boldsymbol{P_3}$ is the cross product of the vectors $\boldsymbol{i} = (i_u, i_v, i_w)$ and $\boldsymbol{j} = (j_u, j_v, j_w)$.

5.2.2 Fundamental Equations and POSIT

The fundamental equations relate the row vectors $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$, the coordinates of the object vector $\boldsymbol{M_0M_i}$ and coordinates (x_i, y_i) of the perspective images m_i from M_i.

$$\begin{aligned} \boldsymbol{M_0M_i} \cdot \boldsymbol{I} &= x_i' \\ \boldsymbol{M_0M_i} \cdot \boldsymbol{J} &= y_i' \end{aligned} \tag{5.4}$$

with

$$\boldsymbol{I} = \frac{f}{T_z}\boldsymbol{P_1}, \quad \boldsymbol{J} = \frac{f}{T_z}\boldsymbol{P_2}, \tag{5.5}$$

$$x_i' = x_i\,(1+\epsilon_i)\,, \quad y_i' = y_i\,(1+\epsilon_i) \tag{5.6}$$

and

$$\epsilon_i = \boldsymbol{M_0 M_i} \cdot \boldsymbol{P_3}/T_z - 1 \tag{5.7}$$

The coordinates x_i' and y_i' are the scaled orthographic images p_i from M_i (see 5.1). To calculate ϵ_i we need $\boldsymbol{P_3}$ (Eq.(5.7)) which can be computed only after $\boldsymbol{I}$ and $\boldsymbol{J}$ have been computed.

We get Eq.(5.7) by assuming that the projective projection (Eq.(5.2)) is related to SOP (Eq: (5.1)) through ϵ_i. We relate the SOP of the x-coordinate X_i of M_i that is $x_i' = fX_i/Tz$ to the projective projection with Eq.(5.6). This leads to:

$$\begin{aligned} \frac{1}{(1+\epsilon_i)}\,\frac{fX_i}{T_z} &= \frac{fX_i}{Z_i} \\ \Rightarrow (1+\epsilon_i)\,T_z &= Z_i \end{aligned} \tag{5.8}$$

We obtain the z-coordinate Z_i of M_i as the dot product $\boldsymbol{M_0 M_i} \cdot \boldsymbol{P_3}$. Therefore we can write:

$$(1+\epsilon_i)\,T_z = \boldsymbol{M_0 M_i} \cdot \boldsymbol{P_3} \tag{5.9}$$

which leads to Eq.(5.7).

With a given ϵ_i we can solve the linear equation system provided by (Eq.(5.4)) to obtain $\boldsymbol{I}$ and $\boldsymbol{J}$. Once we know $\boldsymbol{I}$ and $\boldsymbol{J}$ we can compute $\boldsymbol{P}$. From $\boldsymbol{I}$ we get $\boldsymbol{R_1}$ as (I_1, I_2, I_3). The norm of $\boldsymbol{R_1}$ is equal to f/T_z. Vector $\boldsymbol{i}$ is the normalized vector $\boldsymbol{R_1}$. We get $\boldsymbol{P_1}$ by dividing $\boldsymbol{I}$ through the norm of $\boldsymbol{R_1} = f/T_z$. With similar operations we obtain $\boldsymbol{j}$ and $\boldsymbol{P_2}$ from $\boldsymbol{J}$. With $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ we can compute $\boldsymbol{P_3}$ with $k = i \times j$, T_z is computed before. This algorithm to calculate one pose matrix $\boldsymbol{P}$ with a given ϵ_i is called POS (Pose from Orthography and Scaling). The solution of POS is only an approximation, if ϵ_i is not exact, but we can compute a better ϵ_i from a given solution. Applying POS and calculating a better ϵ_i in turns is called POSIT (POS with Iterations [5]).

Initially we assume $x_i' = x_i$, and $y_i' = y_i$ which implies $\epsilon_i = 0$. With $\epsilon_i = 0$ we assume that points M_i and P_i from Figure 5.2 coincide. We iterate until current ϵ_i is close enough to the ϵ_i from previous loop. Typically 4-5 iterations are sufficient.

5.2.3 Solving POS Equation System

Within the iteration we have to solve Eq.(5.4) with a given ϵ_i and known coordinates of M_i and m_i. We can define a matrix $\boldsymbol{A}$ with the object points M_i

$$A = \begin{matrix} U_0 & V_0 & W_0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ U_i & V_i & W_i & 1 \\ \vdots & \vdots & \vdots & \vdots \\ U_n & V_n & W_n & 1 \end{matrix} \tag{5.10}$$

to rewrite the equations in a more compact form:

$$\boldsymbol{AI} = \boldsymbol{x}', \quad \boldsymbol{AJ} = \boldsymbol{y}' \tag{5.11}$$

where $\boldsymbol{x}'$ is a column vector with i-th coordinate equal to x_i'.

There are four unknown coordinates in vectors $\boldsymbol{I}$ and $\boldsymbol{J}$, therefore matrix $\boldsymbol{A}$ must have at least rank 4 for the system to provide solutions. This requirement is satisfied with at least four non coplanar object points; also corresponding image points of the object points are required.

The solution of the equation system in a least squares sense are given by

$$\boldsymbol{I} = \boldsymbol{B}\boldsymbol{x}', \quad \boldsymbol{J} = \boldsymbol{B}\boldsymbol{y}' \tag{5.12}$$

where $\boldsymbol{B}$ is the *pseudo inverse* of matrix $\boldsymbol{A}$. The *pseudo inverse* $\boldsymbol{B}$ is calculated either by operation $\left[\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right]^{-1}\boldsymbol{A}^{\mathrm{T}}$ or by decomposing matrix $\boldsymbol{A}$ by Singular Value Decomposition (SVD). SVD has the advantage of giving diagnosis of the rank and condition of $\boldsymbol{A}$ as well as numerical advantages. The *pseudo inverse* $\boldsymbol{B}$ can be precomputed, because it depends only on the relative geometry of the object points.

5.2.4 Summary

This analytic formulation of the POSIT algorithm in homogeneous form allows us to find the pose of an object based on corresponding points between object feature points and image points without the need to locate the image of the origin.

Here are the steps of POSIT algorithm:

1. ϵ_i = best guess, or $\epsilon_i = 0$ if no pose information is available

2. Start of loop: Solve for $\boldsymbol{I}$ and $\boldsymbol{J}$ in the following system

$$\boldsymbol{M_0 M_i} \cdot \boldsymbol{I} = x_i', \quad \boldsymbol{M_0 M_i} \cdot \boldsymbol{J} = y_i'$$

with

$$x_i' = x_i \left(1 + \epsilon_i\right), \quad y_i' = y_i \left(1 + \epsilon_i\right)$$

3. From $\boldsymbol{I}$, get

$$\begin{aligned} \boldsymbol{R_1} &= \left(I_1, I_2, I_3\right), \\ f/T_z &= \left|\boldsymbol{R_1}\right|, \\ \boldsymbol{i} &= \left(T_z/f\right) \boldsymbol{R_1}, \\ \boldsymbol{P_1} &= \left(T_z/f\right) \boldsymbol{I} \end{aligned}$$

Similar operations yield $\boldsymbol{j}$ and $\boldsymbol{P_2}$ from $\boldsymbol{J}$.

4. $\boldsymbol{k} = \boldsymbol{i} \times \boldsymbol{j}, \quad \boldsymbol{P_3} = \left(k_u, k_v, k_w, T_z\right), \quad \epsilon_i = \boldsymbol{M_0 M_i} \cdot \boldsymbol{P_3}/T_z - 1$

5. If the current ϵ_i is close enough to the ϵ_i from the previous loop, EXIT, else go to step 2.

6. $\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{P_3}$ along with $\boldsymbol{P_4} = (0, 0, 0, 1)$, are the four rows of the pose matrix $\boldsymbol{P}$.

5.3 Estimation of Posture

We need to adapt the posture of our model in order to improve the accuracy. With the 3D-Position of the 2D-Marker we can use the methods from Section 4.5. To estimate the 3D-Position we use the method described in [28]. This method uses SOP (Section 5.1).

The method takes the foreshortening between each of the body segments in the model and its image into account. In Figure 5.4 we see a line segment of known length l projected onto the image under SOP.

The two end points are represented by (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2); their images are represented by (u_1, v_1) and (u_2, v_2). If we know the scale factor s, it would be easy to compute the relative depth between the two endpoints with following equations.

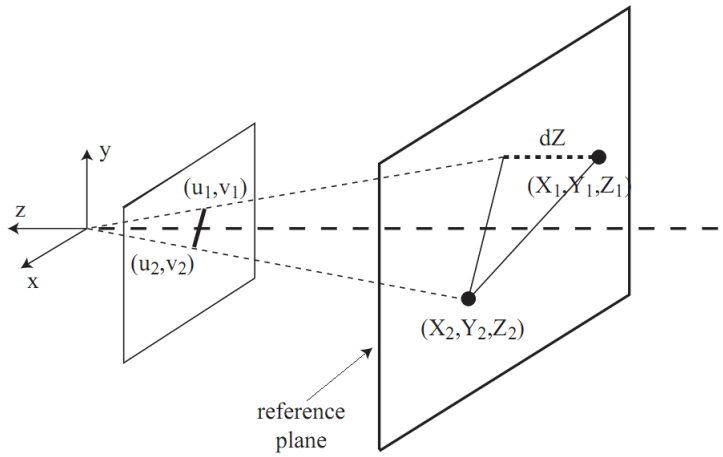


Figure 5.4: Projection of a line segment onto an image under SOP (Image from [28]).

$$\begin{aligned} l^2 &= (X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2 \\ (u_1 - u_2) &= s\,(X_1 - X_2) \\ (v_1 - v_2) &= s\,(Y_1 - Y_2) \\ dZ &= (Z_1 - Z_2) \\ \Rightarrow dZ &= \sqrt{l^2 - \left((u_1 - u_2)^2 + (v_1 - v_2)^2\right)/s^2} \end{aligned} \tag{5.13}$$

For a given value of s there are still two distinct solutions possible. Either point 1 or point 2 could have the smaller z-coordinate. We can obtain a lower bound on the scale factor s since dZ cannot be a complex number and therefore the quantity under the square root must be non-negative which leads to the following inequality:

$$s \geq \frac{\sqrt{\left((u_1 - u_2)^2 + (v_1 - v_2)^2\right)}}{l} \tag{5.14}$$

The scale parameter s directly affects the result. The larger s the greater the foreshortening of each segment and the more stretched the solution along the z-axis. Figure 5.5 shows this behaviour.

The relations can be extended to the case of a jointed mechanism; for example a kinematic chain of four points and three line segments. By using Eq.(5.14) with every line segment we can obtain an overall minimum scale s. Still we have the problem of the ambiguities of dZ. In this example we get eight possible solutions for three line segments. Therefore the user needs to specify which segment is closer.

