FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING

Master's Thesis

Reconstruction of a patient-specific surface model of the humerus bone

Jan Quereš

PILSEN, CZECH REPUBLIC

2023

FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING

Master's Thesis

Reconstruction of a patient-specific surface model of the humerus bone

Bc. Jan Hereš

Thesis advisor
Eva C. Herbst, Ph.D., B.A.

PILSEN, CZECH REPUBLIC **2023**

Citation in the bibliography/reference list:
HEREŠ, Jan. *Reconstruction of a patient-specific surface model of the humerus bone*. Pilsen, Czech Republic, 2023. Master's Thesis. University of West Bohemia, Faculty of Applied Sciences, Department of Computer Science and Engineering. Thesis advisor Eva C. Herbst, Ph.D., B.A.

University of West Bohemia
Faculty of Applied Sciences
Academic year: 2023/2024

ASSIGNMENT OF DIPLOMA THESIS

(project, art work, art performance)

Name and surname:	**Bc. Jan HEREŠ**
Personal number:	**A22N0044P**
Study programme:	**N0613A140040**
Work topic:	**Reconstruction of a patient-specific surface model of the humerus bone**
Assigning department:	**Department of Computer Science and Engineering**

Theses guidelines

1. Describe the problem of reconstructing missing distal regions of a surface model based on an available generic model.
2. Analyze existing methods for surface model registration, with a focus on the methods relevant to this problem, which have an available implementation.
3. Propose a solution for reconstructing the missing parts of the model using the methods described above.
4. Implement and deploy the solution to the 3D Slicer application.
5. Perform testing of the proposed solution and discuss the results.

Length of diploma thesis:	**50 pages recommended**
Extent of graphics content:	**not specified**
Form processing of diploma thesis:	**printed/electronic**
Language of elaboration:	**English**

Recommended resources:

Will be provided by the supervisor

Supervisors of diploma thesis:	**Eva C. Herbst, PhD, B.A.** ETH Zürich
Consultant of diploma thesis:	**Doc. Ing. Josef Kohout, Ph.D.**

Date of assignment of diploma thesis:	**September 8, 2023**
Submission deadline of diploma thesis:	**May 16, 2024**

L.S.

Doc. Ing. Miloš Železný, Ph.D. dean	**Doc. Ing. Přemysl Brada, MSc., Ph.D.** Head of Department

V Plzni October 11, 2023

Declaration

In Pilsen, on 31 December 2023

. . . .
Jan Hereš

Abstract

This paper takes the reader into the field of reconstructing a missing distal region of a 3D humerus model, partially reconstructed from CT scans, using a mean model. As clinicians utilize modern medical imaging methods, some of them utilize X-rays, which are deemed to be harmful for the patient. Therefore, CT scans are usually limited to only the region necessary for the diagnosis. On the other hand, the clinicians need to gather as much imaging data as possible to enable exploring new possibilities, for instance biomechanical analysis, patient specific bone reconstruction surgery planning, and personalized implant design. The goal of this thesis is to propose a new approach to reconstructing the full 3D surface model of a patient-specific humerus bone based on a partial model rendered from a typical shoulder CT scan showing only the proximal humerus.

To achieve this goal, this paper analyses existing research on methods for reconstructing 3D surface models with focus on (humerus) bone models. Based on this analysis, the paper proposes a new pipeline utilizing a mean humerus model alongside the Bayesian Coherent Point Drift for the model reconstruction and the Random Sample Consensus combined with Iterative Closest Points algorithms for model pre-alignment.

The proposed pipeline is implemented in form of a module into the 3D Slicer application and used for demonstrating achieved results. The module is tailored specifically for the humerus. However, it is not strictly limited in terms of use cases; the algorithm parameters can easily be adjusted for other studies.

Abstrakt

Tato diplomová práce uvádí čtenáře do problematiky rekonstrukce chybějící distální části 3D modelu pažní kosti, který byl částečně zrekonstruován z CT snímků, za využití statistického vzorového modelu.

Lékaři běžně využívají moderní snímkovací technologie, z nichž část využívá principy rentgenového záření, které je ale považováno za škodlivé pro člověka. Z toho důvodu jsou běžně CT snímky omezeny pouze na oblast nezbytnou pro stanovení diagnózy. Na druhou stranu, lékaři potřebují shromáždit co nejvíce snímků pro umožnění zkoumání nových možností a metod v oblastech biomechanické analýzy, plánování chirurgické rekonstrukce pacientovy paže či tvorby implantátů pro daného pacienta.

Cílem této práce je navrhnout nový postup pro rekonstrukci kompletního 3D povrchového modelu pacientovy pažní kosti, jehož základem je její částečný model, který byl zpětně zrekonstruován z CT snímků ramene, na kterých se je vyobrazena pouze její distální část.

K dosažení tohoto cíle tato práce analyzuje stávající výzkum metod pro rekonstrukci 3D povrchových modelů, se zaměřením na modely (pažní) kosti. Na základě této analýzy je navržen nový postup rekonstrukce využívající model statistický model pažní kosti společně s metodu *Bayesian Coherent Point Drift* pro rekonstrukci modelu a metodu *Random Sample Consensus* v kombinaci s algoritmem *Iterative Closest Points* pro správné předzarovnání a natočení modelů v rámci dané scény.

Navržený postup je implementován formou modulu do aplikace 3D Slicer a slouží k demonstraci dosažených výsledků. Modul je přizpůsoben speciálně pro pažní kost, z hlediska využití však není striktně omezen, jelikož lze parametry využívaných metod lze snadno upravit.

Keywords

3D models • Slicer • Python • Mesh reconstruction • Bone reconstruction • Surface models

Acknowledgement

Thank you to Klevis Aliaj, Heath B. Henninger, Hema Sulkar, and Christopher Kolz for making the bone models from their biplane dataset openly available. Thank you to David Ackland and Yichen Huang for sharing their humeral models with us. I thank the Cancer Imaging Archive and creators of the QIN-HEADNECK dataset for sharing the imaging data for these humerus bones, and Tracy Nolan of the Cancer Imaging Archive for support regarding reuse of the QIN-HEADNECK dataset. QIN-HEADNECK was supported by Grants: U24 CA180918 (http://qiicr.org) and U01 CA140206.

Special thanks goes to Eva C. Herbst (ETH Zürich) for the excellent thesis supervision, Josef Kohout (University of West Bohemia) for providing consults on the official matters and Arthur Porto (Louisiana State University) for all of the valuable consults in the department of mesh model morphing.

Contents

Intro 1

In the fields of medical imaging and orthopedics, advancements in technology have revolutionized the way we approach diagnostics, treatment planning and surgical interventions. One significant area of progress is the creation of patient-specific skeletal models derived from their medical imaging data. These models serve as invaluable tools that bridge the gap between medical imaging and clinical practice, enabling healthcare professionals to gain enhanced insights into anatomical structures and customize treatments to the specific needs of individual patients.

Among various bones in the human body, the **humerus** bone holds a particular significance due to allowing for a wide variety of movement for our arms. While the humerus bone can be considered a relatively strong bone, it can be easily prone to substantial damage, mainly from serious traumatic events such as car accidents and high falls. If we combine this fact with the underlying data of a high frequency of recorded fractures [Mel14] and simpler physical geometry (relatively to other human bones), the humerus presents a perfect candidate for the patient-specific skeletal modelling research. Furthermore, routine shoulder CT scans typically only show the proximal third of the humerus, necessitating reconstruction of the distal region for the creation of patient-specific models. By accurately reconstructing surface models [1] of the humerus bone from medical imaging data, clinicians and researchers will be enabled to explore new possibilities including biomechanical analyses, patient specific bone reconstruction surgery planning, and personalized implant design.

Medical imaging provides the foundation for constructing personalized anatomical models. Technologies used for this task are often divided into several main categories:

1. Magnetic resonance imaging (MRI)
2. Sonography
3. and Radiography

[1]a 3D model representing a real object constructed as a mesh of geometrical primitives without a solid volume (more details in 1.2.2)

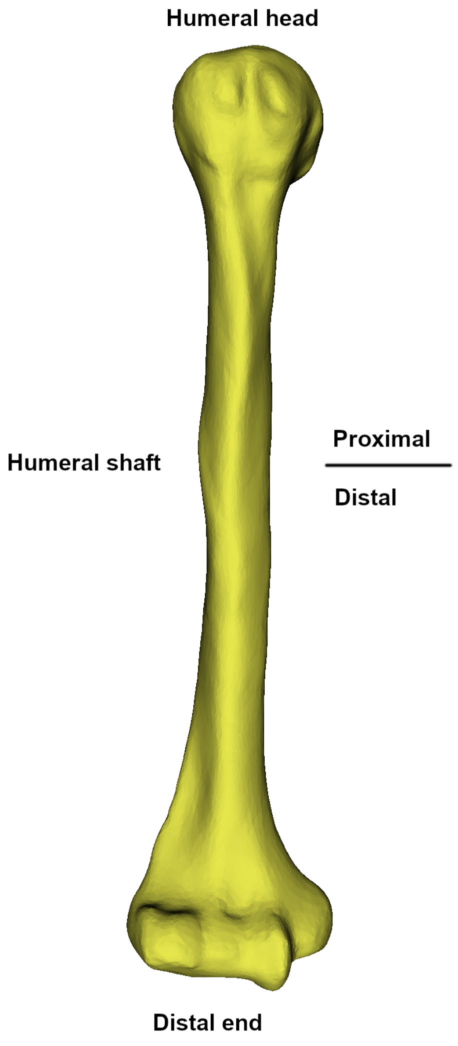


Figure 1.1: Example of a model of the humerus bone

In reality, there are many other methods for medical imaging, but for the purposes of this thesis, only the ones enumerated above are essential for understanding core concepts of this thesis.

1.1 Medical Imaging Methods

This section briefly describes each type of medical imaging method (as enumerated in section 1). An overview of all of commonly used imaging methods (in relation to the time of writing) is available at the end of this section in Table 1.1.

1.1.1 Magnetic Resonance Imaging

Magnetic resonance imaging (*abbrv. MRI*) is a technology utilizing two physical principles: magnetic fields and radio waves [BB]. It is a non-invasive imaging technology that produces 3D detailed images of the monitored regions of the human body. It is mostly used to visualize soft tissues, organs, and structures that might not be as clearly visible as with other imaging methods. The principle of an MRI scanner [BB] is quite ingenious in terms of the physics behind it as we will see in the following description.

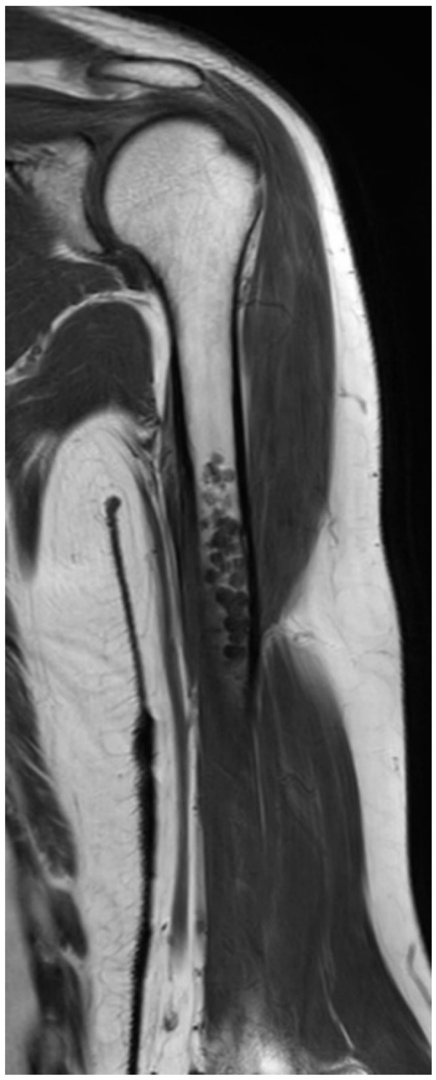
An MRI scanner (Figures 1.3 and 1.4) always contains a powerful magnet producing a strong magnetic field. This magnetic field is so strong that it physically forces the protons present in the human body to align with the direction of the field. Inside the tubular chamber of a scanner, a patient is surrounded by radio-frequency coils (*abbrv. RFC*) which produce radio-frequency waves. These waves are used to spin the protons out of an *equilibrium*[1], forcing them to misalign with the magnetic field. As soon as the radio-frequency coils stop producing their signal, the protons realign back with the direction of the magnetic field. However, this process is not instant and requires a significant amount of energy that is then released by each proton. Based on the type of tissue that the proton is present in, the amount of energy and time to fully realign varies substantially. By monitoring these properties, the differences are depicted using a greyscale with different levels of contrast in the produced image (Figure 1.2).

The MRI is particularly well suited for imaging soft tissues due to their great variance in chemical structure, thus resulting in great variance in the proton behaviour. One of the most significant advantages of the MRI is that it does not utilize any kind of ionizing radiation of X-rays compared to other methods covered in the following sections, therefore using this method does not introduce any health risks.

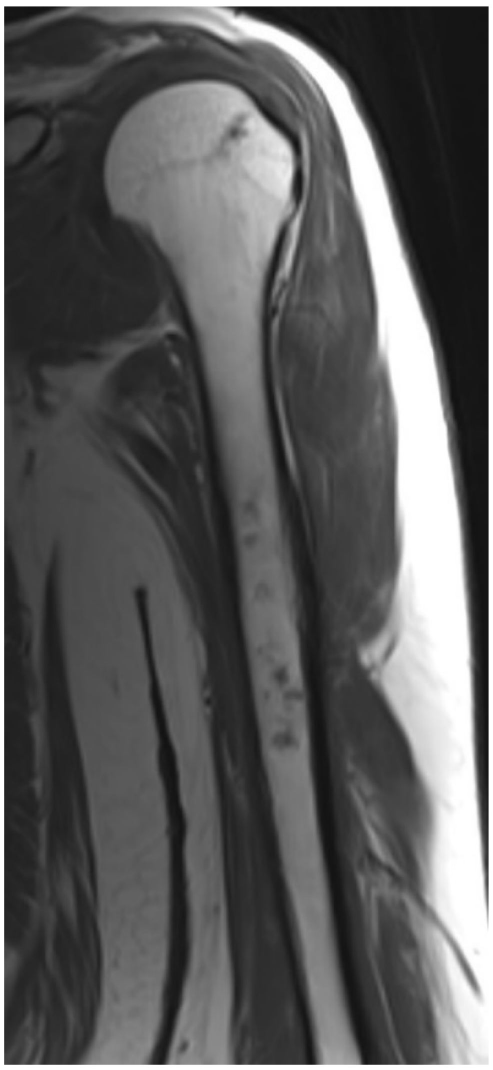
The downside of using an MRI scanner is its strong magnetic field which prohibits presence of any metal object inside or outside the patient, as well as in the near distance of the machine. This is usually a problem when any kind of metal implant such as pacemakers, defibrillators, insulin-pumps, or metal joint implants is present in a patient's body. The scanning process can be substantially longer compared to other commonly used imaging methods. Following, the scanning process itself is usually more expensive when compared to other imaging methods.

Overall, the MRI method is a great visualizing tool for the healthcare professionals, but is not universal. Nevertheless, it is often used as one of the first-line imaging procedure for pinning down the exact location of a problem.

[1]a state in which opposing forces or influences are balanced



(a)



(b)

Figure 1.2: An example of a humerus bone scanned using the MRI method. (a) depicts a humerus with a cartilaginous tumor, (b) depicts this condition in an earlier stage. Courtesy of [Dec+21] (shared under the CC-BY 4.0 DEED license)

Figure 1.3: Example of a real MRI scanner. Courtesy of MART PRODUCTION (freely available under the Pexels licence)

1.1.2 Sonography

Sonography describes a family of methods utilizing ***ultrasound*** waves in the imaging process. The research concerning sonography can be dated as far back as the 17th century to Lazaro Spallanzi [Bor23] and his research about bat navigation [D20]. The term ***ultrasound*** describes sound waves of higher frequency, typically in the range from 1 MHz to 20 MHz. For comparison, the human ear can capture frequencies generally between 20 Hz to 20kHz, which is an order of magnitude lower. The principle of using an ultrasound in the field of medical imaging is described in [Bor23], section *Basic Ultrasound Physics*, as following:

"Modern-day ultrasounds utilize piezoelectricity technology to convert electricity to sound waves and convert sound waves back into electricity through the use of lead zirconate titanate (PZT), which are human-made crystals (also called ceramic). The vibration occurs as these crystals are electrically stimulated, which produce longitudinal sound waves in the previously described frequency, thus producing ultrasound waves."

When such waves are produced, they are sent in the direction from the sonograph to the examined section of the patient. As they hit the patient, their energy is partially absorbed and the rest of the wave "bounces" off back to the sonograph, thus vibrating the PZT crystals, which convert it back to the electric signal. The amount of energy that is returned back depends on the type of the material and its density.

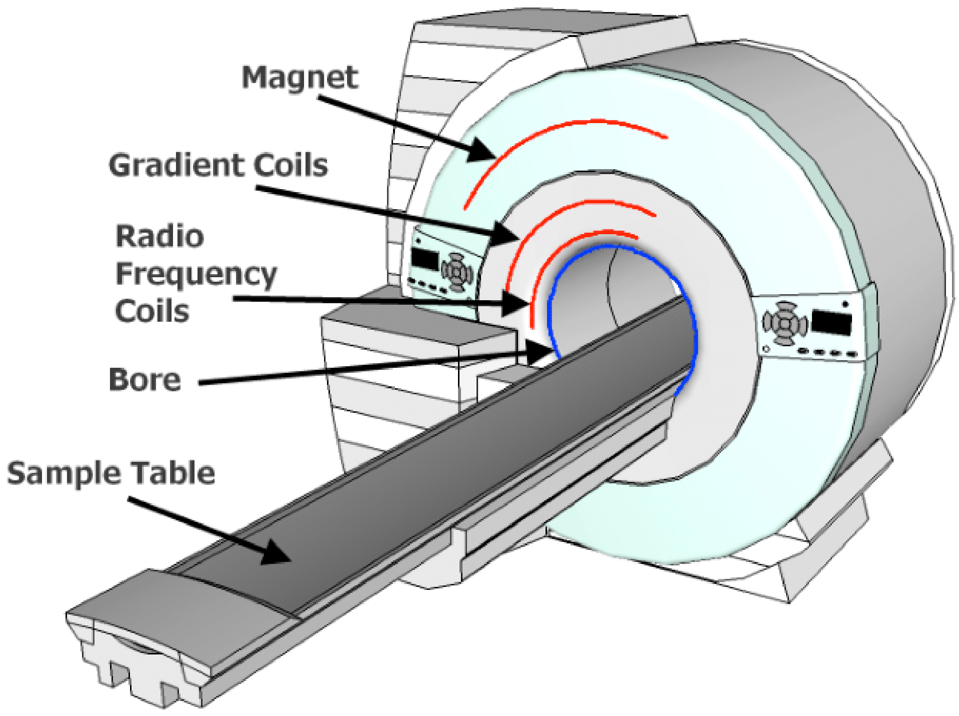


Figure 1.4: Cross-section of an MRI scanner. Courtesy of [HH13] (shared under BGS licence for non-commercial use)

Based on that knowledge, we are able to determine which type of tissue has been encountered as different tissues absorb different amounts of the waves' energy.

Sonography is a non-invasive type of imaging, which is one of its significant advantages. However, the issue with this method lies within the produced images as their overall resolution is heavily dependant on the skills of the sonographer. When setting up for this type of imaging, a certain amount of configuration is available, mainly concerning the characteristics of the sound waves. Those settings play an important role in determining the overall quality of the images. When the sonograph is not properly operated, a high number of undesirable image artifacts may be present in the results.

Overall, sonography is a great non-invasive alternative to other imaging methods. Nonetheless, based on the example of results shown in Figure 1.5, the produced images present a lot of noise compared to other mentioned methods (even when operated properly), which makes them not a strong candidate for our purposes.

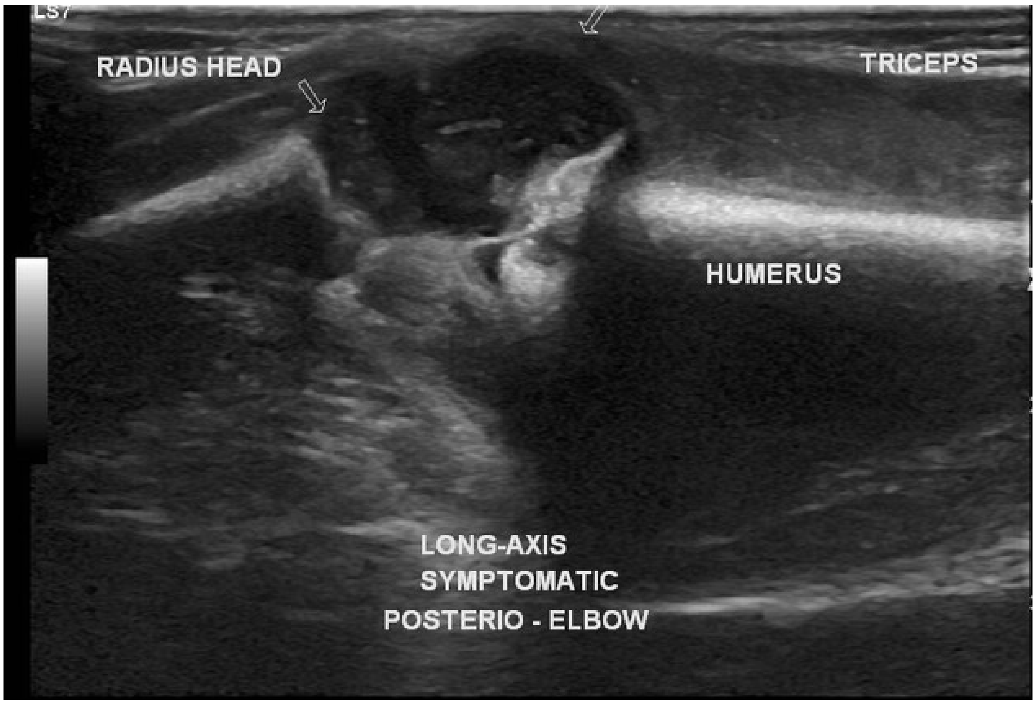


Figure 1.5: An example output of the sonography imaging method. Courtesy of [Pat23] (shared under the Radiopaedia licence)

1.1.3 Radiography

Radiography is a family of methods which use an ***X-Ray*** radiation (commonly referred to as **Röentgen radiation**) in the imaging process [FA]. An ***X-ray*** is a type of electromagnetic radiation which was discovered in 1895 by a German physicist Wilhelm Röentgen. The radiation is being utilized as a ray of high-energy photons, which resolves into a light beam with of a low wavelength, thus making it invisible for a human eye. When such beam passes through a soft human tissue, not much of it is absorbed since human tissues usually consist of very small atoms. On the other hand, structures like bones consist of much larger atoms, therefore they are capable of absorbing way more of the beam's energy.

The X-Ray machine (Figure 1.6) contains, amongst other parts, a pair of electrodes (cathode and tungsten anode) which are a crucial part of machine. The cathode is heated to release the electrons, leading to their attraction to the anode. Due to a high difference between voltages on the electrode pair, the electrons travel with extreme force. After the electron collides with the anode, it releases a highly energized particle in a form of a X-ray photon. Notably, when a photon is released from the electron, it disperses some amount of energy in a form of heat. To mitigate this effect, the anode is attached to a rotor which spins the anode in order to prevent

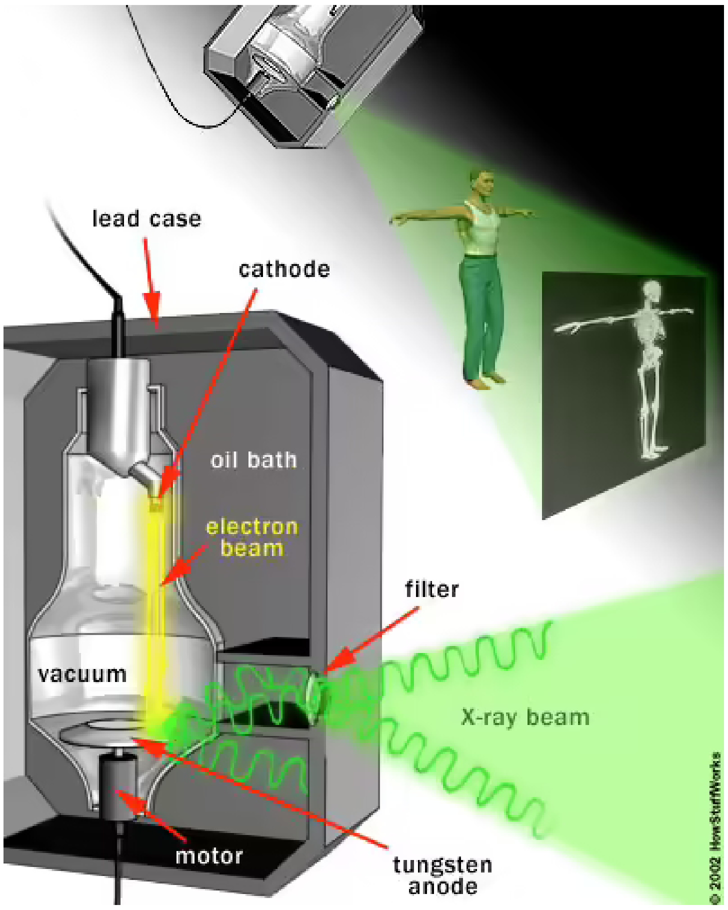


Figure 1.6: An example of the insides of an X-ray machine. Courtesy of [Har24] (shared freely for educational purposes)

overheating (and possibly meltdown) of the material. Naturally, these photons could be emitted throughout the whole surface of the machine but its lead shielding mitigates this effect, therefore the photons are forced to leave the space only through the designated area. Once the beam passes through a patient, a camera is present on the opposite side catching the beam of light, similarly to an ordinary camera. Based on the amount of energy caught by the camera, an inverted brightness scale (i.e. the parts which absorbed the most amount of the X-ray are shown as the brightest) is used to produce an image.

As noted earlier, soft tissues in a body will be shown only slightly, if at all. General muscle and fat areas are still visible, however it is quite difficult to distinguish between them. On the other hand, in modern medicine it is often desirable to include them all in the image (one of the reasons might be to reduce the needed amount of different imaging to help diagnose a patient). For this, clinicians usually need to use a special contrast media infused inside the body of a patient which help with absorbing as much of the X-ray particles as possible.

To address the most important issue: Are X-rays a health risk? In a short answer yes, though a quick explanation will present an important context to the reader. As mentioned, X-rays are a form of radiation. By definition, this presents a handful of possible risks, where the most significant is the likely increase of cancer occurrence due to the radiation overexposure to the human body. When cells are exposed to this type of radiation, several things happen [ZH]. First of all, the molecular structure of each exposed cell has increased risk of structural damage. This damage then could possibly lead to unpredictable cell mutations or possibly cell deaths (also called *necrosis*), which, potentially, could lead to conditions such as hair loss, skin burns, infertility and more. The amount of damage usually correlates to the amount of exposed radiation a human body has suffered in practice, but that is yet to be fully proven due to a numerous amount of unknowns in play. Based to the possible side effects the X-ray radiation might have on the human body, it is only desirable to limit the amount of radiation a human body is exposed to as much as possible.

1.1.4 Computed Tomography

Computed tomography (*abbrv.* CT), is an imaging technique closely related to the X-ray imaging. It was invented in 1972 by Sir Godfrey Hounsfield [Her23] and has been an important tool in clinical diagnosis to this day.

CT scanning process utilizes a rotating X-ray tube coupled with X-ray detectors placed in various angles (Figure 1.7). During the scanning procedure, the tube "shoots" out a narrow X-ray beam through the patient. On the opposite side of tube, the amount of energy of the beam reduced by the amount that has been absorbed by the scanned spot, is detected by the detectors. The measured amount is then sent to a computer which attached to the scanner. After the tube performs a full rotation, all of the gathered measurements are then processed by this computer using a sophisticated reconstruction algorithm, thus creating a single image (a "slice"). All of these steps are repeated until the desired amount of images is gathered.

This principle has the advantage that when using a CT, we are able to gather more information in form higher resolution images of the scanned area from different angles. Also, thanks to having scans of the examined area from different angles we're able to reconstruct it in a 3D virtual space. However, this process does not mitigate the effects of using an X-ray but quite the opposite, actually. The amount of radiation that the patient is exposed to is significantly higher and can get up to more than 10x [Pub21] the average dosage of a simple X-ray image. Due to this unfortunate property, the CT imaging is limited to a specific region for the examination, which restricts the amount of data to be gathered. An example how a CT scan result might look like can be found in Figure 1.8.

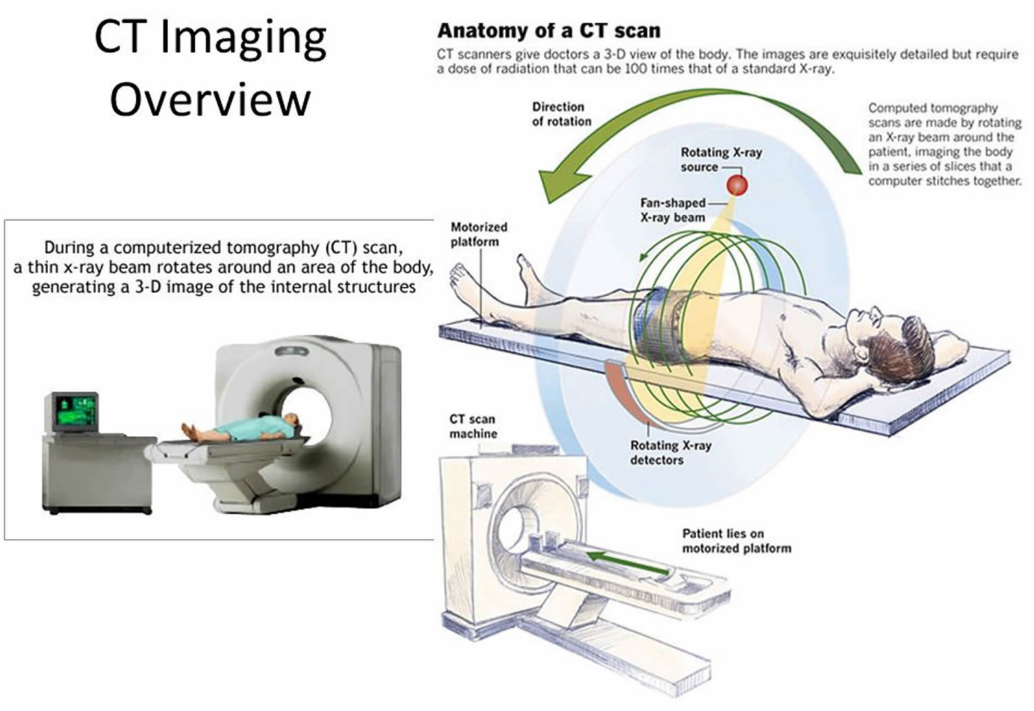

CT Imaging Overview
During a computerized tomography (CT) scan, a thin x-ray beam rotates around an area of the body, generating a 3-D image of the internal structures
Anatomy of a CT scan
CT scanners give doctors a 3-D view of the body. The images are exquisitely detailed but require a dose of radiation that can be 100 times that of a standard X-ray.
Direction of rotation
Computed tomography scans are made by rotating an X-ray beam around the patient, imaging the body in a series of slices that a computer stitches together.
Rotating X-ray source
Fan-shaped X-ray beam
Motorized platform
CT scan machine
Rotating X-ray detectors
Patient lies on motorized platform

Figure 1.7: Image of the insides of a CT scanner with short descriptions. Courtesy of HealthJade (shared freely for educational purposes)