Figure 5.5: Impact of the scale factor s on the estimation. The scale is increasing from left to right. Image from [28]

.

5.4 Texture Coordinates

The calculation of the texture coordinates is straight forward. At first we transform the 3D-Mesh with the rotation $\boldsymbol{R}$ and translation $\boldsymbol{T}$ found in Section 5.2. Then we determine which points of the 3D-Mesh will be visible in the image. We take the dot product of the camera view vector $\boldsymbol{c}$ with the normal $\boldsymbol{n_i}$ of each point. A negative dot product denotes that the normal vector points away from the camera, therefore the point can not be seen. These points are ignored for further computation. For example if the image shows us the front side of a person we cannot see the backside therefore we can omit the points of the backside for further calculations. Normal vectors of the backside will give us a negative dot product whereas the dot product with normal vectors of the front side is positive.

The remaining points are projected on the image plane with perspective projection. Points which are outside the image limits are ignored because they are not visible in the image. The remaining projected points are used as texture coordinates. Since we use the Coin library[i] we need to make sure that the texture coordinates are within $[0, 1]$.

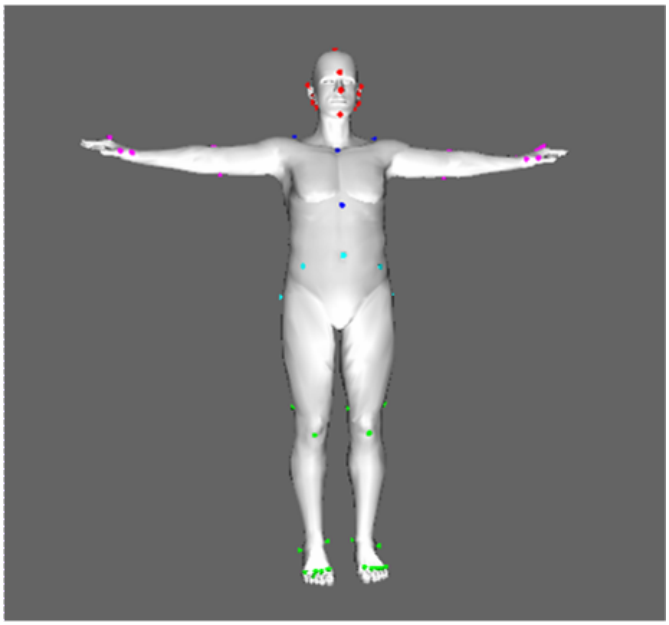
5.5 Images of example data

We present a test case to show the image of the mesh registration process. The first Figure 5.6 shows us the input data. The input consists of an image with specified markers and the reference model from Chapter 3.

The second Figure 5.7 shows the result of the image to mesh registration process. We see the original image with the projected points of the reference model, which we use as texture coordinates, and we see the reference model with the original image as texture.

[i] `http://www.coin3d.org`

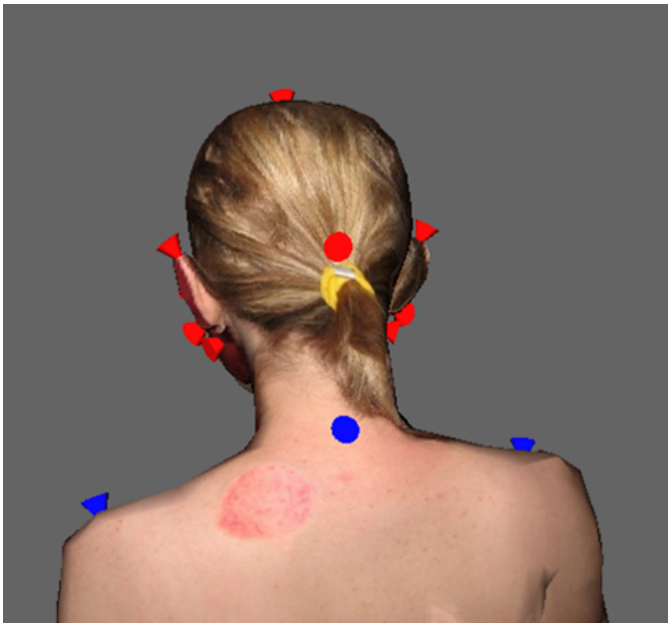
(a) Image with markers



(b) Reference model

Figure 5.6: Initial state of image to mesh registration. In a) we see the image with specified marker and b) shows us the initial reference model.

(a) Image with Marker and texture coordinates



(b) Reference model with the image as texture

Figure 5.7: Result of image to mesh registration. In a) we see the image with marker and texture coordinates as light blue dots and b) shows us the 3D-Model were the image is applied as texture.

Chapter 6

Experiments

Contents

We perform experiments for volume to mesh registration and for image to mesh registration. The experiments with volume to mesh registration are carried out with a volumetric data set in the analyze file format (**.hdr* and **.img* files). This data set contains a human body from knee to hip with resolutions $x = 1.11607$, $y = 1.11607$, $z = 6$. In addition to the volumetric data set we specify the marker positions within the volumetric data set.

At first we investigate the influence of the mesh simplification step within the volume to mesh registration process in Section 6.2.1. Therefore we vary the target number of faces of the mesh simplification step while all other parameters and the marker positions stay constant.

Secondly we explore the capture range of the ICP algorithm within the volume to mesh registration process in Section 6.2.2. We add noise of different magnitude to the marker positions while all other parameters remain constant. The capture range of the ICP algorithm is the one over which the ICP algorithm converges to the same minimum as without noise.

For experiments with image to mesh registration we use images in the JPEG File Interchange format (**.jpg*). We specify the marker positions within each image manually.

Then we investigate the impact of three error sources. First, we perform experiments

to show the influence of noisy marker positions by adding noise of different magnitudes to the marker positions within the image. Secondly we show the influence of the posture by using images where the posture is different from the posture of the reference model. Thirdly we explore the impact of perspective by using images with different perspective.

6.1 Compare Transformations

Both, volume to mesh registration and image to mesh registration, result in a transformation. Therefore we need to be able to compare transformations which consist of rotation, translation and scale in a meaningful manner. For graphical comparison we transform a regular tetrahedron (see Figure 6.1) with each transformation. The center of the untransformed tetrahedron is at $(0, 0, 0)$.

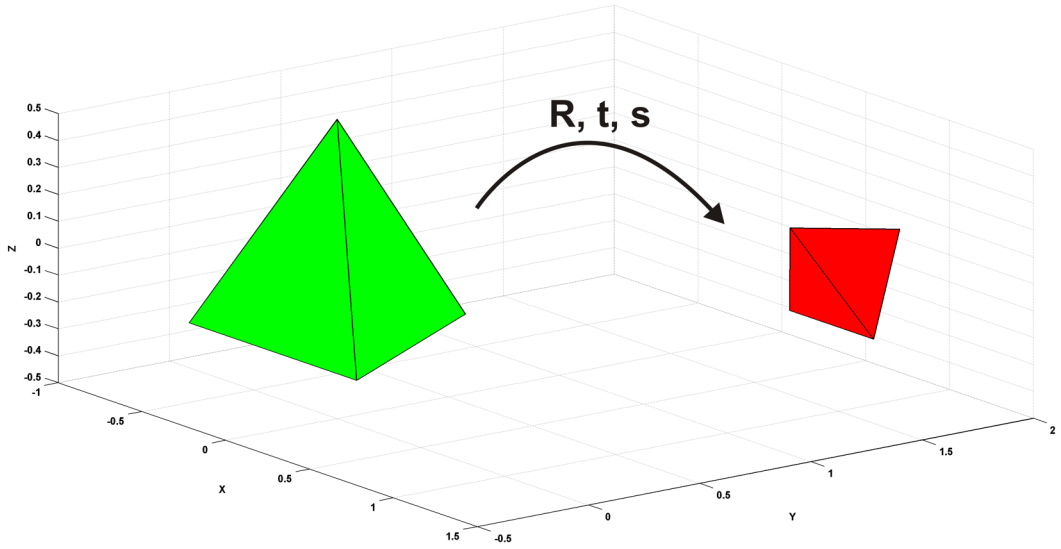


Figure 6.1: The base tetrahedron is transformed according to the given rotation, translation and scale.

We want to calculate an error value ϵ between the transformations. We compute the rotation error ϵ_R between two rotations as the minimal angle between the two quaternions representing each rotation. For the translation error ϵ_t we use the distance between the two translation vectors. We use isometric scaling therefore we compute the scale error ϵ_s as ratio between the two scale factors. The error value ϵ is calculated as weighted sum of rotation error ϵ_R, translation error ϵ_t and scale error ϵ_s. Therefore we specify

corresponding weights w_R, w_t and w_s for each of the three error values. Eq.(6.1) shows how we calculate the error value ϵ.

$$\epsilon = w_R * \epsilon_R + w_t * \epsilon_t + w_s * \epsilon_s \tag{6.1}$$

with

$$w_R = 90/10 \quad w_t = 1 \quad w_s = 10 \tag{6.2}$$

The weights of Eq.(6.2) denote that a rotation of 90° equals a translation of 10 and a translation of 1 equals a scale change of 10%.

6.2 Volume to Mesh Registration Experiments

6.2.1 Mesh Simplification

The reference mesh has in general a lower resolution than the output of the Marching Cubes algorithm (Section 4.1). Therefore we investigate how much we can simplify the mesh until the error is not tolerable. For example a difference of 5 cm is not tolerable. A simpler mesh reduces the computational effort of the following steps but the result has to be close to the result of the original mesh.

We simplify the mesh of the volume as shown in Section 4.2. We define the simplification factor sf between 0 and 1. The target number of faces is the product of the simplification factor sf and the original number of faces.

After the Marching Cubes step we have a mesh with 95234 vertices and 188122 faces. At first we create different meshes by reducing the number of faces of the volume mesh with the simplification factors sf of Table 6.1. A selection of the meshes is shown in Figure 6.2. Marker based registration depends only on the markers and not on the mesh therefore marker based registration gives us the same transformation for every mesh. More interesting is the registration with ICP (Section 4.4). Since ICP depends on the points of the mesh we get a different transformation depending on the simplification factors sf. We use the transformation of the original mesh as basis. The other transformations are compared to this basis.