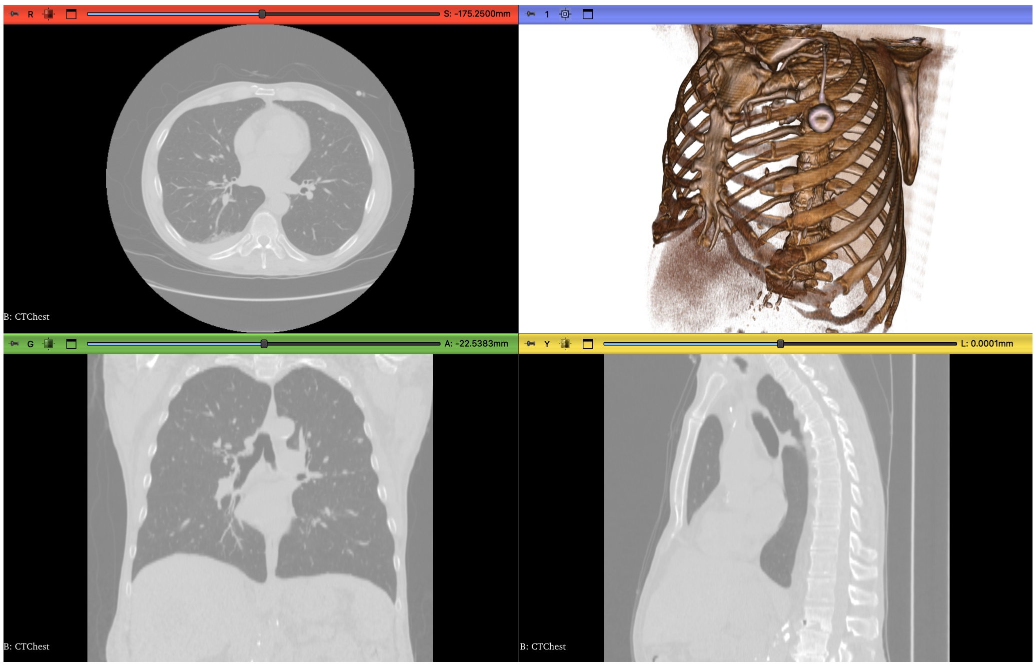


Figure 1.8: Example of how a CT scan and its 3D reconstructed result might look like

Table 1.1: Overview of commonly used medical imaging methods. Source: Healthcare Industry BW

Beginning of Table			
Method	**Principle**	**Description**	**Main areas of application**
X-ray image (radiography)	X-radiation (image display on film material, mostly digital)	Imaging of inner layers with or without contrast agent	Radiology, surgery, orthopaedics, dentistry
Fluoroscopy	X-radiation (dynamic X-ray images on monitors with image intensifiers)	Imaging of moving organs, usually with contrast agent	Gastrointestinal tract, lungs, heart (also for pacemaker positioning), venous vessels
Computed tomography (CT)	X-radiation (3D display with the aid of a gantry(rotating scanner))	3D image generation through computer calculation of the individual image signals	Radiology, oncology
Magnetic resonance imaging (MRI)	Electrical signals generated by magnetic fields of the hydrogen atoms in the tissue	Copmuter-generated thin-slice images of the soft tissues with or without contrast agent	Cardiology, oncology (CNS, internal organs, cardiovascular diseases)
Angiography (MR angiography, MR-A; CT angiography, CT-A)	Imaging of the blood and lymph vessels by MRI or CT	Mostly with contrast agents or fluorescent dyes	Stroke, heart attack, varicose veins, macular degeneration of the eye
Sonography (ultrasound)	Reflection of ultrasound waves on organs and tissues	Representation of all externally accessible fluid and soft structures	All areas. One of the most common imaging methods

Continuation of Table			
Method	**Principle**	**Description**	**Main areas of application**
Colour-coded Doppler sonography (FKDS)	Sonography that exploits the Doppler effect created by flowing liquids	Representation of the flow direction and velocity of arterial and venous blood	Angiology, cardiology
Scintigraphy	γ-radiation after incorporation of a short-lived radionuclide (“tracer”)	Detection with a gamma camera; colour-coded intensity display (scintigram)	Endocrinology, pneumology, oncology (especially screening for bone metastases)
Single photon emission computed tomography (SPECT)	Like scintigraphy with γ-ray tracet, but 3D detection via rotating collimators	2D or 3D scitigrams computer calculated from individual images with intensity coding	Cardiology, oncology (representation of metabolic intensity)
Positron emission tomography (PET)	Simultaneous detection of two γ-photons emitted during the β+-decay of the tracer	Imaging of metabolic functions in conjunction with CT or MRT	Oncology, neurology, cardiology
Endoscopy	Fibre optics in visible light with camera (also video)	Visualisation of hollow organs and body cavities	Gastroenterology, surgery, orthopaedics, pneumology
Electrical impedance tomography (EIT)	Measurement of electrical conductivity based on the free ions in the tissue	Representation of the condition (a-EIT) or function (f-EIT)	Pneumology, neurology, mammography, cardiology
Thermography	Measurement of infrared radiation with thermal imaging camera	Visualization of inflammatory lesions under the skin	Sports medicine

1.2 Patient Model Reconstruction

When gathering imaging data about a patient created by conventional methods (section 1.1), they are usually in form of a 2D slice or a set of such. Being able to create 3D models of the set of slices is important as they can provide more comprehensive overview of the anatomy (for example for clinical diagnosis of complex fractures), as well as facilitating 3D anatomical and biomechanical analysis. Methods for achieving this task ultimately share the same goal - create a 3D object out of the provided set of images, though their principles vary significantly.

1.2.1 CT Image Reconstruction

Figure 1.8 depicts two types of outputs - typical CT reconstructed images and their 3D model representation. Those are not the data gathered from the X-ray detectors themselves (based on the principle discussed in section 1.1.4). To create 3D models out of the data from the CT detector, we first need to reconstruct the image slices out of the raw sensor values for which various methods and algorithms were developed throughout time.

The first of these methods was discovered in a feasibility study performed by the Czech-Austrian mathematician Johann Karl August Radon. His research *On the Determination of Functions From Their Integral Values Along Certain Manifolds* [Rad86] was a breakthrough for modern CT image processing we use today. His findings, mainly the **Radon transform** method, forms a solid foundation for many modern image reconstruction methods today.

The idea of the Radon transform is the following. Let there be a source 2D image represented by a two-dimensional pixel matrix. Let there be a function $f(x) = f(x, y)$ that satisfies the three regularity conditions [Rad86]:

1. $f(x)$ is continuous,
2. the double integral

   $$\int\int \frac{|f(x)|}{\sqrt{x^2+y^2}} dxdy,$$

   over the whole plane converges,
3. for any arbitrary point [x, y] it holds that:

   $$\lim_{r->\infty} \int_0^{2\pi} f(x + rcos\phi, y + rsin\phi)d\phi = 0$$

4. Radon transform is then defined on the space of straight lines $L \subset R^2$ as following:

$$Rf(L) = \int_L f(x)|dx|$$

In other words, an image can be described by a line integral over each parallel line sent through the image. This fascinating observation strongly correlates with the idea of **forward projection**, which is a process of sending a set of parallel rays through the patient's body.

The concept of Radon transform is fully applicable for the forward projection. As the projection plane rotates (analogous to the rotating CT scanner), the output of the projection create a ***sinogram*** (as depicted in Figure 1.9, part A).

After the gathering of the data, the next steps is the **tomography reconstruction**. To this day, there are many of different reconstruction techniques, such as:

- Analytical reconstruction
- Iterative reconstruction
- Filtered back projection
- Deep learning reconstruction
- and others …

For the purposes of this thesis, the **filtered back projection** principle will suffice as an explanation of the idea behind tomography image reconstruction. The filtering part is introduced to sharpen the output image, because the output of the back projection is too blurry due to the initial blurriness of the sinogram (as depicted in Figure 1.9, part B). As the name suggests, the **back projection** principle is an **inverse** operation to the forward projection, which has just been covered.

At this point, we have reproduced a single slice of the scanned object. To reconstruct a 3D model of the object, the CT needs to produce higher number of scans throughout the whole part, which can be achieved by translating the scanner along one axis.

1.2.2 3D Model Rendering

When the 2D CT image slices are gathered, the final step of the reconstruction process is to render the 3D representation. To achieve that, the slices need to undergo several image processing methods to gather enough data about the object that needs to be rendered, since the scans usually contain other nearby parts of the patients body.

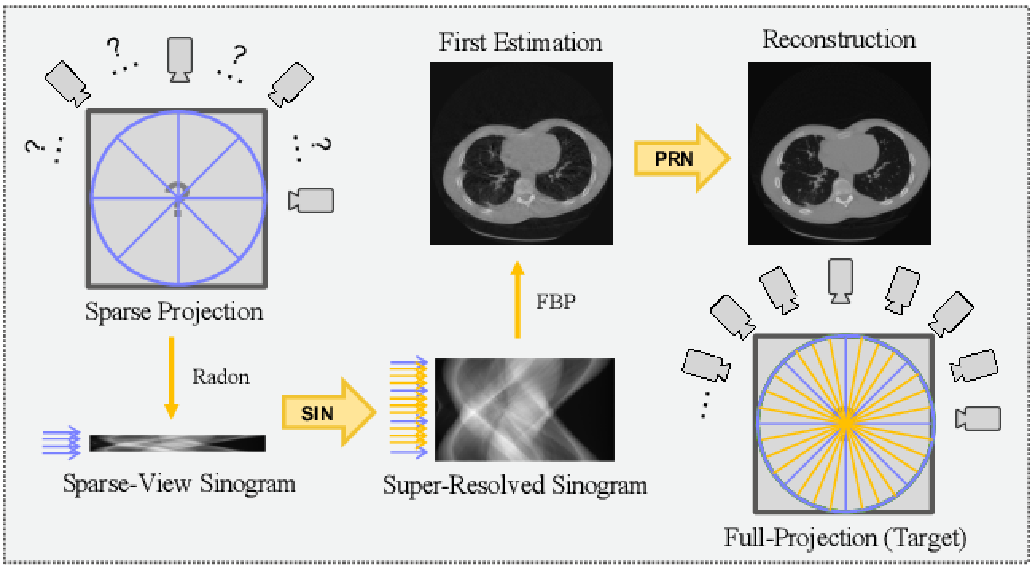


Figure 1.9: Pipeline of a CT image reconstruction. Courtesy of [Wei+20] (shared under the CC BY 4.0 DEED licence)

First, we need to perform an object **segmentation**, i.e. extracting parts of the source image that are relevant. This is usually done automatically by using an image segmentation algorithm, sometimes with manual input from a clinician or researcher to improve the accuracy as much as possible. Since the source scans are represented in a **grayscale** color space, we can also apply a **global threshold** to each pixel to filter out the noise. The thresholding can be also used for separating structures of different densities for further segmentation processes. The output of this step would be a set of images, each corresponding to its source scan, where only the relevant objects are showed.

Next, we need to convert each segmented image to a 3D surface model. This conversion can be achieved by a number of methods, one of which is the marching cubes algorithm [FLJ21], [LC87]. As this method is quite complicated, its details can be found in the cited resources. For the purposes of this thesis, only the output of this method is relevant, which is a ***triangular surface mesh model***. As mentioned in section 1, the term ***surface model*** can be defined as a "3D model representing a real object constructed as a mesh of geometrical primitives without a solid volume". Geometrical primitives that are used for constructing such meshes are usually some type of **polygons**. Every polygon consists of a set of **vertices, faces and edges**. Based on the type of the polygons, we can define the type of the mesh itself. For instance, a **triangle mesh** (Figures 1.10 and 1.11) would consist of polygons where each polygon would have faces represented as triangles.

At this stage of the pipeline, the last step remaining is to **visualize** the object.

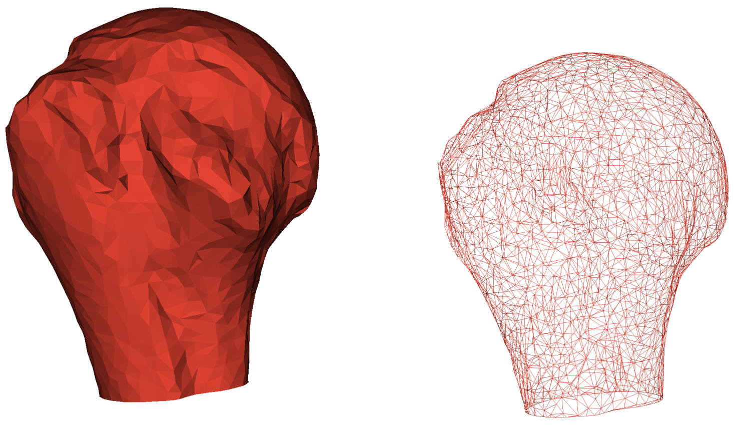


Figure 1.10: Example of a humeral head as a surface model (left) and its corresponding wireframe (right) consisting only of triangles

Again, many different visualization tools exist, however due to other supporting aspects, we will be using the **3D Slicer** (more details in chapter 3.1).

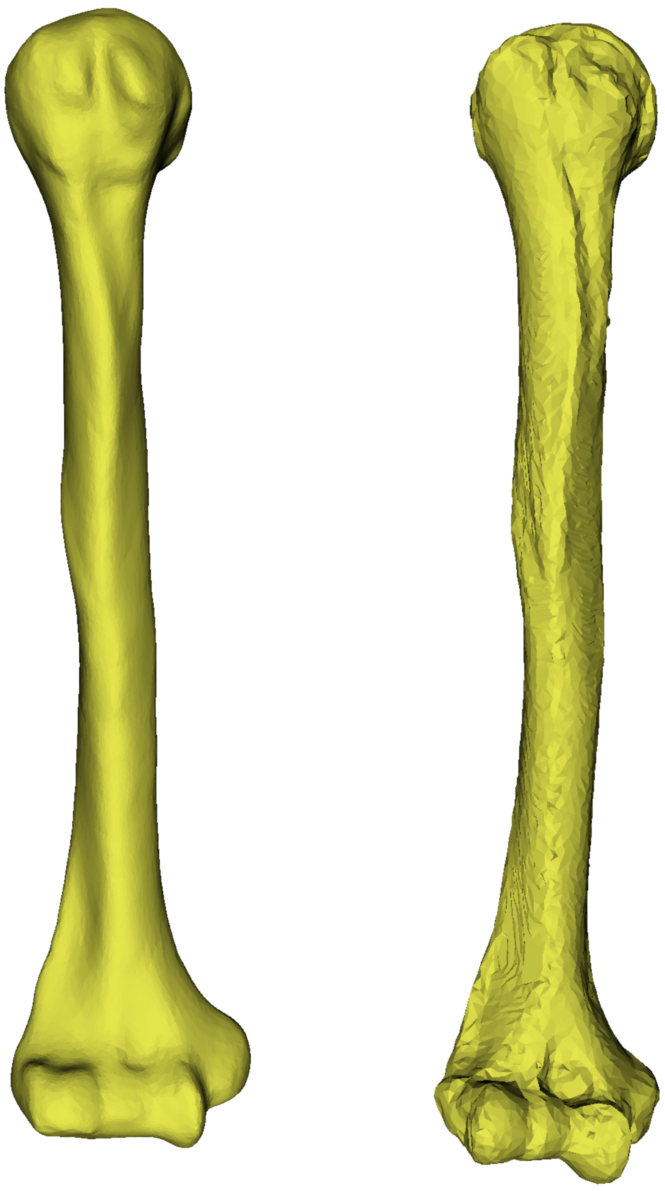


Figure 1.11: Visualization of different mesh resolution (density)

2 Humerus Surface Model Reconstruction Analysis

In this chapter, we will dive into possibilities of reconstructing missing **distal regions** (depicted in Figure 1.1) of the incomplete, patient-specific humerus surface model with the use of a statistical humerus surface model.

2.1 Statistical Humerus Surface Model

As noted earlier, to reduce the amount of radiation a patient is exposed to, shoulder scans are often **incomplete**; the resulting surface models of the humerus are often only showing the proximal one third. Without any further information, it would be impossible to know how to generate the missing regions. To solve this problem, we need some set of additional information, based on which we will be able to generate the missing distal regions of the surface models. One way to provide such information would be to use a second, **complete** model representing a humerus bone. It is very important to note here that this model must be **statistically representative** to properly capture the "average" anatomy of the humerus bone.

There are multiple ways to gather such a model. First of them is to acquire one that is publicly available. This is arguably the easiest solution, although low availability (if any) is a valid concern. What can be considered as even more significant drawback of this approach is the question whether random, publicly available, model is representative enough in terms of the anatomical features of the humerus.

The second way would be to gather enough CT scans of the humerus from a set of patients that would be willing to share their scans with a third party. As the CT machines are available only at hospitals or a clinician office, it is very difficult to gain access to it. Also, gathering enough CT images for a generic model could get quite expensive, so this approach is often skipped.

The third option would be to borrow data from existing studies and create the statistical model yourself, which is the approach that has been chosen for gathering the dataset for this thesis. The dataset and its contents are discussed in section 3.2.

2.1.1 Statistical Shape Modelling

For creating a statistical model we can use the **statistical shape modelling** approach (*abbrv. SSM*) [Amb+19]. As the name suggests, the goal is to create a statistical model by calculating a **mean template** throughout the provided set of **aligned** meshes. Hereafter, we will refer to this model as the ***mean model***.

Before we are able to create this model, we need to align all of the source meshes. In a 3D space, each model relative to another can be either translated, rotated or a combination of them. For the scale aspect, the scale of the models corresponds 1:1 to the source scans and real life size.

As shown in Figure 1.1, each humerus has its **proximal** and **distal** sections. These parts are essential due to their natural differences, by which we can easily determine the proper orientation in comparison to each other. For each mesh, we used **landmarking**, in other words placing marks on a selected subset of **recognizable anatomical features** of the humerus. Some of these points are labeled in Figure 2.1.

Landmarking. For the landmarking, we have three options: either do it manually, which is applicable only if the number of meshes is fairly small, or use one of the available automatic landmarking tools or combination of both. Each have their respective advantages and disadvantages - the manual approach can be more precise with the price of more manual labour, in comparison to using an automatic tool, which does ease up the amount of work significantly, although the precision of the landmarks can be non-negligible. For this analysis, we initially generated the landmarks using an automatic tool (details in section 3.3) and then manually checked these landmarks and adjusted any inaccuracies as needed - such inaccuracies were always slight, in the range of a few millimeters.

Mesh alignment process. With the landmarks created, we can proceed to the mesh alignment process. For this task, many methods exist which can be categorized into two main categories: ***non-rigid*** and ***rigid*** transformation methods. Methods in the first category **allow** for the source model deformation in terms of shape or size, whereas in the latter the **size and shape** remain **preserved**. Usually, the for the rigid type of alignment, rotation and translation transformation are applied to **each point of the meshes** uniformly. In the non-rigid category, more affine transformations such as scaling or shear mapping methods are utilized. In this stage, we do not want

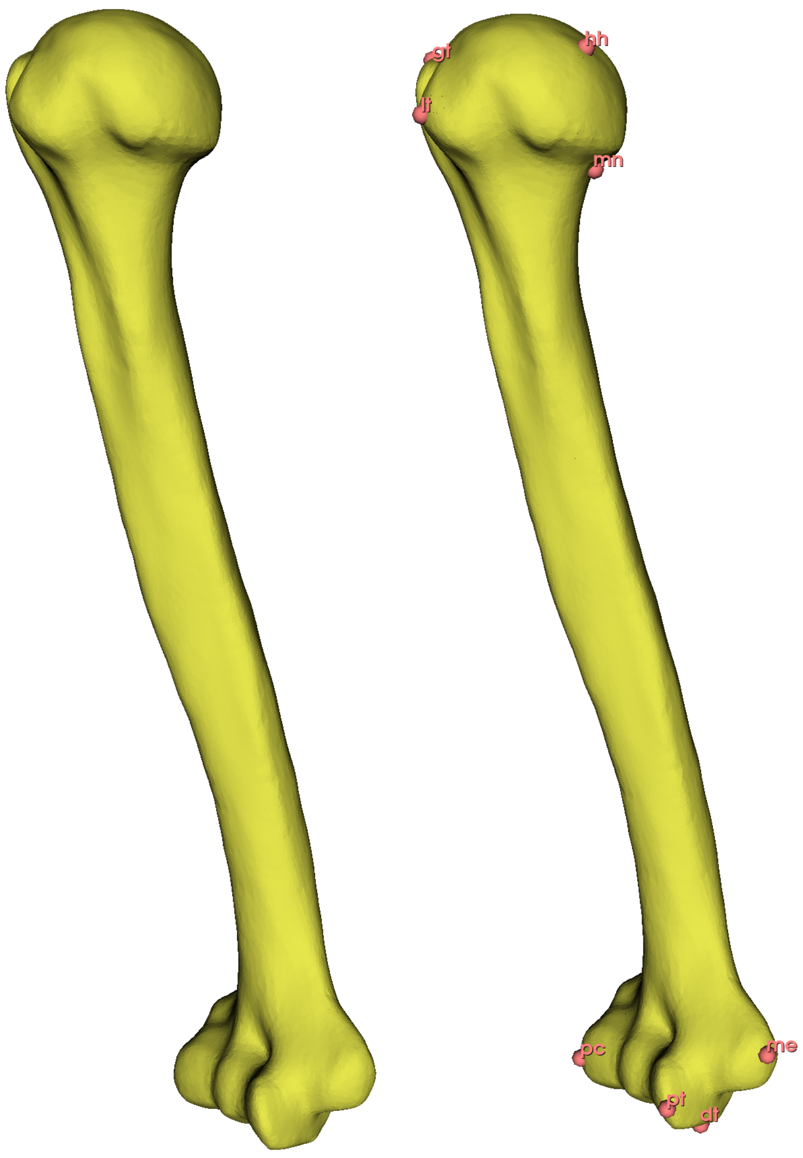


Figure 2.1: Example of a male model from the dataset A with (right) and without (left) some of the landmarks visible (right, pink points). Landmark legend: *gt* - greater tubercle, *lt* - lesser tubercle, *hh* - humeral head center, *mn* - medial anatomical neck, *pc* - posterior capitulum, *pt* - posterior trochlea, *dt* - distal trochlea, *me* - medial epicondyle

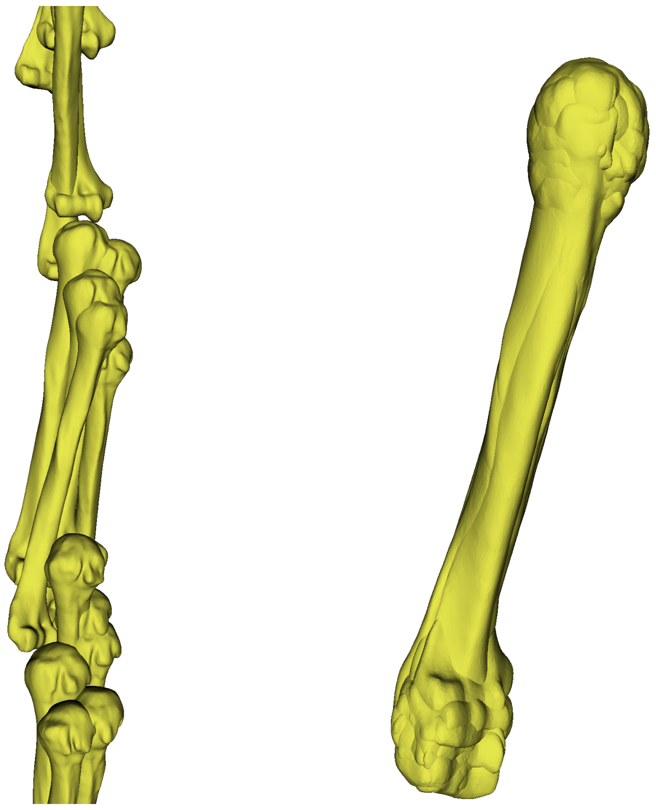


Figure 2.2: Set of male models from the dataset A before (left) and after (right) the mesh alignment process

to change the shape of the models at all, therefore we are limited here only to **rigid** alignment methods (more details in section 3.3.2). A possible output of this step is depicted in Figure 2.2.

Point correspondence analysis. After all of the meshes are aligned, the next step is usually to calculate the **point-point correspondence**. A point correspondence is generally a very difficult task to solve with various approaches (e.g. correlation-based or feature-based). However since we have aligned the models beforehand, it is possible to take a more simpler approach to this problem. The core idea is that thanks to the pre-alignment of the models, there's a very high chance that the corresponding points will be relatively close to each other. Details of the implementation are discussed section 3.3.2.

Mean mesh model generation. At this stage, all of the models are properly aligned and we have calculated all of the point correspondences. Let us continue to the mean mesh generation. The process of generating the mean model can be described as following:

1. Let us have a source mesh model A and a set of aligned models $B_1...B_n$ of size N, point $a \in A$ and its corresponding set of points $b_i \in B_i$, where $i \in< 1; N >$,
2. Calculate the mean position M of points $b_1...b_n$,
3. Move a to the newly calculated position M,
4. Repeat for each $a \in A$.

2.2 Humerus Surface Model Reconstruction

At this stage, we have two models available for use - the incomplete patient's humerus model and the mean humerus model. The question now is how can we utilize the mean model in order to reconstruct the missing distal region of the patient's humerus model?

To be able to evaluate different methods that are commonly used, let us first break it down the core of the problem. The humerus, as any other bone, is defined by its **real anatomical properties** [Pol+17]. These properties are different for the its proximal head, shaft and the distal end. They give a physical boundary, in which the generated model is deemed possible to exist. This means that the missing region needs to abide to its physical template in terms of the general shape, which means that the methods used for this case need to take a general shape into account for the reconstruction.

The mean model will be used as the **anatomical template**, based on which the missing region of the humerus will be reconstructed. The boundaries are virtually impossible to be fully quantified, since the humerus can vary significantly throughout the population [Pol+17]. However, by capturing the anatomical variation in a form of the statistical shape model of a population, we can get a better idea of the overall feasible shape space that a humerus is able to occupy. It is important to keep in mind that the mean model captures only the anatomical variation of the source dataset which is the reason why we want as representative dataset as possible (more details in section 3.2). For the purposes of the thesis, manual measurements and visual checks will be performed by qualified individual(s) in order to determine the quality of the reconstruction.

The main idea of utilizing the template is to find (point to point) **mapping** between the source (mean model) and the target (incomplete model) - a process called **point cloud registration** (recall section 2.1.1 where we were trying to find the point-point correspondence). If we are able to find this mapping, we would be able to determine the se of transformations to perform on the source proximal part in order to match the target proximal part as closely as possible. It is important to note here that this set of transformation cannot be applied to the rest of the source model because it would end up disproportionally deformed due to the lack of detail of the overall morphology of the target since it was determined just from the available proximal part. Arguably, the hardest challenge in our case is to find a mapping which maximizes the fit between the proximal parts of both of the models while minimizing the amount of necessary deformation of the source model.

As with the registration process (section 2.1.1), there are both **rigid** and **non-rigid** techniques for this task, just like the types of transformation. As a logical consequence, we want to introduce the least amount of non-rigid transformation to preserve as much of the anatomical resemblance as possible.

For this task, two main algorithmic approaches have been widely adopted through the time - the **iterative closest points** (ICP; recall sections 2.1.1) and **coherent point drift** (*abbrv. CPD*). In the following sections, we will go through both of the algorithms while explaining their core principles.

2.2.1 Iterative Closest Points

The ICP algorithm was proposed independently by Besl and McKay [Bes92] in 1992, and Zhang [Zha94] in 1994. Both propositions share the same principle, although Zhang's proposal includes a statistical method to deal with outliers, occlusion, appearance and disappearance due to which we will focus solely on the Zhang's implementation.

The core idea of the algorithm is very simple. Let us have two point clouds - source and target. The target point cloud will be fixed, whereas the source will be iteratively transformed in order to maximize the point match between them. The algorithm then works throughout the whole source cloud as following:

1. In each iteration:
    a) For each point in the source point cloud, match it with a point in the target point cloud to which the distance is the **lowest**
    b) For all of the formed point pairs, estimate a combination of rotation and translation operations by minimizing their root mean square point-point distance metric
    c) Apply the resulting transformation to all of the points of the source cloud
    d) Repeat from step 1

2. After the user-specified number of iterations is done or the user-specified convergence criteria are met (e.g. minimum distance threshold; dependant on the concrete implementation)

As hinted earlier, Zhang proposed a **k-d tree** algorithm in his paper for effective closest point computation based on the probability distribution of the distances, which mitigates the negative effect of outlier presence.

This algorithm is very simple in terms of the main idea and converges very quickly in the first few iterations, however several significant key observations are to be discussed. Firstly, the algorithm very easily gives only a **local optimum** solution, rather than a global one. Secondly, to achieve any kind of shape resemblance within the clouds, it is strongly recommended to **pre-align** the shapes using a **rigid** registration technique.

Overall, the ICP is a great algorithm for use when we have the sources already pre-aligned and are looking for a slight enhancement in the rough registration.

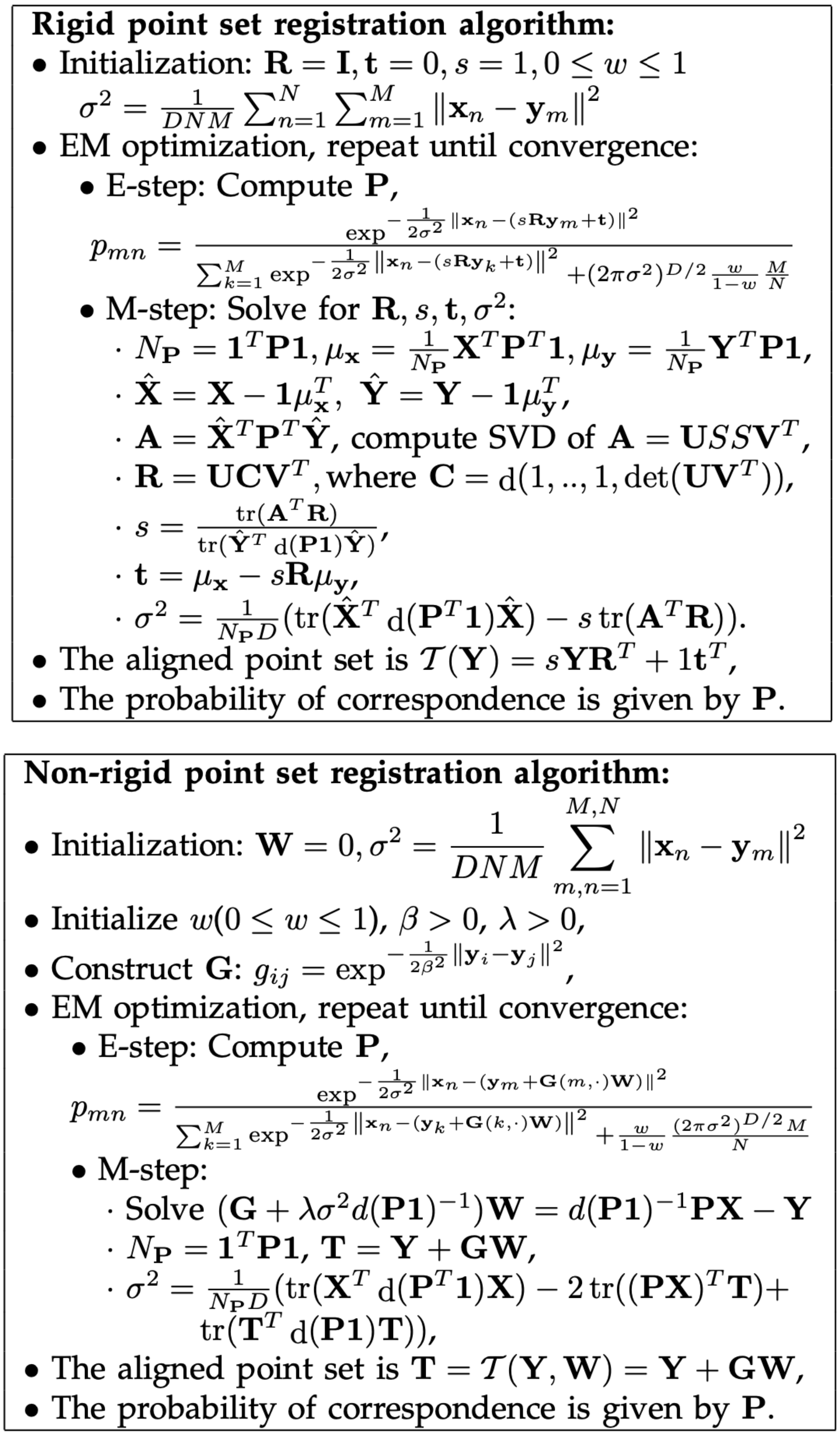
2.2.2 Coherent Point Drift

To mitigate the drawbacks of the ICP (2.2.1) algorithm, an alternative was proposed back in 2009 by Andryi Myronenko and Xubo Song called **coherent point drift** [Myr10]. This algorithm transforms the point cloud registration problem to a probability density estimation problem.