We can see the results of the experiment in Figure 6.3 and in Table 6.2. Based on the result we can see that the error of a simplification factor sf of 1/128 is still tolerable. With simplification factors sf smaller than 1/128 the error increases rapidly. We suspect

Simplification Factor sf	**Number of Faces**
1/1	188122
1/2	94061
1/4	47031
1/8	23515
1/16	11758
1/32	5879
1/64	2939
1/128	1470
1/256	735
1/512	367
1/1024	184

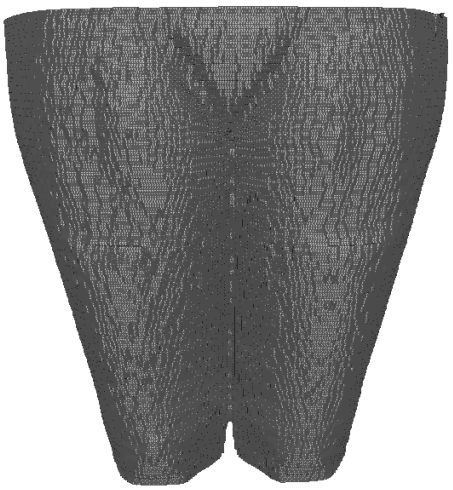
Table 6.1: Simplification Factors for the test volume.

Simplification Factor sf	**Rotation Error**	**Translation Error**	**Scale Error**	**Weighted Sum**
1/1	0.000000	0.000000	0.000000	0.000000
1/2	0.820824	0.085293	0.016514	**0.341636**
1/4	0.783481	0.071310	0.007573	**0.234094**
1/8	0.750588	0.048350	0.003697	0.168721
1/16	0.550236	0.053482	0.002468	0.139301
1/32	0.695058	0.073364	0.004133	0.191927
1/64	0.795949	0.070860	0.001051	0.169806
1/128	0.782260	0.106230	0.000099	**0.194139**
1/256	0.814548	0.333537	0.022062	0.644662
1/512	3.155333	0.397813	0.032766	1.076069
1/1024	3.259726	0.407875	0.038479	1.154855

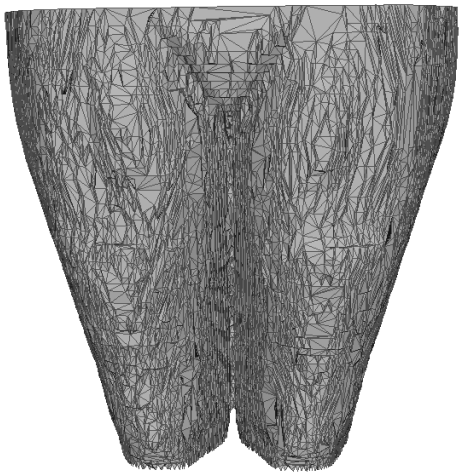
Table 6.2: Relation between mesh simplification factor and the difference of the resulting transformation.

the increased error values at simplification factors sf at 1/2 and 1/4 arise from the fact that the high resolution of the mesh causes the ICP algorithm to get trapped in a local minimum.

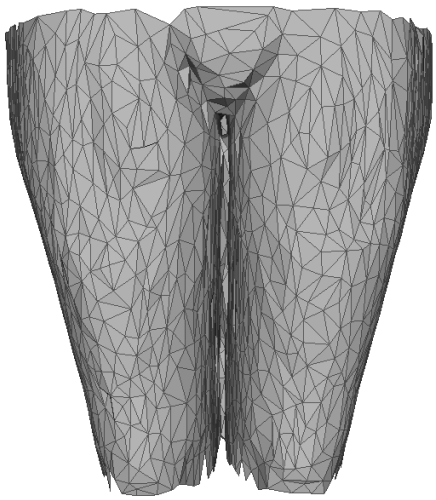
With mesh simplification we can speed up the registration process but a too small simplification factor sf gives us a mesh which cannot represent the original mesh good enough to give satisfying results. The simplification factors sf has to be selected so that the mesh has about the equal or greater resolution than the reference model in order to get a good result. Based on this knowledge we use a simplification factor sf of 1/64 in the experiments of Section 6.2.2.



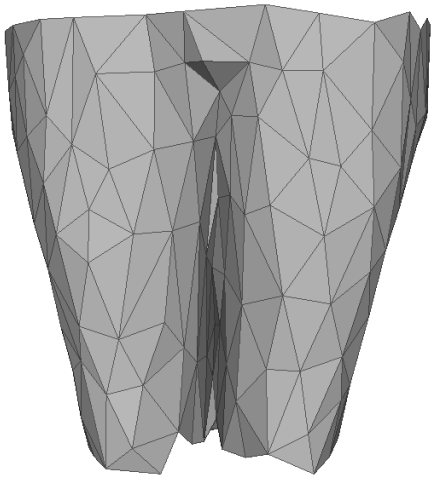
(a) 188122 faces, sf = 1/1



(b) 23515 faces, sf = 1/8

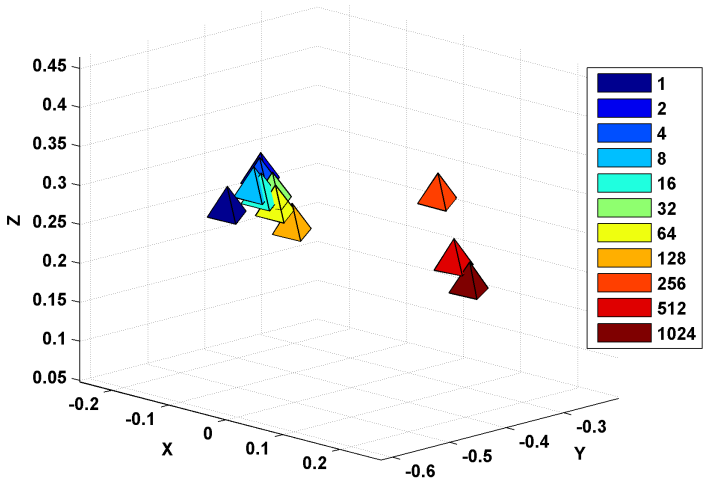


(c) 2939 faces, sf = 1/64

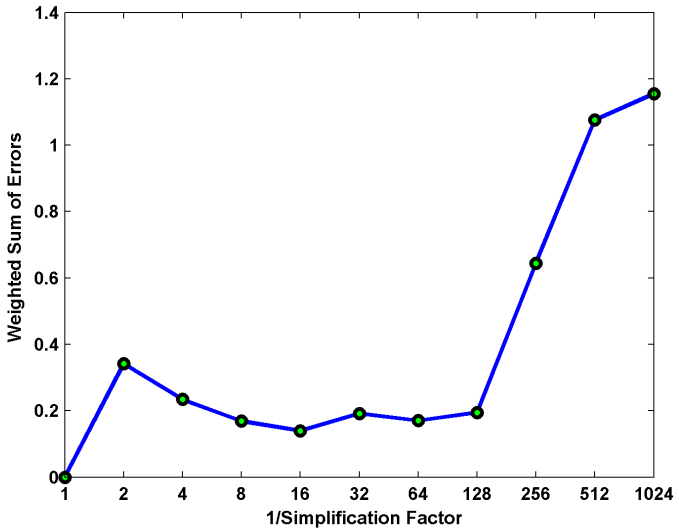


(d) 367 faces, sf = 1/512

Figure 6.2: The image shows the original mesh and reduced meshes with different simplification factors.



(a) tetrahedron



(b) error

Figure 6.3: The figure shows the comparison of the resulting transformation of the mesh simplification experiment. Each transformation is obtained from a volume with a different simplification factor. As we can see there is no significant difference until the simplification factor decreases under 1/128.

6.2.2 ICP capture range

Since marker positions are not perfectly accurate we investigate the impact of inaccurate marker positions on the result of volume to mesh registration. We assume that the ICP algorithm has robustness to noise. This leads us to the question how much tolerance of the marker position we have got until ICP cannot calculate the correct transformation.

We deactivate the calculation of a first estimate of the posture because of the different estimated postures we cannot perform an objective comparison of the transformations.

First we calculate normally distributed random numbers between $[-1, 1]$. We add the random numbers to the marker positions multiplied by a factor N. This factor N is the maximal variation of the marker position in millimetre.

In this experiment we used the same volumetric data set as before which contains the body from knee to hip. We added noise of different magnitudes to the marker positions. Figure 6.4 displays all marker positions we have used in this experiment. The result of the comparison of the transformation obtained for each factor N can be seen in Figure 6.5 and in Figure 6.6. Table 6.3 and Table 6.4 present the error values.

After marker registration there is a large difference between the transformations. The error increases with increasing noise. Whereas after registration with ICP the error is about ten times smaller than after marker registration. With the ICP algorithm we have robustness against inaccurate marker positions but, nevertheless, more accurate marker positions give better result.

Result after marker registration				
N	**Rotation Error**	**Translation Error**	**Scale Error**	**Weighted Sum**
10	1.577818	0.163456	0.025791	0.596677
10	0.964197	0.124938	0.029813	0.530201
10	1.452554	0.130405	0.018435	0.476149
10	1.455480	0.121373	0.014965	0.432747
10	1.282198	0.116281	0.017393	0.432680
20	1.827105	0.268523	0.006187	0.533407
20	1.873319	0.306805	0.067625	1.191203
20	1.748970	0.239011	0.037613	0.809469
20	2.138322	0.250056	0.049027	0.977918
20	3.726052	0.191777	0.007365	0.679432
30	6.202536	0.323094	0.019377	1.206036
30	4.818383	0.552186	0.073504	1.822606
30	2.281566	0.101149	0.041581	0.770462
30	1.484033	0.328976	0.094744	1.441309
30	4.514871	0.283748	0.011351	0.898909
40	5.295873	0.747185	0.137440	2.710013
40	2.067709	0.313200	0.008434	0.627284
40	4.678655	0.261945	0.015581	0.937610
40	7.386098	0.870024	0.117775	2.868457
40	3.250382	0.466167	0.060713	1.434451
50	5.272520	1.283192	0.305839	4.927422
50	8.932774	0.655299	0.132064	2.968471
50	10.763595	0.526612	0.090617	2.628734
50	6.647253	0.701616	0.101365	2.453851
50	6.760129	1.044667	0.175954	3.555331
Mean Values				
10	1.346450	0.131290	0.021279	**0.493691**
20	2.262753	0.251234	0.033563	**0.838286**
30	3.860278	0.317831	0.048111	**1.227864**
40	4.535743	0.531704	0.067989	**1.715563**
50	7.675254	0.842277	0.161168	**3.306762**

Table 6.3: Result after marker registration. The error values refer to the transformation without noise $N = 0$. The last column is the outcome of Eq.(6.2) for each row.