Again, we have two point clouds - source and a target. The source represents a **Gaussian mixture model centroids** (*abbrv. GMM*) which are coherently moved to the target point cloud. The GMM is a clustering method calculating the incidence of a point in a cluster in terms of probability. A GMM centroid then describes a set of points (=GMM cluster) for which the mean position can be calculated. The core of the CPD algorithm is to calculate a highest probability position of the centroids while applying different set of transformation. The important aspect of this algorithm is that as the centroids are defined by their mean point position, the set of calculated transformation is applied to each point of the cluster **equally**, thus moving the whole cluster somewhat coherently.

In the paper [Myr10], the authors present the algorithm in both rigid and non-rigid variants, which enables for wide variety of application. As the paper goes into great depth in terms of the algorithm, more details can be found in [Myr10]. The exact steps of both variants are depicted in Figure 2.3.

Seemingly, the CPD algorithm should, in theory, bear better results in terms of preserving the original shape, since it offers a rigid variant and takes clusters of points into an account, whereas the ICP focuses on singular points. As we will get to section 2.2.3, it is always not the case.



```
**Rigid point set registration algorithm:**
• Initialization: $\mathbf{R} = \mathbf{I}, \mathbf{t} = 0, s = 1, 0 \leq w \leq 1$
    $\sigma^2 = \frac{1}{DNM} \sum_{n=1}^{N} \sum_{m=1}^{M} \|\mathbf{x}_n - \mathbf{y}_m\|^2$
• EM optimization, repeat until convergence:
    • E-step: Compute $\mathbf{P}$,
        $p_{mn} = \frac{\exp^{-\frac{1}{2\sigma^2}\|\mathbf{x}_n - (s\mathbf{R}\mathbf{y}_m + \mathbf{t})\|^2}}{\sum_{k=1}^{M} \exp^{-\frac{1}{2\sigma^2}\|\mathbf{x}_n - (s\mathbf{R}\mathbf{y}_k + \mathbf{t})\|^2} + (2\pi\sigma^2)^{D/2} \frac{w}{1-w} \frac{M}{N}}$
    • M-step: Solve for $\mathbf{R}, s, \mathbf{t}, \sigma^2$:
        · $N_{\mathbf{P}} = \mathbf{1}^T\mathbf{P}\mathbf{1}, \mu_{\mathbf{x}} = \frac{1}{N_{\mathbf{P}}}\mathbf{X}^T\mathbf{P}^T\mathbf{1}, \mu_{\mathbf{y}} = \frac{1}{N_{\mathbf{P}}}\mathbf{Y}^T\mathbf{P}\mathbf{1}$,
        · $\hat{\mathbf{X}} = \mathbf{X} - \mathbf{1}\mu_{\mathbf{x}}^T, \ \hat{\mathbf{Y}} = \mathbf{Y} - \mathbf{1}\mu_{\mathbf{y}}^T$,
        · $\mathbf{A} = \hat{\mathbf{X}}^T\mathbf{P}^T\hat{\mathbf{Y}}$, compute SVD of $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{S}\mathbf{V}^T$,
        · $\mathbf{R} = \mathbf{U}\mathbf{C}\mathbf{V}^T$, where $\mathbf{C} = \mathrm{d}(1, .., 1, \det(\mathbf{U}\mathbf{V}^T))$,
        · $s = \frac{\mathrm{tr}(\mathbf{A}^T\mathbf{R})}{\mathrm{tr}(\hat{\mathbf{Y}}^T \mathrm{d}(\mathbf{P}\mathbf{1})\hat{\mathbf{Y}})}$,
        · $\mathbf{t} = \mu_{\mathbf{x}} - s\mathbf{R}\mu_{\mathbf{y}}$,
        · $\sigma^2 = \frac{1}{N_{\mathbf{P}}D}(\mathrm{tr}(\hat{\mathbf{X}}^T \mathrm{d}(\mathbf{P}^T\mathbf{1})\hat{\mathbf{X}}) - s\,\mathrm{tr}(\mathbf{A}^T\mathbf{R}))$.
• The aligned point set is $\mathcal{T}(\mathbf{Y}) = s\mathbf{Y}\mathbf{R}^T + 1\mathbf{t}^T$,
• The probability of correspondence is given by $\mathbf{P}$.
```

```
**Non-rigid point set registration algorithm:**
• Initialization: $\mathbf{W} = 0, \sigma^2 = \frac{1}{DNM} \sum_{m,n=1}^{M,N} \|\mathbf{x}_n - \mathbf{y}_m\|^2$
• Initialize $w (0 \leq w \leq 1), \beta > 0, \lambda > 0$,
• Construct $\mathbf{G}$: $g_{ij} = \exp^{-\frac{1}{2\beta^2}\|\mathbf{y}_i - \mathbf{y}_j\|^2}$,
• EM optimization, repeat until convergence:
    • E-step: Compute $\mathbf{P}$,
        $p_{mn} = \frac{\exp^{-\frac{1}{2\sigma^2}\|\mathbf{x}_n - (\mathbf{y}_m + \mathbf{G}(m,\cdot)\mathbf{W})\|^2}}{\sum_{k=1}^{M} \exp^{-\frac{1}{2\sigma^2}\|\mathbf{x}_n - (\mathbf{y}_k + \mathbf{G}(k,\cdot)\mathbf{W})\|^2} + \frac{w}{1-w} \frac{(2\pi\sigma^2)^{D/2} M}{N}}$
    • M-step:
        · Solve $(\mathbf{G} + \lambda\sigma^2 d(\mathbf{P}\mathbf{1})^{-1})\mathbf{W} = d(\mathbf{P}\mathbf{1})^{-1}\mathbf{P}\mathbf{X} - \mathbf{Y}$
        · $N_{\mathbf{P}} = \mathbf{1}^T\mathbf{P}\mathbf{1}, \mathbf{T} = \mathbf{Y} + \mathbf{G}\mathbf{W}$,
        · $\sigma^2 = \frac{1}{N_{\mathbf{P}}D}(\mathrm{tr}(\mathbf{X}^T \mathrm{d}(\mathbf{P}^T\mathbf{1})\mathbf{X}) - 2\,\mathrm{tr}((\mathbf{P}\mathbf{X})^T\mathbf{T}) +$
          $\mathrm{tr}(\mathbf{T}^T \mathrm{d}(\mathbf{P}\mathbf{1})\mathbf{T}))$,
• The aligned point set is $\mathbf{T} = \mathcal{T}(\mathbf{Y}, \mathbf{W}) = \mathbf{Y} + \mathbf{G}\mathbf{W}$,
• The probability of correspondence is given by $\mathbf{P}$.
```

Figure 2.3: The CPD algorithm in rigid and non-rigid variants. Courtesy of [Myr10] (shared freely under the IEEE license freely for thesis reuse; Copyright ©IEEE, 2010)

2.2.3 Bayesian Coherent Point Drift

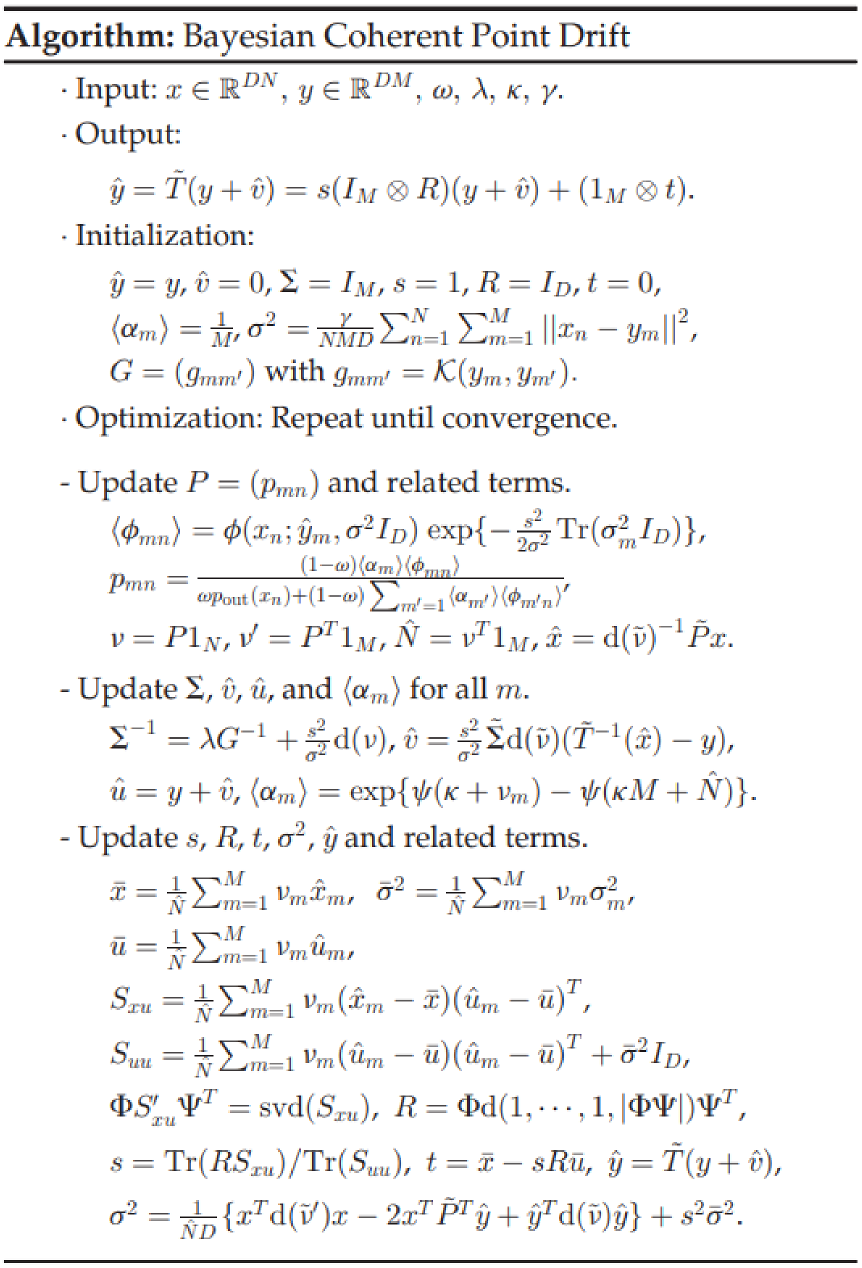
In 2020, Osamu Hirose (ORCID) from the Kanazawa University published a paper on the CPD algorithm called "A Bayesian Formulation of Coherent Point Drift" [Hir21]. In this paper, the author observes several main flaws to the CPD algorithm. Among theoretical flaws are the uncertainty of the convergence of the algorithm and the meaning of parameters controlling the centroid motion coherence. Among practical flaws are sensitivity to the target shape rotation in comparison to the source and that the CPD algorithm's acceleration is restricted to the use of the Gaussian kernel. To mitigate all of the identified negative aspects, the paper proposes a Bayesian formulation of the CPD algorithm named "**Bayesian Coherent Point Drift** (*abbrv. BCPD*).

The main idea of the reformulation to guarantee the convergence of the algorithm is to replace the motion coherence theory, based on which is the original CPD proposal, with the theory of variational Bayesian inference (*abbrv. VBI*). Firstly, a probabilistic model generating the target point cloud from the source point cloud is defined. The model is parametrized by **unobserved** random variables θ from a set of observations z, given a specific source and target point clouds. The set θ is then estimated either by computing the maximum mode of the posterior distribution $p(\theta|z)$ or the expectation of θ over $p(\theta|z)$. However, due to several drawbacks to this approach, such as high computational cost or unavailability of the maximum mode because of the multimodality of the posterior distribution, the VBI relaxes the difficulty by approximating the $p(\theta|z)$ using an alternative distribution $q(\theta)$, for which the maximum mode or the expectation easily computed. This distribution q is, however, unknown so the goal of finding an estimation of the θ is replaced by finding the distribution $q(\theta)$ which approximates $p(\theta|z)$ as closely as possible. In other words, the goal of VBI can then be formulated by the following equation:

$$\hat{q}(\theta) = argmin_q(-\int q(\theta)ln(\tfrac{p(\theta|z)}{q(\theta)})d\theta) \text{ [Hir21]}$$

The motion coherence used in [Myr10] is then reformulated as a prior distribution of displacement vectors. An important change from the CPD is that BCPD encapsulates both rigid and non-rigid transformation variants under a single algorithm. The rigidity is then defined by the value of the λ parameter, which represents the **inverse** of the expected length of the deformation vectors.

With all of the adjustments proposed in [Hir21], the final algorithm is depicted in Figure 2.4.



```
Algorithm: Bayesian Coherent Point Drift
  · Input: $x \in \mathbb{R}^{DN}$, $y \in \mathbb{R}^{DM}$, $\omega$, $\lambda$, $\kappa$, $\gamma$.
  · Output:
      $\hat{y} = \tilde{T}(y+\hat{v}) = s(I_M \otimes R)(y+\hat{v}) + (1_M \otimes t)$.
  · Initialization:
      $\hat{y} = y, \hat{v} = 0, \Sigma = I_M, s = 1, R = I_D, t = 0,$
      $\langle\alpha_m\rangle = \frac{1}{M}, \sigma^2 = \frac{\gamma}{NMD}\sum_{n=1}^{N}\sum_{m=1}^{M}||x_n - y_m||^2,$
      $G = (g_{mm'})$ with $g_{mm'} = \mathcal{K}(y_m, y_{m'})$.
  · Optimization: Repeat until convergence.

  - Update $P = (p_{mn})$ and related terms.
      $\langle\phi_{mn}\rangle = \phi(x_n; \hat{y}_m, \sigma^2 I_D)\exp\{-\frac{s^2}{2\sigma^2}\mathrm{Tr}(\sigma_m^2 I_D)\},$
      $p_{mn} = \frac{(1-\omega)\langle\alpha_m\rangle\langle\phi_{mn}\rangle}{\omega p_{\mathrm{out}}(x_n)+(1-\omega)\sum_{m'=1}\langle\alpha_{m'}\rangle\langle\phi_{m'n}\rangle},$
      $\nu = P1_N, \nu' = P^T 1_M, \hat{N} = \nu^T 1_M, \hat{x} = \mathrm{d}(\tilde{\nu})^{-1}\tilde{P}x.$
  - Update $\Sigma$, $\hat{v}$, $\hat{u}$, and $\langle\alpha_m\rangle$ for all $m$.
      $\Sigma^{-1} = \lambda G^{-1} + \frac{s^2}{\sigma^2}\mathrm{d}(\nu), \hat{v} = \frac{s^2}{\sigma^2}\tilde{\Sigma}\mathrm{d}(\tilde{\nu})(\tilde{T}^{-1}(\hat{x}) - y),$
      $\hat{u} = y + \hat{v}, \langle\alpha_m\rangle = \exp\{\psi(\kappa + \nu_m) - \psi(\kappa M + \hat{N})\}.$
  - Update $s, R, t, \sigma^2, \hat{y}$ and related terms.
      $\bar{x} = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m\hat{x}_m, \quad \bar{\sigma}^2 = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m\sigma_m^2,$
      $\bar{u} = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m\hat{u}_m,$
      $S_{xu} = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m(\hat{x}_m - \bar{x})(\hat{u}_m - \bar{u})^T,$
      $S_{uu} = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m(\hat{u}_m - \bar{u})(\hat{u}_m - \bar{u})^T + \bar{\sigma}^2 I_D,$
      $\Phi S'_{xu}\Psi^T = \mathrm{svd}(S_{xu}), \quad R = \Phi\mathrm{d}(1, \cdots, 1, |\Phi\Psi|)\Psi^T,$
      $s = \mathrm{Tr}(RS_{xu})/\mathrm{Tr}(S_{uu}), \quad t = \bar{x} - sR\bar{u}, \quad \hat{y} = \tilde{T}(y+\hat{v}),$
      $\sigma^2 = \frac{1}{\hat{N}D}\{x^T\mathrm{d}(\tilde{\nu}')x - 2x^T\tilde{P}^T\hat{y} + \hat{y}^T\mathrm{d}(\tilde{\nu})\hat{y}\} + s^2\bar{\sigma}^2.$
```