Result after ICP registration				
N	**Rotation Error**	**Translation Error**	**Scale Error**	**Weighted Sum**
10	0.187669	0.013682	0.000846	0.042992
10	0.132475	0.019911	0.004470	0.079328
10	0.203483	0.013086	0.000915	0.044848
10	0.102714	0.004384	0.000269	0.018484
10	0.057964	0.012969	0.002898	0.048394
20	0.214888	0.026502	0.005243	0.102807
20	0.313666	0.024809	0.002250	0.082157
20	0.221043	0.013892	0.001749	0.055941
20	0.126051	0.020338	0.004131	0.075652
20	0.194289	0.029370	0.006463	0.115587
30	0.228828	0.030679	0.006824	0.124349
30	0.897072	0.106778	0.017949	0.385942
30	0.170432	0.013426	0.001759	0.049957
30	0.155046	0.012180	0.001796	0.047368
30	0.846114	0.109351	0.009221	0.295571
40	0.351065	0.045728	0.002206	0.106793
40	0.228418	0.018386	0.003722	0.080981
40	0.962163	0.115971	0.013475	0.357631
40	0.222287	0.023212	0.004298	0.090896
40	0.225681	0.029611	0.005061	0.105301
50	0.897020	0.237397	0.017560	0.512669
50	0.214081	0.016142	0.001445	0.054377
50	0.427790	0.042784	0.008307	0.173387
50	0.259496	0.025929	0.003665	0.091411
50	0.454745	0.040978	0.003626	0.127763
Mean Values				
10	0.136861	0.012806	0.001880	**0.046809**
20	0.213987	0.022982	0.003967	**0.086429**
30	0.459498	0.054483	0.007510	**0.180637**
40	0.397923	0.046582	0.005753	**0.148320**
50	0.450626	0.072646	0.006921	**0.191921**

Table 6.4: Result after ICP registration. The error values refer to the transformation without noise $N = 0$. The last column is the outcome of Eq.(6.2) for each row.

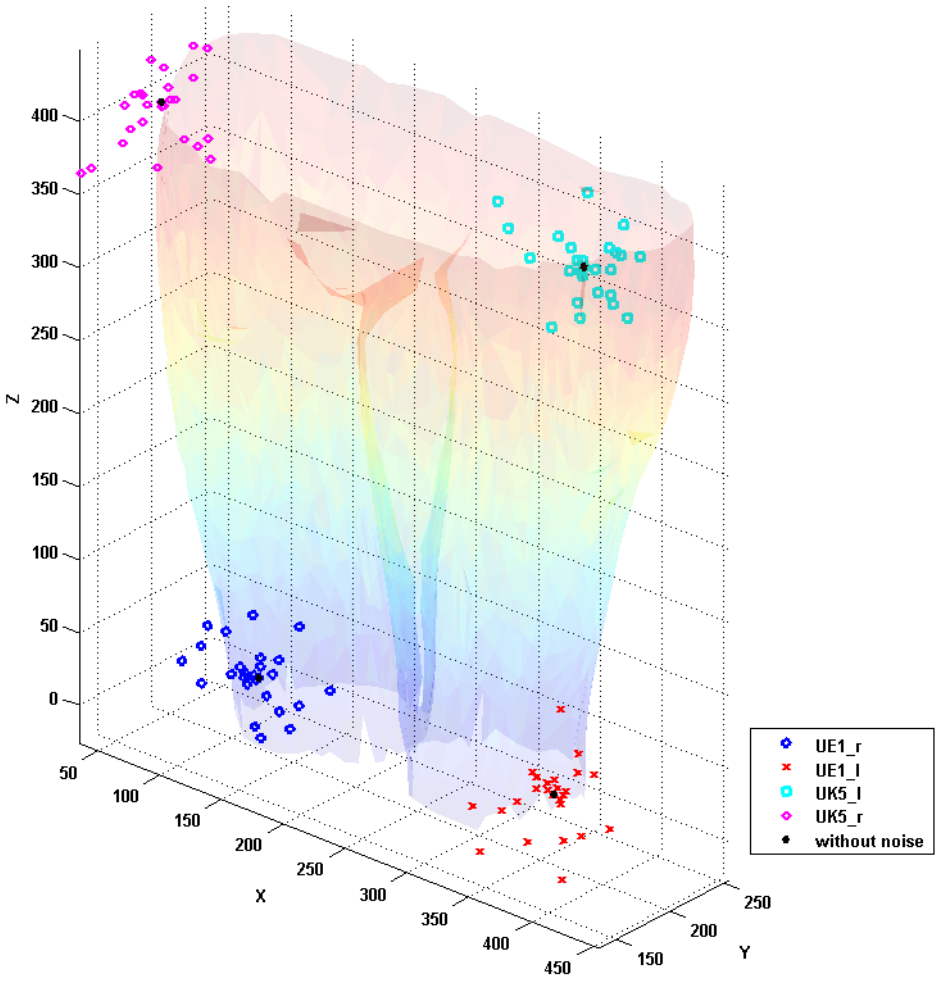


Figure 6.4: Here we can see all marker point sets we used in this experiment. The marker point set without noise is coloured black.

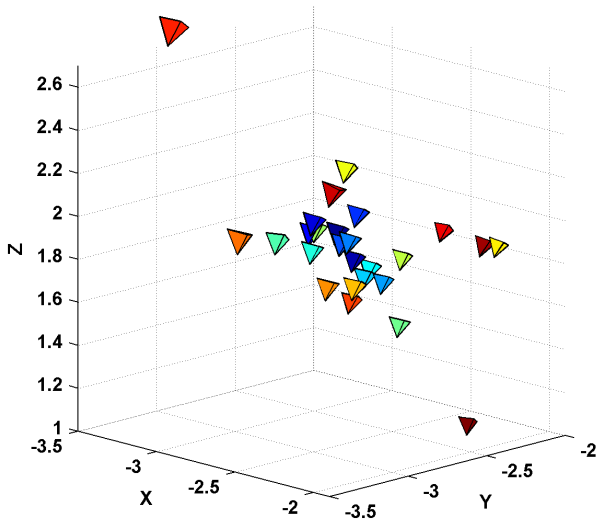
6.3 Image to Mesh Registration

In this Section we perform experiments with image to mesh registration. In the figures of this Section are images with light blue dots. These dots are the texture coordinates of the reference model.

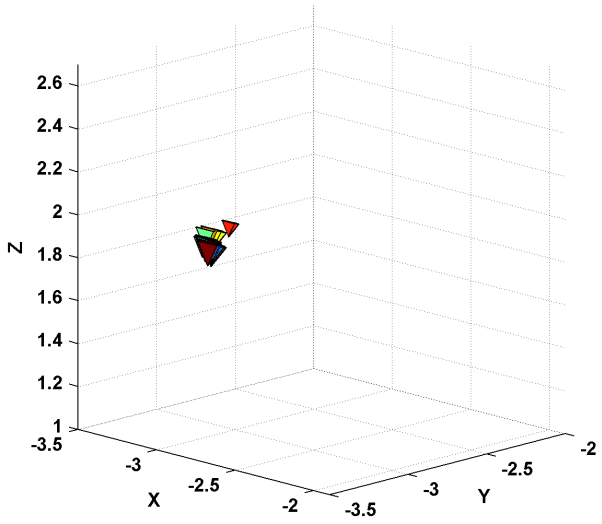
6.3.1 Noise

We assume that the POSIT algorithm incorporates robustness against noise like the ICP algorithm since the marker locations are not perfectly accurate.

We created an image of the reference model therefore it is simple to locate the markers

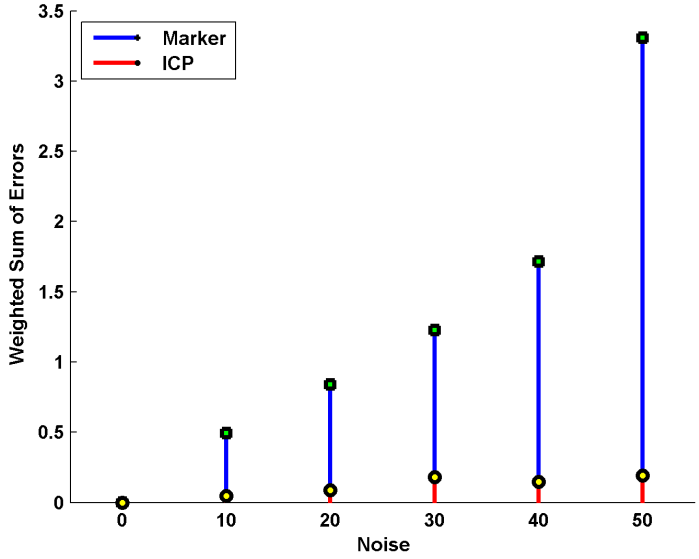


(a) tetrahedron after marker registration

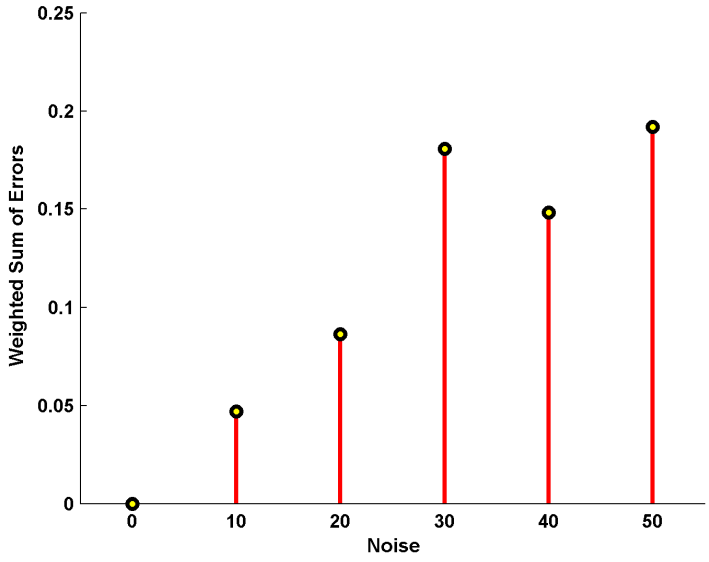


(b) tetrahedron after ICP registration

Figure 6.5: The figure shows the transformed tetrahedrons for each marker after the marker registration step in image a) and after the ICP registration step in image b). After marker registration the tetrahedrons are scattered but after ICP registration they are clustered. That denotes that the difference of the transformations after ICP registration is small.



(a) weighted sum error after marker and ICP registration



(b) weighted sum error after ICP registration

Figure 6.6: Comparison of the transformations after marker registration and ICP registration. Both figures show the weighted sum error for the mean values for each factor N. In image a) the weighted sum error after marker registration and ICP registration. Image b) shows the weighted sum error after ICP registration at a larger scale. The error after ICP registration is about ten times smaller.

in the image. Based on this image we estimate the pose without noise; the result is used as a reference. The noise consists of normal distributed random numbers between $[-1, 1]$ which we multiply with a maximal dislocation in pixels. We add the noise to the marker positions, Figure 6.7 shows the noisy marker locations. Subsequently we estimate the pose with the POSIT algorithm (Chapter 5).

Figure 6.8 shows us the comparison of the transformation and Figure 6.9 shows the difference between the marker points of the image and the projected marker points of the reference model.

Since the POSIT algorithm only depends on the marker points, noise directly affects the result. Therefore the marker points have to be obtained with the highest possible accuracy.

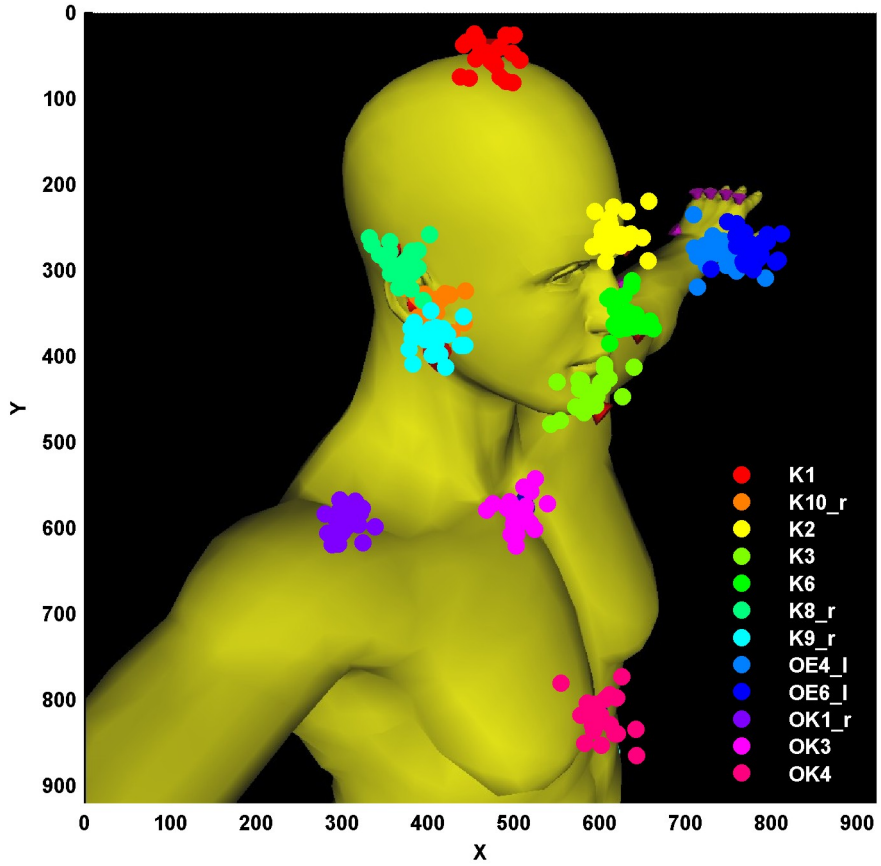
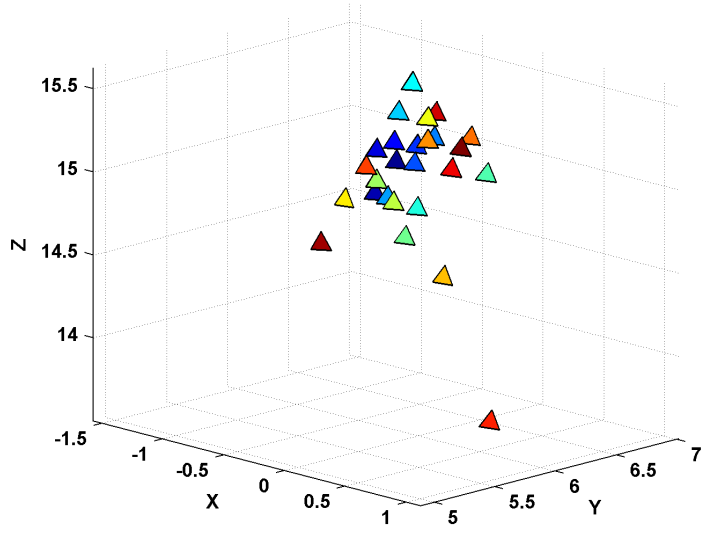
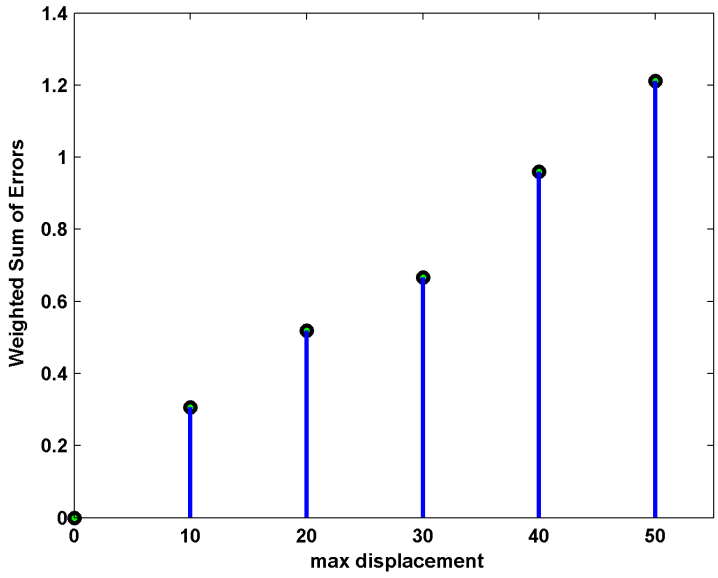


Figure 6.7: The figure shows the used test image and the different marker positions with applied noise.



(a) tetrahedron



(b) error

Figure 6.8: Comparison of the transformations after POSIT with different noise levels. The noise level is defined as the highest possible displacement in pixel.

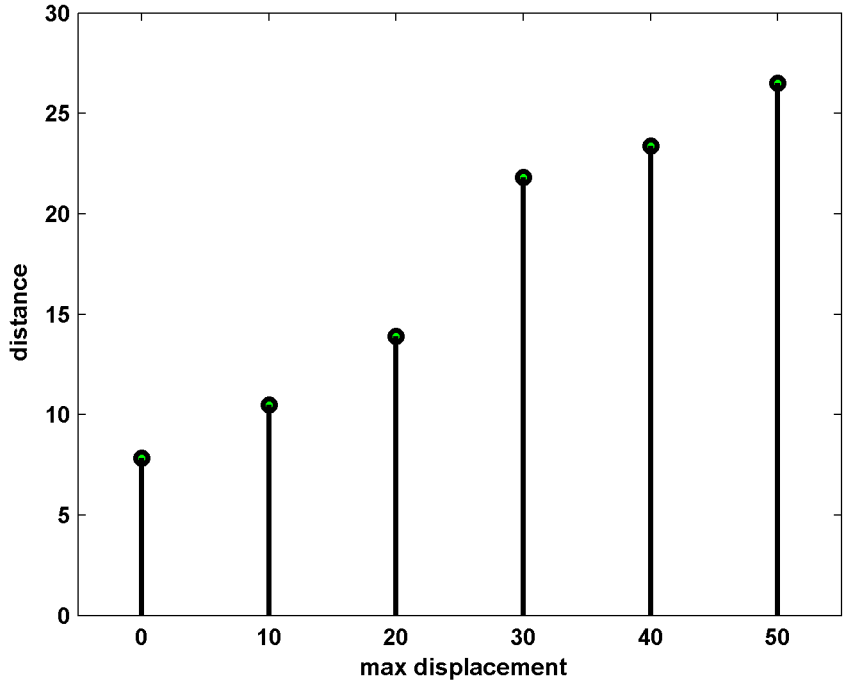


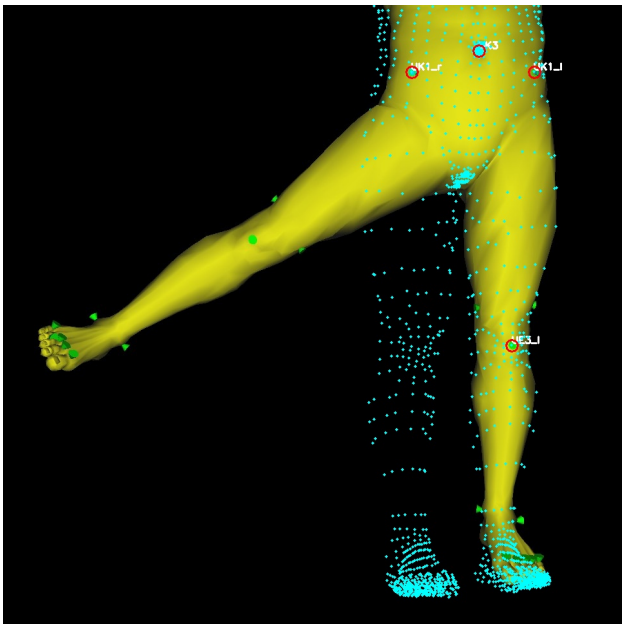
Figure 6.9: The Figure shows us the distance between the selected markers and the projected markers depending on the noise level.

6.3.2 Posture

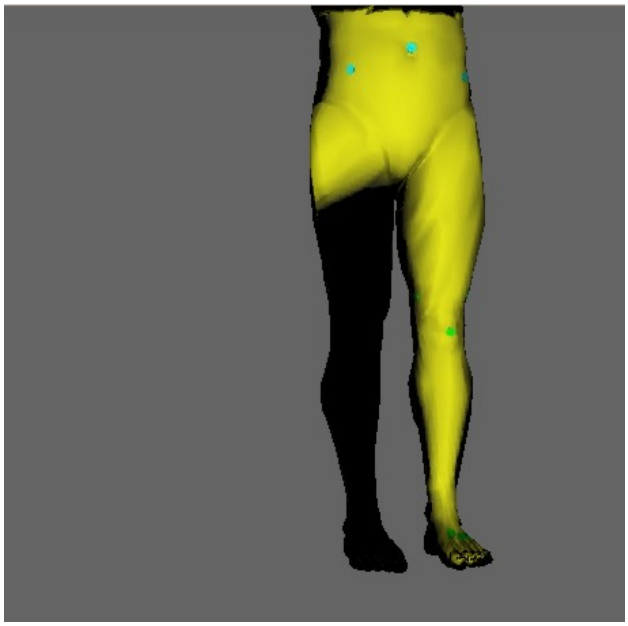
Posture has a large impact on the result. The aim of the experiments is to show the influence of the posture.

In the current state the functionality to adapt the posture automatically is not reliable enough therefore we adapted the posture manually.

We can try to use only markers which are not affected by posture. The POSIT algorithm uses the available marker to estimate the transformation. The texture coordinates are calculated from the reference model without adaption of the posture. As a result we would not see the influenced parts on the reference model (see Figure 6.10). By simply ignoring posture we get the solution in Figure 6.11. A transformation, which is calculated with the POSIT algorithm between markers of the image with posture and markers of the model without posture, is unusable. In Figure 6.12 we see the result with adapted posture.