Figure 2.4: The BCPD algorithm. " The tilde symbol attached to a matrix denotes the Kronecker product between the matrix and I_D, e.g., $\sum^{\sim} = \sum \oplus I_D$ and $P^{\sim} = P \oplus I_D$, whereas the tilde symbol attached to a vector denotes the Kronecker product between the vector and I_D, e.g., $v^{\sim} = v \oplus I_D$ and $v^{\sim'} = v' \oplus I_D$. The symbol ψ represents the digamma function. The mth diagonal element of $\sum$ is denoted by σ_m^2. The singular value decomposition is denoted by "svd". The determinant and trace fo a square matrix are denoted by $|.|$ and $Tr(.)$, respectively. Unlike CPD, BCPD simultaneously estimates the variables of non-rigid and similarity transformations. ". Courtesy of [Hir21] (shared under the CC-BY 4.0 DEED license)

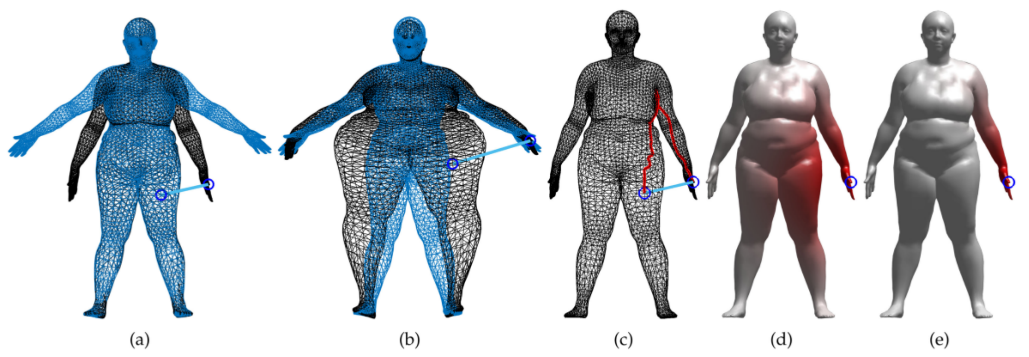


Figure 2.5: An example depicting advantages of the GBCPD algorithm compared to the BCPD. "(a) Human body shapes to be registered. The shapes colored black and blue are the source and target shapes, respectively. (b) Registration of the body shapes using BCPD with a Gaussian kernel. (c) Euclidean distance and geodesic distance between a pair of points, colored blue and red, respectively. (d) Visualization of a Gaussian kernel.". Courtesy of [Hir22] (shared under the CC-BY 4.0 DEED license)

2.2.4 Geodesic-Based Coherent Point Drift

Osamu Hirose also published a paper in May of 2023 proposing a new enhancement to his BCPD method called "**Geodesic-Based Bayesian Coherent Point Drift**" (*abbrv. GBCPD*) [Hir22]. It's an enhancement to the BCPD in terms of the possible deformations. The BCPD allows, in theory, for unnatural deformations of a shape, which in our case is not desirable. The solution is either to use the rigid "variant" of the BCPD or the proposed GBCPD.

The core of the algorithm remains the same. As CPD and BCPD consider Euclidean distance when performing the registration, GBCPD utilizes the **geodesic** distance, which is defined as the **shortest route between two points on a shape's surface**. The geodesic distance can be significantly larger between two points, depending on the shape of the object, therefore when registering the point clouds, the point mapping might be done with a higher precision, although the gains will be more significant in more complicated shapes. An important note here is that the GBCPD is solely a **non-rigid** registration algorithm, unlike the previous variant.

In Figure 2.5, the possible advantages in registration are shown.

2.2.5 Random Sample Consensus

As noted earlier, the ICP can easily converge to only **locally** optimal solutions and as with CPD, it suffers from the same disadvantage (however only for large planes; [Myr10]). An alternative tackling this problem from a **global** perspective is the **Random Sample Consensus** (*abbrv. RANSAC*) algorithm [FB81]. RANSAC is a commonly used algorithm for estimating parameters of a mathematical model containing **inliners** and **outliners**. The main idea of the algorithm is that outliers should have **no** influence on the value of the searched parameters whatsoever, thus a basic assumption of having enough inliners can be made. Let's define the following terms:

- Let's have an input set of data points U.
- Let's have a **parameter function** $f(S)$: $S \rightarrow p$, where $S \subset U$ and p is the calculated model parameters vector for subset S.
- Let's have a **cost function** $\rho(p, x)$ which calculates the cost for a single data point.

A pseudocode for this algorithm is shown in snippet 2.1.

Source code 2.1: Random sample consensus pseudocode

```
k = 0

c_thresh = 0.9 # Threshold for the cost
c_best = 0.0 # Best solution cost holder
p_best = [] # Best solution parameters

c_curr = 0.0
p_curr = []

i = 0
max_iter = 1000

while c_best < c_thresh and i < max_iter:
i = i + 1
k = k + 1
s_curr = random_select(U) # Select random subset of U
p_curr = f(s_curr) # Calculate the parameters for the subset S_k using function f

c_curr = sum(cost(p_curr, x))
if c_curr > c_best:
c_best = c_curr
p_best = p_curr
```

The output of this algorithm is a vector $p*$ containing the searched parameters of the model which had the maximized cost function.

Now, to be able to utilize this algorithm, we need to choose the **cost and parameter** functions. In the case of the cost, it is fairly simple - we will calculate the sums of distances between each pair of source and target points of the models. The parameter we are looking for are the distances per each pair of source and target points. The tricky part is then to determine the corresponding point pairs, for which we can utilize the **(fast) point feature histogram**.

The **point feature histogram** [Rus+08] is a histogram of values encoding the overall geometrical properties of the point's selected radius. The values are calculated as following.

Point feature histogram algorithm. Given a set of points $P = p_1, ..., p_n$, for each point p_i in P:

1. If normal n_i does not exist:
   a) Select all P_{k_α} neighbours of p_i within a given radius r_α
   b) Approximate n_i by the normal of the least-square plane to the P_{k_α} surface using Principal Component Analysis [Jol86]
   c) Use existing viewpoint information v to re-orient n_i consistently: $if \frac{<v-p_i;n_i>}{||v-p_i||} < 0$, then $n_i = -n_i$
2. For each point p_i in P, select all P_{k_α} neighbours of p_i within give radius $r_\beta > r_\alpha$
3. For each pair of point p_{j_1} and p_{j_2} $(j_1 < k_\beta, j_1 \neq j_2, j_2 < j_1)$ in P_{k_β} and their estimated normals n_{j_1} and n_{j_2}, select a source p_s and target p_t, the source being the one having the smaller angle etween the associated normal and the line connecting the points:
   a) If $< n_{j_1}, p_{j_2} - p_{j_1} > \leq < n_{j_2}, p_{j_1} - p_{j_2} >$
      Then $p_s = p_{j_1}, p_t = p_{j_2}, n_s = n_{j_1}, n_t = n_{j_2}$
      Else $p_s = p_{j_2}, p_t = p_{j_1}, n_s = n_{j_2}, n_t = n_{j_1}$
   b) Define the Darboux frame with the origin in the source point as:
      $u = n_s, v = (p_t - p_s)x\frac{u}{||p_t-p_s||}, w = u$
4. from p_s, p_t, n_s and n_t, comput a set of 4 features that measure the angle differences between the points' normals and the distance vector between them and bin the values into a histogram:
   a) $f_0 =< v, n_t >$
   b) $f_1 = ||p_t - p - s||$
   c) $f_2 = \frac{<u,p_t-p_s>}{f_i}$
   d) $f_3 = atan(< w, n_t >, < u, n_t >)$

Based on the definition in the description of the algorithm in 2.2.5, the algorithm presents a very high computational complexity (N^4 where N = point count) In 2009, Rusu R., et al. then presented a fast variant of the original calculation of the point feature histogram [RBB09] which we will be utilizing later.

2.3 Previous Work on Humerus Reconstruction

The goal to reconstruct a patient-specific bone has been attempted in the past by the following studies: [Hua+21], [Pol+17], [Vla+18].

[Hua+21]. In this paper, the authors attempted to reconstruct a glenohumeral joint by using statistical shape modelling (*abbrv. SSM*) paired with with landmarks close to the glenohumeral joint. The source dataset used is the same as our dataset A (more details in section 3.2) with applied smoothing and re-meshing to achieve surfaces with nominal element size of 1.0mm. The SSM was comprised of three main steps: **non-rigid registration, rigid body alignment** and **principal component analysis** (*abbrv. PCA*). The non-rigid registration step, based on the **radial basis foundation** algorithm (*abbrv. RBF*) [JAF18], was used to point-point correspondences throughout the dataset. The rigid body alignment step utilizing the **iterative closest point** algorithm was then applied to minimize the sum of squared distances between corresponding points of two shape surfaces. Finally, the PCA allowed any shape in the training set to be approximated as the mean shape plus a linear combination of the first k principal components [Abl+18]. This set of steps produced 4 statistical shape models (two for male/female scapulas, two for male/female humeri').

For the landmarking, 8 main humeral landmarks (4 in proximal area and 4 in distal area) adapted from cited studies and 15 clinically relevant scapular landmarks (each manually identified) were set for each source model. Landmarks of each source mesh were then registered to the landmarks of the corresponding mean model using **singular value decomposition** (*abbrv. SVD*) to ensure that the variations in the three-dimensional coordinates of each landmark were a consequence of geometric variations only.

For the reconstruction process a **shape prediction** approach was used. The prediction models were constructed by using an **univariate absolute shrinkage** concept paired with the **selection (Lasso) operator regression** [Tib96]. For each of the four SSMs, the most representative shape prediction model was chosen as the one formed by the set of Lasso regressors which had the lowest sum of Bayesian information criterion, and the optimal number of principal components which was selected as the set producing the smallest "leave-one-out" root mean square errors (*abbrv.* RMSE) between the predicted and actual surface mesh models.

A very important note by the authors states that they focused on predicting the distal morphology based on the proximal landmarks as the CT scans did not include the distal morphology of the humerus.

In the results section, the authors present the following results.

For the male humerus, shape prediction error was the largest (≥ 2.0mm) around the greater tubercule, lesser tubercule and the lateral epicondyle For the female humerus, shape prediction error was the largest (≥ 1.6mm) around the superior and inferior-posterior regions of the humeral head as well as the inferior edge of the distal humerus.

For humeral landmarks, the highest mean error was found at the "most medial point of the outline of the supraspinatus insertion" landmark→6.0 mm and 4.8 mm for male and female humeri, respectively. The lowest mean error was found at the "the incision point between the medial epicondyle and medial part of the trochlea" landmark→3.6 mm for the male humerus and the "the most lateral point of the lateral epicondyle" landmark→2.8 mm for the female humerus. The mean RMS error of the proximal humerus was 1.8 mm and 1.4 mm for males and females, respectively.

The lowest achieved mean surface RMS error were 2.7mm and 2.1mm for the male and female humerus models, respectively. For the male and the female humeri, shape prediction error was the largest (≥ 5.0mm and 3.5mm, respectively) around the superior region of proximal humerus and the inferior edge of the distal humerus

For humeral landmarks, the highest mean error was found at the "most inferior point of the medial trochlea" landmark→10.0 mm and 7.4 mm for male and female humeri, respectively. The lowest mean error for the male humerus was found at the "most lateral point of the outline of the subscapularis insertion" landmark→6.8mm. The lowest mean error for the female humerus was found at the "most medial point of the articular perimeter of the proximal humerus" landmark→5.2mm. The mean RMS error of the proximal humerus was 3.0 mm and 2.5 mm for males and females, respectively. For comparison, the usual CT slice thickness used in model development ranges from 3.0 to 5.0mm.

[Pol+17]. The aim of this paper was to reconstruct the pre-morbid anatomy 3D of the proximal humerus by utilizing the **statistical shape modelling** approach. The term "morbid" describes a humeral bone in an affected stage due to severe osteoarthritis or fracture(s). The authors created a statistical 3D model of an average humerus from a dataset consisting of 57 CT scans (38 women, 19 men with their mean age of 66 years (23 to 87)), all of which had not shown any unusual anatomical features.

The source models were landmarked automatically, although no exact description of the tools used was given. All landmarks sets were **normalized** in terms of **humeral size and shape** in order to accommodate for their anatomical variations. For each pair of the models the **B-splines image registration** algorithm was used [Xie04]. This process was repeated until matching between all bone surfaces in the dataset was complete and this match represented the statistical shape model (*abbrv.* SSM).

Using the created SSM, a reconstruction test was performed on a second dataset consisting of 52 CT scans of human shoulders including the whole humeri. The concrete method used for fitting the created SSM to a particular testing model is not explicitly disclosed, however it is presumed that the B-splines method was again utilized for this task. Each testing model was clipped using four different clipping planes, each mimicing different possible clinical situations.

The paper proposed the following results. With the metaphysis included, mimicking osteoarthritis, the errors of prediction for retroversion, inclination, height, radius of curvature and posterior and medial offset of the head of the humerus were 2.9° (± 2.3°), 4.0° (± 3.3°), 1.0 mm (± 0.8 mm), 0.8 mm (± 0.6 mm), 0.7 mm (± 0.5 mm) and 1.0 mm (± 0.7 mm), respectively. With the metaphysis excluded, mimicking a fracture of the surgical neck, the errors of prediction for retroversion, inclination, height, radius of curvature and posterior and medial offset of the head of the humerus were 3.8° (± 2.9°), 3.9° (± 3.4°), 2.4 mm (± 1.9 mm), 1.3 mm (± 0.9 mm), 0.8 mm (± 0.5 mm) and 0.9 mm (± 0.6 mm), respectively.

The authors also acknowledge the following limitations. The proposed algorithm only estimated the proximal humeral anatomy. Also, for surgical purposes, images showing at least 6cm of the proximal humerus need to be available to the surgeon. Lastly, the testing dataset was acknowledged by the authors for possibly not being accurate enough in terms of capturing enough of the anatomical variation of the humeri throughout the population.

[Vla+18]. The aim of this study was to evaluate whether a statistical shape model (SSM) has the potential to predict accurately the pretraumatic anatomy of the humerus from the post-traumatic condition. To achieve this goal, the authors extracted one hundred 3D triangular surface mesh models from paired (50 left and 50 right) cadaveric humeri without any pathological conditions. The mesh correspondence for the SSM was created with use of the **non-rigid Gaussian Process Morphable Models** [Lüt+16] algorithm. Subsequently, all of the meshes were rigidly aligned using the **Procrustes alignment** [Ume91] approach to a single humerus model. However, no further information on the choice of the single humerus model was given in the study. The SSM itself was created by performing **principal component analysis** [Jol86] upon the rigidly aligned meshes.

To test out the precision of the created SSM, segments of the humerus of predefined length excluding the part to predict were synthetically generated and evaluated by the distal and proximal humeral prediction (d-HP and p-HP) errors. These errors were defined as the deviation of the predicted humerus model from the original.

The measured mean p-HP error was 3.8° ± 1.9° and the mean d-HP error 5.5° ± 2.9° while using 85% of the proximal part of the original humerus. The limitation of the SSM approach is the reliability of the restored model heavily depends on the amount of the original humerus provided in the reconstruction. To give an example, when using 50% of the original humerus, the mean p-HP error raised to 8.9° ± 5.6° and the mean d-HP error raised to 9.9° ± 5.7°.

Implementation 3

In this chapter we will focus on the implementation of the reconstruction process using methods introduced in chapter 1.2. The implementation will be in form of a **module** into the **3D Slicer** application, details of which will be discussed in section 3.1.

3.1 3D Slicer

In this section, we will take a quick tour of the application for which we will be developing the pipeline. We will get back to the reconstruction implementation later in section 3.4.1. As noted at the beginning of this chapter, the reconstruction will be implemented in form of a module for the 3D Slicer [Fed+12]. 3D Slicer is a free open-source application for (bio)medical data visualization and manipulation available for MacOS (both ARM and x86), Windows, and various Linux-based distributions. This application offers a wide variety of operations such as segmentation, registration, transformation, annotation and many more. All of these operations can be performed upon 3D models and 2D medical images. Hereafter, we will be referring to 3D Slicer as just "Slicer". The version used for development was **5.6.1**.

After launching Slicer, the user is welcomed into the main UI of the application (Figure 3.2). The main point for interest for our usage is the **scene view** and the **module pane**. A scene is the main data structure containing data about all objects (also called **nodes**) which are currently loaded into the application and additional data about how different structures should be rendered.

All of the data is internally stored in a **Medical Reality Markup Language** (*abbrv. MRML*) [Fed+12] which is a open-source data model for storing this type of data. As Slicer is the only major application using it to the day of writing this thesis, it is being maintained mostly in the Slicer repository, but the library is developed independently. The MRML is based on the Visualization Toolkit (*abbrv. VTK*, `https://vtk.org`) file format, which is a very popular format used in the 3D computer graphics department. Slicer also offers support for other 3D data formats, such as the **STL** or the **PLY** file formats.

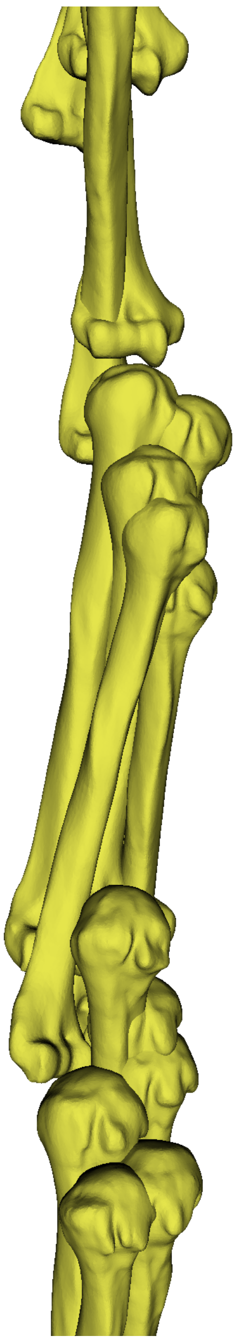


Figure 3.1: Subset of male humerus models from the dataset A imported into a 3D scene

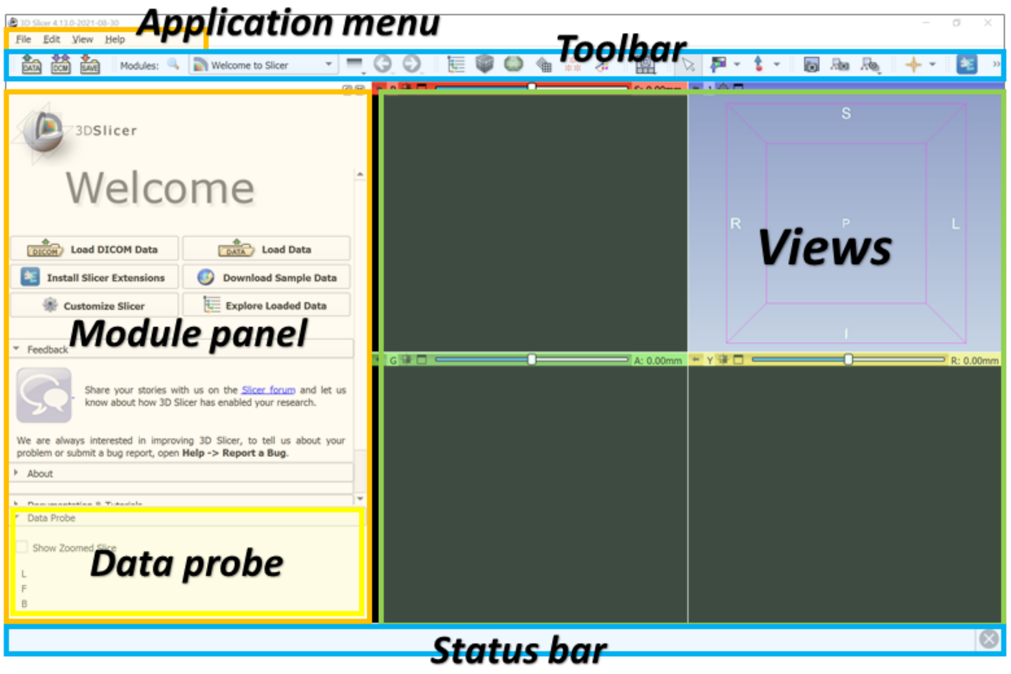
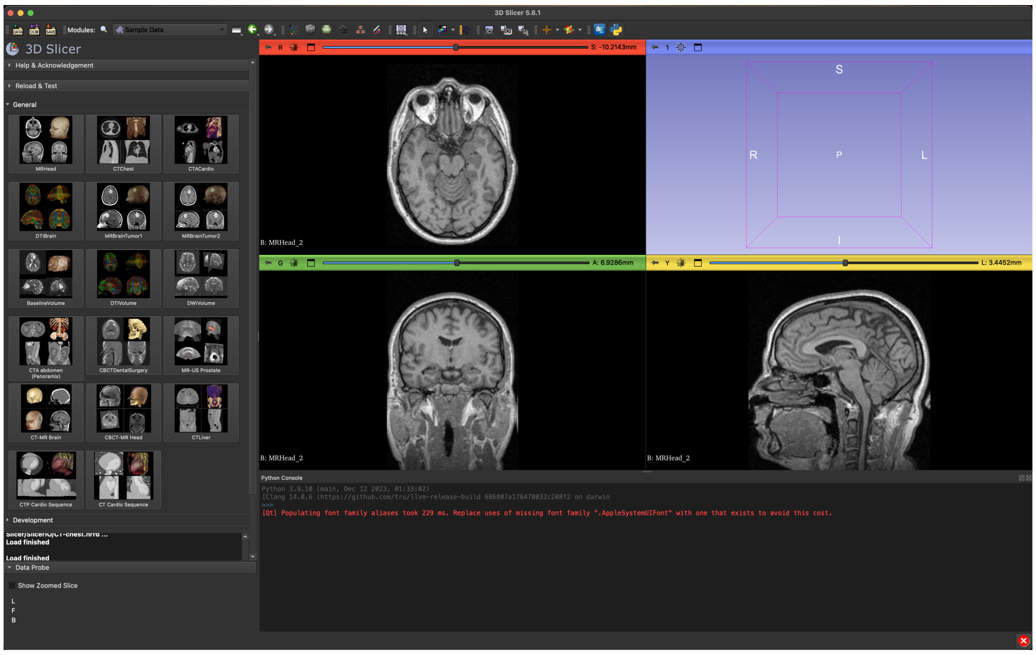

Application menu
Toolbar
Module panel
Views
Data probe
Status bar
File Edit View Help
Modules:
Welcome to Slicer
3DSlicer
Welcome
Load DICOM Data
Load Data
Install Slicer Extensions
Download Sample Data
Customize Slicer
Explore Loaded Data
Feedback
Share your stories with us on the Slicer forum and let us know about how 3D Slicer has enabled your research.
We are always interested in improving 3D Slicer, to tell us about your problem or submit a bug report, open Help -> Report a Bug.
About
Data Probe
Show Zoomed Slice
S
R
L
P
I

Figure 3.2: Slicer user interface. In the center of the UI is the main 3D scene with light blue background. The left side of the UI shows controls for the currently selected module. The top bar shows all of the available tools for modifying the objects (also called **nodes**) present in the current scene. On the bottom is a status bar showing the status of the last operation. In the lower left corner is the Data Probe used for displaying information about view content at the position of the mouse pointer. Source: `https://slicer.readthedocs.io/en/latest/user_guide/user_interface.html`

A scene can contain one or more **nodes** of different types. Currently supported types by Slicer are following:

- Data nodes - node carrying basic data properties, i.e. the models themselves. As such data can be presented in different ways, the data nodes are divided into subtypes of which we will focus only on the following
  - Model - surface meshes
  - Markup - simple objects such as points, lines. etc. Used for landmarks
- Display nodes - custom properties of a data node, for instance a color of a node
- Storage nodes - storing options for a node
- View nodes - usually define aspects of the scene itself, for example the background color



```
Python 3.9.10 (main, Dec 12 2023, 01:33:02)
[Clang 14.0.6 (https://github.com/tru/llvm-release-build 686807a176470032c208f2 on darwin
>>>
[Qt] Populating font family aliases took 229 ms. Replace uses of missing font family ".AppleSystemUIFont" with one that exists to avoid this cost.
```

Figure 3.3: Example of how a Slicer scene might look like

- Plot nodes - used only for plotting and charting

One object like a 3D model can have multiple nodes where each represent a different aspect or a property of the viewed data. Each scene can then contain various combinations and hierarchy of different types of nodes.

3.2 Source Datasets

This thesis utilizes two humerus surface mesh datasets provided by third parties. The first dataset, "QIN-HEADNECK" ([Bei+15], [Fed+16], [Cla+13]; referred to as **Dataset A** hereafter) was provided to us by the authors of "Glenohumeral joint reconstruction using statistical shape modeling" [Hua+21], Yichen Huang and David Ackland. The surface models are based on scans from the Cancer Imaging Archive. We have also obtained permission from the Cancer Imaging Archive to use these models. The dataset consists of 54 (60 initially; 6 duplicated were removed) reconstructed 3D humerus models, out of which 24 models represent female humeri with the mean of the age of the subjects at 58.6 ± 10.2 years and 30 models represent male humeri with the mean of the age of the subjects at 61.0 ± 10.5 years. All of the models are represented as triangle meshes (recall section 1.2.2). For most of the subjects the right-hand humerus model was available (filename contains _*RH*_). However, in some cases, two scans (left and right humeri) were available for a single subject. For those, only the right-hand model was used to ensure that one subject is processed only once. For some subjects, only their left hand scan was made so the reconstructed source model was mirrored in order to minimize the amount of possible shape variation in the dataset (name contains *_RHM_*). The source CT images were not shared due to data privacy concerns.

The second dataset (referred to as **Dataset B** hereafter) is an open-source dataset originally used in the paper "Biplane fluoroscopy derived humerus and scapula kinematics during arm elevation and rotation" [Ali+21]. It consists of 18 reconstructed 3D humerus models, out of which 8 models are female humeri with the age of the subjects in the range of <22; 66> and the mean at 37 years and 10 models represent male humeri with the age of the subjects in the range of <22; 66> with the mean at 41.5 years. All of the models are also represented as triangle meshes. The original models of this dataset are available both in the CT coordinate system and with humeral anatomical coordinate systems aligned to the global coordinate system. It was decided to use the data in the CT coordinate system to better assess the effectiveness of our global registration algorithm from typical CT space, without any pre-alignment.

3.3 Statistical Shape Model Creation

This section will focus solely on the exact steps of creating the mean model using the 3D Slicer application. To recall, the steps for creating a mean model are as following (section 2.1.1):

1. Model landmarking
2. Mesh alignment
3. Point correspondence analysis
4. Mean mesh model generation

3.3.1 Model Landmarking

The model landmarking was done using an automated tool called **MALPACA** [Zha+22], which is a part of the **SlicerMorph** project [Rol+21]. SlicerMorph is a set of tools used for, as the name suggests, various morphology tasks. This toolset is distributed in the official Slicer extension repository and can be installed using the Extension Manager right in Slicer. The concrete tool (MALPACA) is a specialized tool for automatic landmark projecting from a source mesh to a target mesh. The landmarks placed on each humerus model can be separated into 3 main areas - the proximal head, the humeral shaft and the distal end. On the proximal head, **6 landmarks** have been placed - **humeral head center, greater tubercle, intertubercular sulcus, lesser tubercle, posterior anatomical neck and medial anatomical neck**. On the humeral shaft there is only the **deltoid tuberosity** landmark as it is very hard to identify other discrete landmarks on the humeral shaft. On the distal end, **6 landmarks** have been placed - **lateral epicondyle, posterior capitulum, medial epicondyle, olecranon fossa, posterior and distal trochlea**. Visualization of placement of some of the landmarks is visible in Figure 2.1.

MALPACA Pipeline. The input for MALPACA is a single, manually landmarked mesh and a target mesh onto which the source landmarks should be Firstly, both of the meshes are converted into two point clouds. Then an initial rigid alignment between the point clouds is calculated. After that, the meshes are subjected to the deformable registration. Finally, the source landmarks are projected onto the target using a point correspondence analysis.

Notably, the pipeline utilizes methods we have already covered in section 2.1.1, therefore no additional in-depth explanations are needed. Each landmark transfer was then visually inspected and adjusted, if necessary to maximize the accuracy of the point placement relative to the anatomical aspects of the humerus.

3.3.2 Creating the Mean Model

Usually, all of the steps to create the mean model (as discussed in section 2.1.1) would be implemented manually. However, in October 2023 a new paper called “A Dense Correspondence Analysis Toolkit for Shape Analysis” [RM23] was published. In this paper, the authors provide methods for complete shape analysis, for instance rigid alignment, PCA [Jol86], mean model generation and more. Alongside this paper, the open-source implementation of all of the methods mentioned in the paper was published on GitHub (link). The implementation is, fortunately enough, tailored specifically for Slicer, therefore it is a perfect tool to fit our needs. To install the DeCA module, a written step by step manual is available in Appendix A.

When the DeCA module is opened, the user is greeted with the module’s main UI (Figure 3.4). The UI consists of tabs, each serving a different purpose. For our needs, we will be focusing only on the “**Rigid alignment**” and “**Generate Mean**” tabs.

Mesh alignment. The next step after landmarking (section 2.1.1) is the model alignment. In DeCA, we will switch to the “Rigid Alignment” tab where are couple parameters to fill out.

- In “Base model” we put the path to one of the source meshes (this will be the base of the alignment)

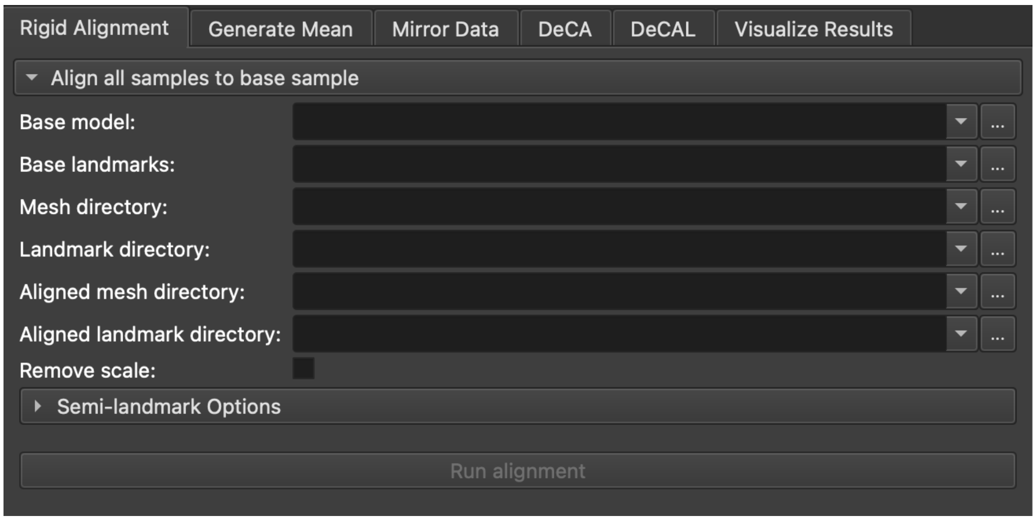


Figure 3.4: Slicer DeCA module main UI

- In “Base landmarks” we put the path to the file containing landmarks of the chosen base model
- In “Mesh directory” we put the path to the folder containing the rest of the files representing the source mesh dataset
- In “Landmark directory” we put the path to the folder containing the rest of the file containing landmarks of the source mesh dataset
- In “Aligned mesh directory” we put the path to the folder to which we want to export the aligned source meshes
- In “Aligned landmarks directory” we put the path to the folder to which we want to export the aligned source dataset landmarks

As the base model, its selection was based on a visual inspection of each of the source models, out of which three models with the highest detail in the distal region of the model was chosen. With each of these three models, a mean model was created (recall section 3.3.2) with the initial model chosen as a base. The mean model used for the analysis of this thesis was the one that had the most amount of detail in the distal area preserved. This approach can be considered limiting in terms of including a selection bias by the author, therefore can be considered as an area for enhancement (later discussed in section 5).

With all of these parameters filled out, we just need to click the **Run alignment** button, which will align all of the provided meshes to the base model. With the aligned dataset we can proceed to the mean model generation.