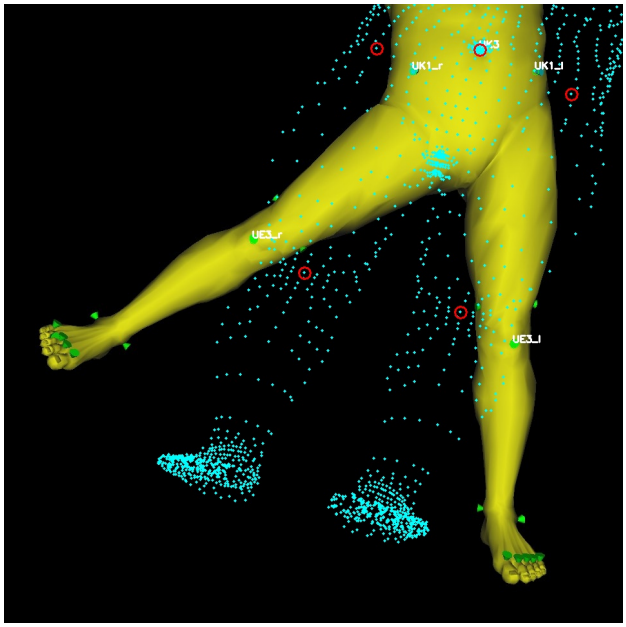


(a) Image with Texture Coordinates

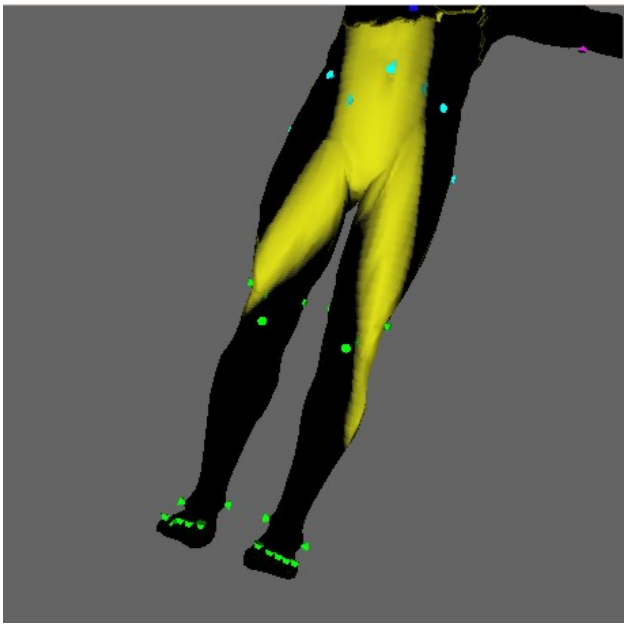


(b) Reference Model with Texture

Figure 6.10: In image a) there are no markers defined on the right leg therefore in the result image b) the right leg is not visible.

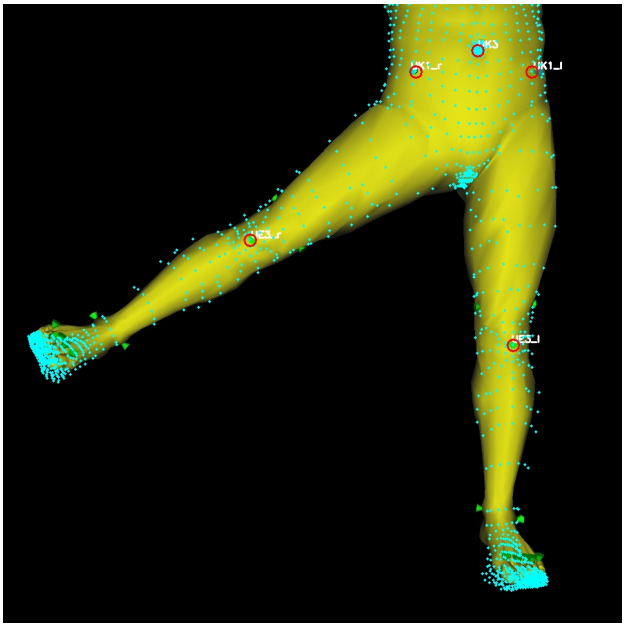


(a) Image with Texture Coordinates

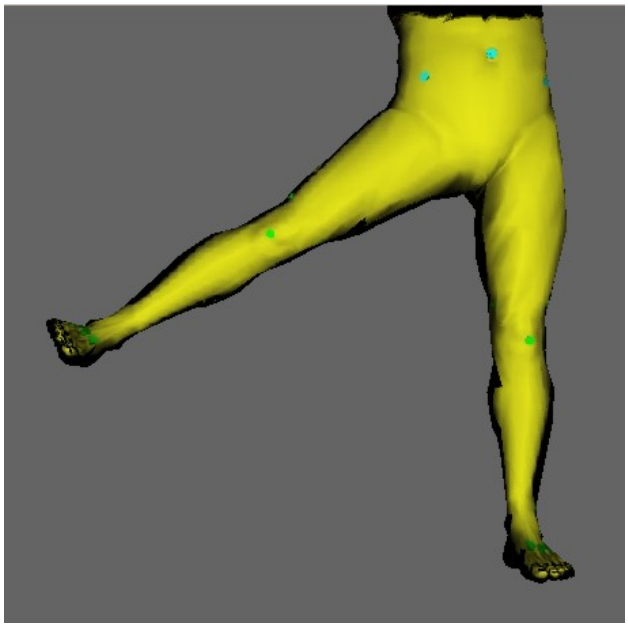


(b) Reference Model with Texture

Figure 6.11: We define markers on both legs but without adapting the posture. The POSIT Algorithm gives us a solution that is not useful.



(a) Image with Texture Coordinates

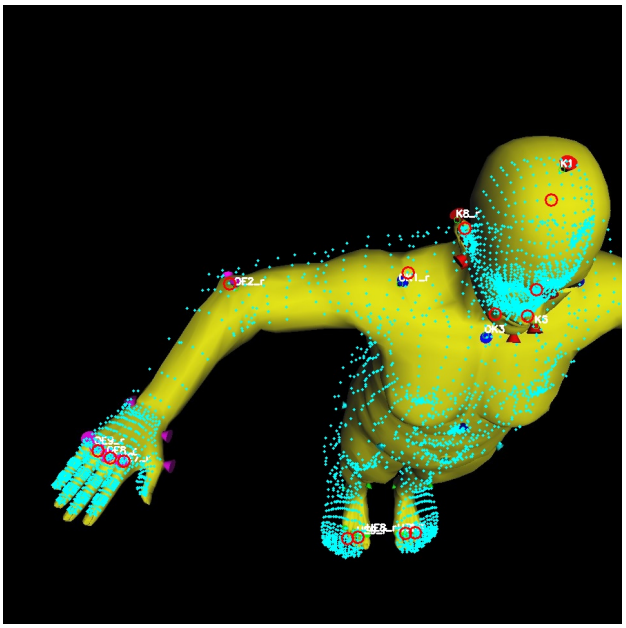


(b) Reference Model with Texture

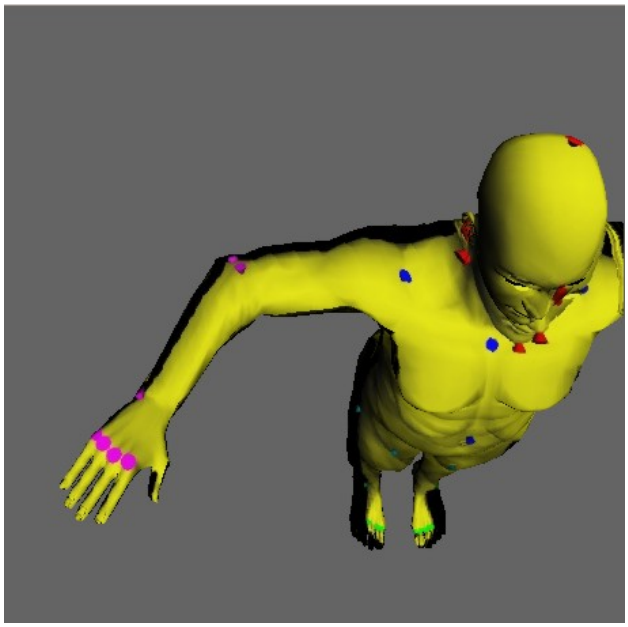
Figure 6.12: We have defined markers on both legs and adapted the posture therefore we can see the image as texture on the reference model.

6.3.3 Perspective

Perspective has a large influence on the result of image to mesh registration. Methods of image to mesh registration use SOP (Section 5.1) and therefore have limits concerning perspective distortion within an image. The effects of perspective can be seen in Figure 6.13. In order to get usable results it is necessary to avoid perspective distortion at image acquisition.



(a) Image with Texture Coordinates



(b) Reference Model with Texture

Figure 6.13: In image a) there is serious perspective distortion. The result image b) shows the problems of perspective. Mainly the feet are affected but also the rest of the body is imprecise.

Chapter 7

Conclusion

Contents

In this work we have shown a tool which incorporates the possibility to display volumetric data from MRI, CT and image data in the same reference system. First, we evaluated different reference models. Secondly we presented two registration processes; one for volume to mesh registration and one for image to mesh registration. Volume to mesh registration computes an initial transformation based on markers and refines the transformation with the ICP algorithm. The experiments have shown that due to the ICP algorithm the registration has a tolerance to measurement errors of the marker locations. The image to mesh registration process uses the POSIT algorithm; here measurement errors of the marker locations directly influence the result.

7.1 Goals and Drawbacks

A drawback of our method is that the appearance of the reference model has to be adjusted manually according to the data. We tried to recover the appearance automatically but the results were not feasible. The main reason for this was the fact that in general the data covers the human body only partially.

The posture of the reference model can be adapted automatically for volume to mesh registration but nevertheless some adjustments have to be done manually. For image to mesh registration the entire posture has to be adapted manually.

Our goal was to create a tool which allows visualizing evidence from image data and

evidence from MRI and CT data. Furthermore the tool was going to be used in correlation studies which compare internal and external evidence with respect to their location.

For visualization tasks the precision is sufficient and the tool can be used in the current version. As for the use in correlation studies the precision has to be increased.

7.2 Outlook

Future work will mainly concentrate on the following challenges:

- **Improvement of the 2D-3D registration.** In the current version we depend on the POSIT algorithm therefore we have to investigate alternatives such as in [29] to solve the pose estimation problem. Furthermore we have to evaluate the benefit of the use of calibrated cameras to obtain the images.
- **Usability.** The tool will be used by physicians and other non technical personal, therefore the user interface needs to be more simple and intuitive.
- **A better reference model.** The resolution of the current reference model is too low in order to document smaller injuries. Furthermore we have to expand the reference model in order to model children.

The work of this thesis will be continued in a cooperation between the Ludwig Boltzmann Institute of Clinical-Forensic Imaging (LBI-CFI) and the RISC Software GmbH[i] which are the developers of *BurnCase3D*[ii]. *BurnCase3D* is a documentation software specialized on burns (see Figure 7.1). Within *BurnCase3D* they use a set of static 3D-Models from infant to adult on which the user can mark the burns.

Within this cooperation the 3D-Models of *BurnCase3D* have to be integrated in a new tool with the possibility to alter the posture of the 3D-Models. Furthermore the functionality to mark burns on a model will be transferred to the new tool.

In the current state of development it is possible to alter the posture of a *BurnCase3D* model based on a skeleton (See Figure 7.2).

[i] `http://www.risc-software.at`
[ii] `http://www.burncase.at`

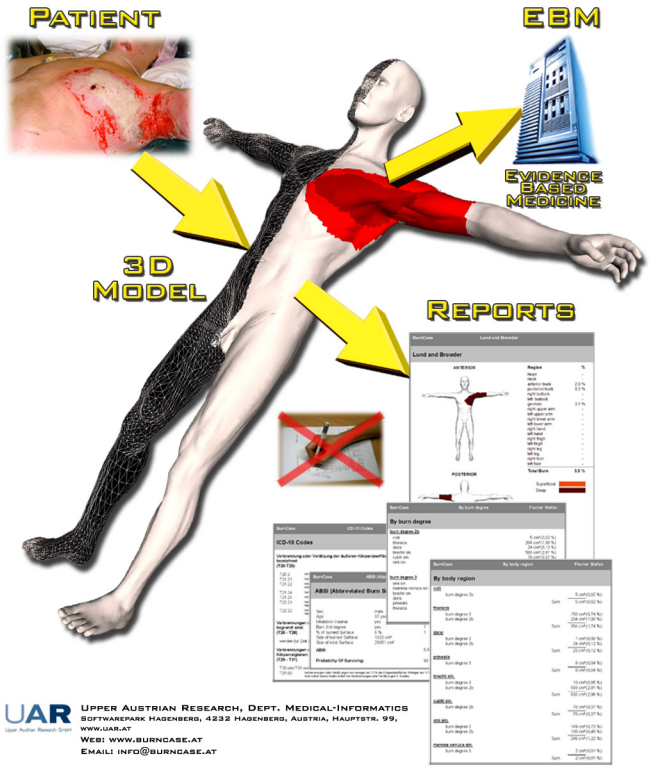


Figure 7.1: Features of *BurnCase3D*

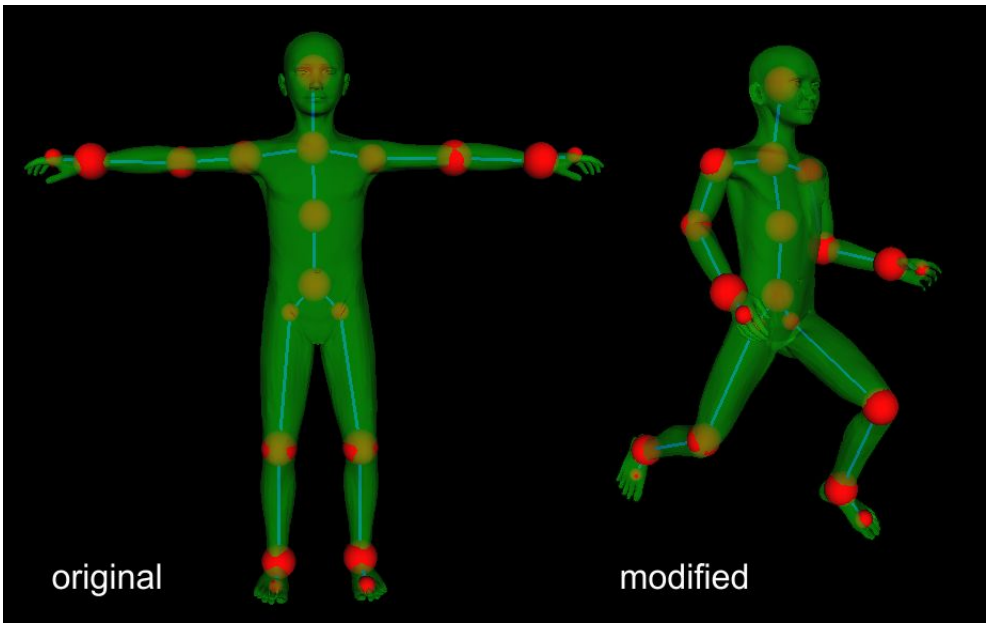


Figure 7.2: Modification of a static *BurnCase3D* model based on a skeleton.

Appendix A

Used libraries

A.1 Used libraries and Tools

Lib/Tool	**URL**	**Version**
ITK	http://www.itk.org/	3.16
Coin	http://www.coin3d.org	3.1.2
Quarter	http://www.coin3d.org/lib/quarter	1.0.0
OpenCV	http://opencv.willowgarage.com/	1.0
Glew	http://glew.sourceforge.net/	1.5.2
Qt	http://qt.nokia.com/	4.6.2
coindesigner	http://sourceforge.net/projects/coindesigner/	2.0
Tridecimator	http://vcg.sourceforge.net/index.php/Main_Page	1.0
CMake	http://www.cmake.org	2.6

Appendix B

Referencemodel

B.1 Reference marker position and name

Name	Beschreibung	Description
head (Kopf)		
K1	Kopfspitze	vertex
K2	zwischen den Augenbrauen	between the eyebrows
K3	Kinnspitze	chin point
K5	Schädel hinten	scull back
K6	Nasenspitze	tip of the nose
K8_r	Ohrlmuschelspitze rechts	auricles right
K9_r	Ohrlaeppchen rechts	earlobe right
K10_r	Ohr Tragus rechts	ear tragus right
K11_r	Kieferwinkel rechts	jaw angle right
K8_l	Ohrmuschelspitze links	tip of auricle left
K9_l	Ohrlaeppchen links	earlope left
K10_l	Ohr Tragus links	ear tragus left
K11_l	Kieferwinkel links	jaw angle left

Table B.1: List of the available markers of the head.

Name	Beschreibung	Description
upper part of the body (Oberkörper)		
OK1_r	Schulterdach rechts	socket (shoulder) right
OK1_l	Schulterdach links	socket (shoulder) left
OK2	7. Halswirbel Cervix	7. cervical vertebrae
OK3	Sternum - Brustbein	sternum - breastbone
OK4	Schwertfortsatz	xiphoid process
lower part of the body (Unterkörper)		
UK1_r	Beckenkamm rechts (vorne)	iliac crest right (front)
UK2_r	Becken hinten rechts	pelvis rear right
UK1_l	Beckenkamm links (vorne)	iliac crest left (front)
UK2_l	Becken hinten links	pelvis rear left
UK3	Nabel	navel
UK5_r	Oberschenkelkopf rechts	femoral head right
UK5_l	Oberschenkelkopf links	femoral head left
UK6	Steissbein	coccyx

Table B.2: List of the available markers of the upper and lower body parts.

Name	Beschreibung	Description
upper extremity (Obere Extremiät)		
OE1_r	Ellbogen rechts	elbow right
OE2_r	Oberarmaussenseite oben rechts	outside of upper arm top right
OE3_r	Oberarmaussenseite unten (narrisches Bein) rechts	outside of upper arm bottom (nervus ulnaris) right
OE4_r	Handgelenk innen rechts	wrist inside right
OE5_r	Handgelenk aussen rechts	wrist outside right
OE6_r	Daumenansatz rechts	root of the thumb right
OE7_r	Zeigefinger (zwei) rechts	forefinger (two) right
OE8_r	Mittelfinger (drei) rechts	middle finger (three) right
OE9_r	Ringfinger (vier) rechts	ring finger (four) right
OE10_r	Finger (fuenf) rechts	finger (five) right
OE1_l	Ellbogen links	elbow left
OE2_l	Oberarmaussenseite oben links	outside of upper arm top left
OE3_l	Oberarmaussenseite unten (narrisches Bein) links	outside of upper arm bottom (nervus ulnaris) left
OE4_l	Handgelenk innen links	wrist inside left
OE5_l	Handgelenk aussen links	wrist outside left
OE6_l	Daumenansatz links	root of the thumb left
OE7_l	Zeigefinger (zwei) links	forefinger (two) left
OE8_l	Mittelfinger (drei) links	middle finger (three) left
OE9_l	Ringfinger (vier) links	ring finger (four) left
OE10_l	Finger (fuenf) links	finger (five) left

Table B.3: List of the available markers of the upper extremities.

Name	Beschreibung	Description
lower extremity (Untere Extemität)		
UE1_r	Aussenseite Oberschenkel rechts	outside of thigh right
UE2_r	Innenseite Oberschenkel rechts	inside of thigh right
UE3_r	Kniescheibe Patella rechts	kneecap patella right
UE4_r	Knöchel innen rechts	ankle inside right
UE5_r	Knöchel aussen rechts	ankle outside right
UE6_r	Kniekehle rechts	hollow of the knee right
UE7_r	Ferse rechts	heel right
UE8_r	Zehengrundgelenk grosse rechts	metatarsophalangeal articulation (large) right
UE9_r	Zehengrundgelenk zweite rechts	metatarsophalangeal articulation second right
UE10_r	Zehengrundgelenk dritte rechts	metatarsophalangeal articulation third right
UE11_r	Zehengrundgelenk vierte rechts	metatarsophalangeal articulation fourth right
UE12_r	Zehengrundgelenk fuenfte (kleine) rechts	metatarsophalangeal articulation fifth (small) right
UE1_l	Aussenseite Oberschenkel links	outside of thigh left
UE2_l	Innenseite Oberschenkel links	inside of thigh left
UE3_l	Kniescheibe Patella links	kneecap patella left
UE4_l	Knöchel innen links	ankle inside left
UE5_l	Knöchel aussen links	ankle outside left
UE6_l	Kniekehle links	hollow of the knee left
UE7_l	Ferse links	heel left
UE8_l	Zehengrundgelenk grosse links	metatarsophalangeal articulation (large) left
UE9_l	Zehengrundgelenk zweite links	metatarsophalangeal articulation second left
UE10_l	Zehengrundgelenk dritte links	metatarsophalangeal articulation third left
UE11_l	Zehengrundgelenk vierte links	metatarsophalangeal articulation fourth left
UE12_l	Zehengrundgelenk fuenfte (kleine) links	metatarsophalangeal articulation fifth (small) left

Table B.4: List of the available markers of the lower extremities.

Appendix C

Animorph

C.1 Animorph File Types

C.1.1 base files

The base files are named *base.** and contain initial information about the mesh e.g. initial vertex and face data.

The base.vertices file contains the position of all vertices. Coordinates of a vertex are stored in one line divided by commas. This file type looks like this:

```
-1.062978,  -7.877367,  0.998311
-0.905655,  -7.875316,  1.043413
-0.922802,  -7.940919,  1.026910
     ⋮           ⋮          ⋮
```

The base.faces file provides information which points define a face.

```
6126,  6259,  6258,  6127
6118,  6127,  6258,  6119
6255,  6254,  6257,  6256
  ⋮      ⋮      ⋮      ⋮
```

Since the mesh is a quad mesh there are four entries in each line which denote the four points that build a face.

The mesh consists of 18618 points which form 19282 quads. We are only interested in the surface of the mesh but it contains a skeleton as well. We remove the skeleton from the

3D-Mesh which we use for further calculations. It is not possible to remove the skeleton at all because all modifiers are made for the 3D-Mesh with the skeleton. The 3D-Mesh of the body surface consists of 9275 vertices and 9505 quads. The other *base.** files contain data like colour for each vertex or texture coordinates for a texture provided by MakeHuman.

C.1.2 PoseRotation files

PoseRotation files define how the mesh is affected by rotation. Each PoseRotation consists of two files. A PoseRotatino file **.rot* looks like this:

```
6878,   0.291815
6879,   0.291815
6887,   0.053244
6888,   0.194306
7115,   0.291815
  ⋮         ⋮
```

The first number is an integer that specifies the affected vertex and the second number is a rotation angle in radians for the affected vertex.

The other file ends with **.rot.info*. This file contains important additional information about the rotation. Here is an example of this file type:

```
12374,12375,12376,12447,12448,12449
X
0,120.0
```

The first line contains the center vertex numbers. The centroid of the vertices specified by the center vertex numbers is the rotation center. In the second line we find the identifier of the affected axis. The third line indicates the minimal and maximal angle for this rotation. Further lines are ignored.

It is possible that more than one set of PoseRotation files for the same axis exist because it is allowed to have multiple PoseRotation as long as the limits of the minimal and maximal angle do not intersect. This is for example useful to specify a different behaviour for positive and negative rotation angles.

C.1.3 PoseTranslation files

PoseTranslation files define how the mesh is affected by translation. The filenames end with **.target* and look like this:

```
4882,   0.002938,   0.008136,    0.002436
4883,   0.001593,   0.007564,    0.002591
4884,   0.002706,   0.012340,    0.002079
4885,   0.003224,   0.010201,    0.000432
4886,   0.002024,   0.006040,   -0.000367
   ⋮          ⋮          ⋮           ⋮
```

The integer at the beginning of every line gives us the index of the affected vertex. The other three float numbers in the line form together the translation vector for the vertex.

If PoseTranslation files are used in combination with pose modifiers there is an additional **.target.info* file for each PoseTranslation. For example:

```
1.243614,1.6552133,1.7787966
-30.0,0
```

The first line consists of three float values which build a vector of the original size. The second line indicates minimal and maximal angle in the same way as for PoseRotation files.

C.1.4 BodySetting files

BodySetting files are used to save the applied set of either body and body detail modifiers or pose modifiers. An example of a BodySettings file for body and body detail modifiers can be found in the Appendix in Section C.2 and an example of a BodySettings file for pose modifiers in the Appendix in Section C.3.

BodySetting files end with **.bs* a line consists of the name of the modifiers and the according value.

C.2 BodySettings Example hero1.bs

This is a list of all used body and body detail modifiers to create the MakeHuman character hero1.

ages/male_30.target,0.770833
ages/male_50.target,0.229167
cheek/cheek001.target,0.35
chin_jaw/chin_jaw001.target,0.5
chin_jaw/chin_jaw020.target,0.21
chin_jaw/chin_quadrangular.target,0.72
eyes/eyes001.target,0.57
eyes/eyes004.target,1
eyes/eyes_big.target,0.18
eyes/eyes_close.target,0.06
forehead/forehead001.target,0.37
forehead/forehead005.target,0.
forehead/forehead_concave.target,0.11
mouth/mouth001.target,1
muscleSize/male_30_big_muscle.target, 0.360725
muscleSize/male_30_skinny_muscle.target, 0.408822
muscleSize/male_50_big_muscle.target, 0.107243
muscleSize/male_50_skinny_muscle.target, 0.121542
neck/neck_large.target,0.62
nose/nose005.target,1
nose/nose007.target,0.36
nose/nose027.target,0.53
nose/nose_short.target,0.29
nose/nosebase001.target,0.56
nose/nosebase002.target,0.55
nose/nosebase003.target,1
shapes/brevilinear_vshape.target, 0.037052
shapes/longilinear_peershape.target, 0.0513034
shapes/longilinear_vshape.target, 0.935255

C.3 BodySettings example dance1.bs

This is a list of all used pose modifiers to create the MakeHuman posture dance1.

020_right_foot/ROT1,60
026_left_footfinger_3_1/ROT1,26
028_left_footfinger_2_1/ROT1,40
030_left_footfinger_1_1/ROT1,54
040_left_foot/ROT1,60
044_right_forefinger_3/ROT1,23
045_right_forefinger_2/ROT1,17
046_right_forefinger_1/ROT1,28
047_right_middlefinger_3/ROT1,32
048_right_middlefinger_2/ROT1,39
049_right_middlefinger_1/ROT1,33
050_right_ringfinger_3/ROT1,35
051_right_ringfinger_2/ROT1,57
052_right_ringfinger_1/ROT1,44
053_right_littlefinger_3/ROT1,55
054_right_littlefinger_2/ROT1,41

055_right_littlefinger_1/ROT1,70
060_right_hand/ROT1,-36
060_right_hand/ROT3,50
066_left_forefinger_1/ROT1,-35
069_left_middlefinger_1/ROT1,-26
072_left_ringfinger_1/ROT1,-18
073_left_littlefinger_3/ROT1,8
074_left_littlefinger_2/ROT1,14
075_left_littlefinger_1/ROT1,6
080_left_hand/ROT1,11
080_left_hand/ROT3,26
100_right_lower_leg/ROT2,81
140_right_lower_arm/ROT1,18
180_right_upper_leg/ROT_BASE2,-30
200_left_upper_leg/ROT_BASE2,71
220_right_upper_arm/ROT_ADJUST0,129
220_right_upper_arm/ROT_ADJUST1,-5
220_right_upper_arm/ROT_BASE6,-45
240_left_upper_arm/ROT_BASE1,60
260_right_collar/ROT3,-10
320_neck/ROT2,-2
360_torso/ROT1,-25
360_torso/ROT2,-90
360_torso/ROT3,28
380_pivot/ROT2,73

Bibliography

[1] Balân, A. O., Sigal, L., Black, M. J., Davis, J. E., and Haussecker, H. W. (2007). Detailed human shape and pose from images. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1 –8.

[2] Bankman, I. N. (2008). *Handbook of Medical Image Processing and Analysis*. Academic Press, second edition.

[3] Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256.

[4] Brown, L. G. (1992). A survey of image registration techniques. *ACM Computing Surveys*, 24:325–376.

[5] DeMenthon, D. F. and Davis, L. S. (1993). Recognition and tracking of 3d objects by 1d search. In *DARPA Image Understanding Workshop*, pages 653–659.

[6] DeMenthon, D. F. and Davis, L. S. (1995). Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15:123–141.

[7] Dragomir, A., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., and Davis, J. (2005). Scape: shape completion and animation of people. *ACM Trans. Graph*, 24:408–416.

[8] Fitzgibbon, A. W. (2003). Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(13-14):1145–1153.

[9] Fitzpatrick, J. M. and Sonka, M. (2000). *Handbook of Medical Imaging, Volume 2. Medical Image Processing and Analysis (SPIE Press Book)*. SPIE–The International Society for Optical Engineering, 1 edition.

[10] Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 209–216, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[11] Goddard, J. S. J. (1997). *Pose and motion estimation from vision using dual quaternion-based extended kalman filtering*. PhD thesis, The University of Tennessee, Knoxville. AAI9840301.

[12] Guan, P., Weiss, A., Balân, A. O., and Black, M. J. (2009). Estimating human shape and pose from a single image. In *Proceedings of the IEEE 12th International Conference on Computer Vision*, Kyoto.

[13] Hajnal, J. V., Hill, D. L., and Hawkes, D. J. (2001). *Medical Image Registration (Biomedical Engineering)*. CRC Press.

[14] Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642.

[15] Horowitz, B. and Pentland, A. (1991). Recovery of non-rigid motion and structure. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 325 –330.

[16] Lee, H.-J. and Chen, Z. (1985). Determination of 3d human body postures from a single view. *Computer Vision, Graphics, and Image Processing*, 30(2):148–168.

[17] Li, H. and Hartley, R. I. (2007). The 3d-3d registration problem revisited. In *International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007, IEEE 11th*, pages 1–8.

[18] Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 163–169, New York, NY, USA. ACM.

[19] Modersitzki, J. (2004). *Numerical Methods for Image Registration (Numerical Mathematics and Scientific Computation)*. Oxford University Press, USA.

[20] Mori, G. and Malik, J. (2006). Recovering 3d human body configurations using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1052.

[21] Newman, T. S. and Yi, H. (2006). A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879.

[22] Onishi, K., Takiguchi, T., and Ariki, Y. (2008). 3d human posture estimation using the hog features from monocular image. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4.

[23] Rohr, K. (1994). Towards model-based recognition of human movements in image sequences. *CVGIP: Image Understanding*, 59:94–115.

[24] Rosenhahn, B. (2003). *Pose Estimation Revisited*. PhD thesis, Inst. f. Informatik u. Prakt. Math. der Christian-Albrechts-Universität zu Kiel.

[25] Schneider, D. C. and Eisert, P. (2009). Fast nonrigid mesh registration with a data-driven deformation prior. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 304–311.

[26] Shammaa, M. H., Suzuki, H., and Michikawa, T. (2007). Registration of cad mesh models with ct volumetric model of assembly of machine parts. In *Computer-Aided Design and Computer Graphics, 2007 10th IEEE International Conference on*, pages 21–21.

[27] Sonka, M., Hlavac, V., and Boyle, R. (2008). *Image Processing, Analysis and Machine Vision*. Thomson.

[28] Taylor, C. J. (2000). Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Computer Vision and Image Understanding*, 80:349–363.

[29] Toyama, F., Shoji, K., and Miyamichi, J. (1998). Model-based pose estimation using genetic algorithm. *Pattern Recognition, International Conference on*, 1:198–201.

[30] Wren, C. R., Azarbayejani, A., Darrell, T., and Pentland, A. P. (1997). Pfinder: real-time tracking of the human body. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):780–785.

[31] Zitová, B. and Flusser, J. (2003). Image registration methods: a survey. *Image and Vision Computing*, 21:977–1000.