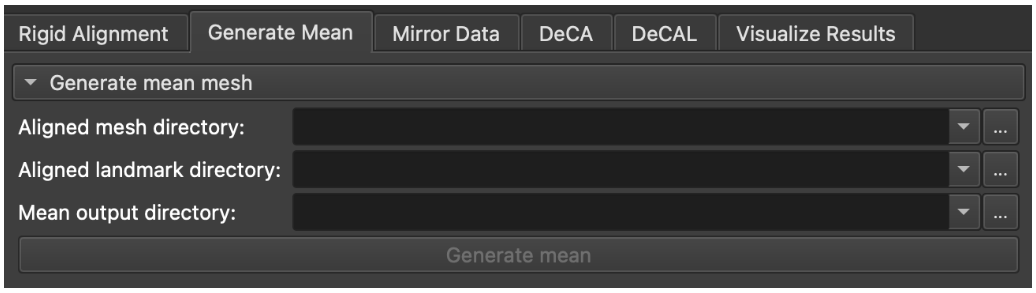


Figure 3.5: Slicer DeCA module **Generate Mean** tab

Generating the mean model. To generate the mean model, we need to switch to the **Generate Mean** tab of DeCA module (Figure 3.5). We need to fill out the following parameters:

- In "Aligned mesh directory" we provide the respective path from the previous alignment step
- In "Aligned landmarks directory" we provide the respective path from the previous aligned step
- In "Mean output directory" we provide the path to directory to which the SSM will be saved

After pressing the **Generate mean** button, we have our mean model available in the "Mean output directory". In our exact case, we have created separate SSM for both male and female. The source dataset A was divided (per sex) into 23 of models being used for the SSM generation, while reserving 13 for further optimizations (discussed in chapter 5). The resulting mean models are depicted in chapter 4, Figures 4.1 and 4.2.

3.4 Module Implementation

In the following sections, we will explore the concrete implementation of the reconstruction pipeline steps as enumerated in 3.4.1. This newly proposed reconstruction pipeline was implemented as a new module in Slicer and will be hereafter referred to as the **SlicerBoneMorphing** module. It is available as an open-source on GitHub (link). The installation process is identical to the DeCA module installation (recall section A). For details, please refer to the B attachment to this paper.

3.4.1 Reconstruction pipeline proposition

In the pipeline, we will presume that the we have the mean model already prepared, for instance as described in section 3.3. In the pipeline, we will refer to the **source** model as the mean model and the **target** as the model to be reconstructed. The proposed pipeline will consist of the following steps:

1. Source and target model preprocessing - downsampling and computing fast feature point histograms (FPFH)
2. Source to target initial rigid alignment - RANSAC registration with ICP fit enhancement
3. Source to target deformable registration for reconstruction
4. Target reconstruction
5. Result postprocessing

Before reconstructing the target (impartial) model, the SSM and the target models are pre-aligned in order to maximize the probability of reaching the best reconstruction results. This pre-alignment stage utilizes the Random Sample Consensus (RANSAC) algorithm for "raw" alignment and the Iterative Closest Points (ICP) algorithm for the alignment enhancement. For the target reconstruction, the Bayesian Coherent Point Drift (BCPD) algorithm was utilized.

3.4.2 Slicer Modules and Extendability

Slicer offers extendability throughout its own application programming interface (*abbrv. API*). This interface is available for external software packages (in the community referred to as **modules**) which represent a standalone deployable unit into the Slicer application. The API is offered for **Python** and **C++**. Majority of the Slicer's source code is written in C++, however Slicer installation includes a fully featured own Python environment, therefore no further installation is required from the user.

For starters, a command line interface (*abbrv. CLI*) is available as part of the Slicer's GUI for command-like interaction with the whole application. In the official documentation (link), authors of Slicer advise to use the CLI only for commandsyntax testing and simple scene manipulation. For any more complicated interactions, it is advised to create a separate module. Based on the selection of the language, Slicer differentiates between **Loadable** and **Scripted** modules.

3.4.2.1 Loadable Modules

With C++, a developer can create a **loadable module**. Loadable modules are C++ plugins that are built against Slicer's C++ core. When a loadable module is created and built, it is being distributed and loaded as a shared library. Loadable modules are recommended by the authors of Slicer only for complex andor performance-critical interactive components or internal infrastructural widget (e.g. low level GUI components). In C++, the developer has **full access** to the Slicer's API and **full control** over the Slicer UI and all internal structural components. Simple, Doxygen-style documentation of available classes and methods inside the Slicer's API is available (link). However, most available tutorials and the rest of the official documentation refers rather to the scripted module development. Overall, when not strictly necessary, it is advised, especially for developers not familiar with the Slicer's API, to proceed with scripted modules.

3.4.2.2 Scripted Modules

Scripted modules are an easier to develop alternative to loadable modules. As noted earlier, Slicer comes with its own fully featured Python environment with which the developer can interact either through the CLI or with an official **Slicer Python package**. This package is available **only** inside the Slicer and is not distributed openly using Python's popular package manager pip. This unfortunate aspect can slow down the prototyping process as the module needs to be loaded in Slicer and not only mocked outside of the application, however the debugger can be attached to a modern style IDE like VSCode or PyCharm. As with loadable modules, when developing using Python, the Slicer API is **fully accessible**. As opposed to the loadable modules, Python does **not** have a full control over the Slicer UI or any of the internal structures. This means that only the GUI of the module itself can be adjusted.

As Python is very popular in this research field, it offers many useful methods and algorithms in the form of **packages**. Packages are somewhat of an analogy of libraries in languages like Java or C++. Packages can be either managed by the Python

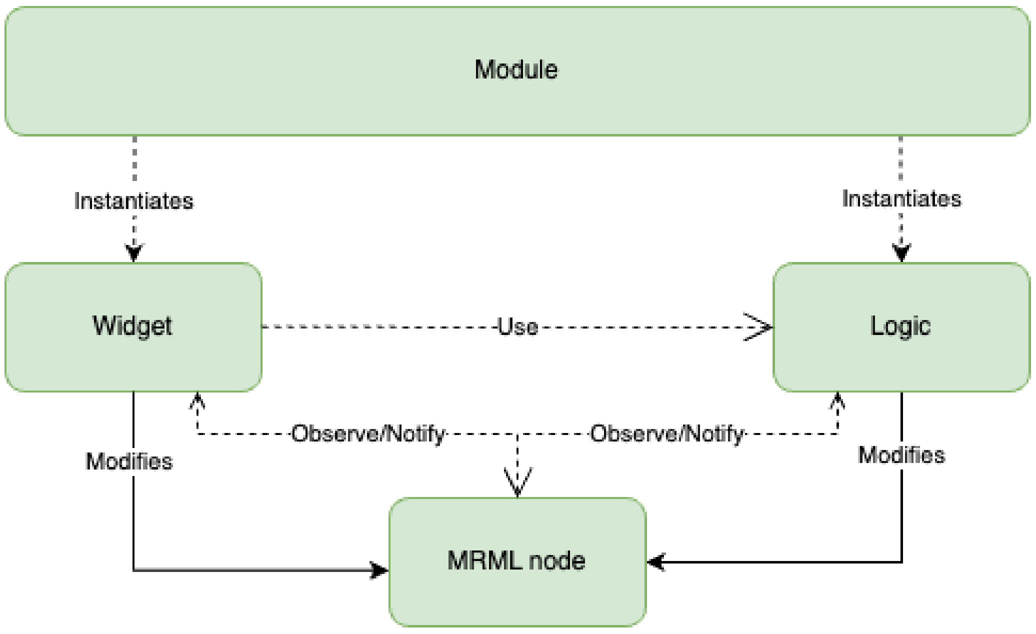


Figure 3.6: General schema of scripted modules in Slicer

package manager (*abbrv.* pip) or manually imported into the codebase. When using *pip*, the packages are usually downloaded from the Python Package Index *abbrv. PyPI* (link, which is the official package repository.

Every scripted module in Slicer must abide to the structural design depicted in Figure 3.6. Before proceeding to the exact implementation, we need to create a statistical shape model out of the provided source dataset, which will be the topic of the following sections 3.3. Details of the reconstruction start with section 3.4.

3.4.3 Prerequisites

The SlicerBoneMorphing module utilizes the Open3D library (link) which is an open-source library for 3D data manipulation and is not a part of a Slicer installation. To install this dependency, we can use the embedded *pip* package manager. This process is automated and the dependencies are checked on every start of the Slicer application. If the Open3D dependency is missing, it will be automatically installed.

To use the SlicerBoneMorphing module, any models that ought to be used with it need to be imported into the currently opened scene as the module.

3.4.4 File Structure

As the module is of type **Scripted** (as discussed in section 3.4.2.2), it abides to the following file structure. In the root of the module's repository, a folder with the name of the module containing all of the sources is present. The only other file that is needed by Slicer is the **CMakeLists.txt**. This file resembles settings and hints for the compiler to know where to look for the source files and how the module is organized. In here there are also some metadata present about the module, for instance module dependencies, authors, category and others.

3.4.5 Module Overview

After the SlicerBoneMorphing module is installed successfully and selected, the user is greeted with the main interface as shown in Figure 3.8. The module is divided into 4 main sections: **Input, Preprocessing parameters, BCPD parameters** and **Postprocessing parameters** (Figure 3.9). We will explore each section and describe its user interface and functionality under the hood in their appropriate following sections. To ease up the source code referencing, every method referred to is as a part of the UML diagram of the module (Figure 3.7). All of the functionality which will be talked about happens after the pressing the Generate button which executes the **generate_model** method. All of the default values of configurable parameters (except BCPD) were chosen empirically based on visual inspection of the results with different configurations. In case of BCPD, the default values are same as in the O. Hirose's implementation.

3.4.5.1 Input Section

The input section is fairly straight forward in terms of UI, however inside the module, a couple of operations need to be performed. As hinted in section 3.4.3, our models are represented as a **vtkMRMLNode** within a scene. Open3D, on the other hand, needs the meshes in an **open3d.geometry.TriangleMesh** format, therefore a conversion method needed to be implemented. Internally, this method is called *convert_model_to_mesh*. The vertices and triangle indices are extracted from the vtkMRMLNode and by using *vtk_to_numpy* function, we convert the them to **numpy** data structures and out of those we are finally able to construct a TriangleMesh. The TriangleMesh is a **clone** of the source mesh, therefore the source remain unchanged.

3.4.5.2 Preprocessing Section

Based on the previous section (3.4.5.1), the meshes are represented as TriangleMeshes. However for the preprocessing section we only need the point set, therefore for this section we will be working only with the **point clouds**. After converting the source and target models into a more convenient format, we can preprocess the meshes. Internally, all of the preprocessing is covered in the *preprocess_model* method.

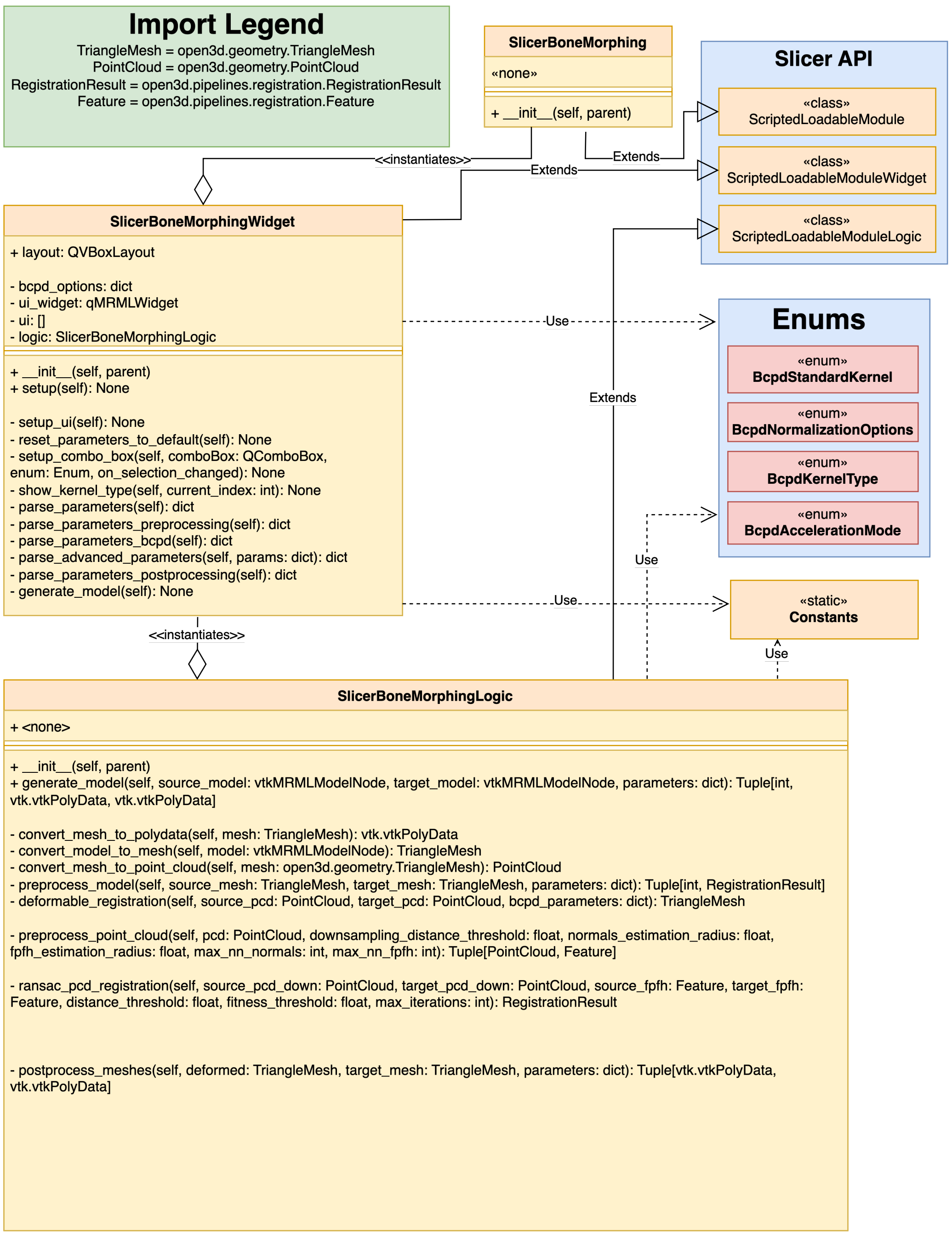

Import Legend
TriangleMesh = open3d.geometry.TriangleMesh
PointCloud = open3d.geometry.PointCloud
RegistrationResult = open3d.pipelines.registration.RegistrationResult
Feature = open3d.pipelines.registration.Feature
SlicerBoneMorphing
«none»
+ __init__(self, parent)
Slicer API
«class»
ScriptedLoadableModule
«class»
ScriptedLoadableModuleWidget
«class»
ScriptedLoadableModuleLogic
<<instantiates>>
Extends
Extends
Extends
SlicerBoneMorphingWidget
+ layout: QVBoxLayout
- bcpd_options: dict
- ui_widget: qMRMLWidget
- ui: []
- logic: SlicerBoneMorphingLogic
+ __init__(self, parent)
+ setup(self): None
- setup_ui(self): None
- reset_parameters_to_default(self): None
- setup_combo_box(self, comboBox: QComboBox, enum: Enum, on_selection_changed): None
- show_kernel_type(self, current_index: int): None
- parse_parameters(self): dict
- parse_parameters_preprocessing(self): dict
- parse_parameters_bcpd(self): dict
- parse_advanced_parameters(self, params: dict): dict
- parse_parameters_postprocessing(self): dict
- generate_model(self): None
Use
Enums
«enum»
BcpdStandardKernel
«enum»
BcpdNormalizationOptions
«enum»
BcpdKernelType
«enum»
BcpdAccelerationMode
Use
Use
«static»
Constants
Use
<<instantiates>>
SlicerBoneMorphingLogic
+ <none>
+ __init__(self, parent)
+ generate_model(self, source_model: vtkMRMLModelNode, target_model: vtkMRMLModelNode, parameters: dict): Tuple[int, vtk.vtkPolyData, vtk.vtkPolyData]
- convert_mesh_to_polydata(self, mesh: TriangleMesh): vtk.vtkPolyData
- convert_model_to_mesh(self, model: vtkMRMLModelNode): TriangleMesh
- convert_mesh_to_point_cloud(self, mesh: open3d.geometry.TriangleMesh): PointCloud
- preprocess_model(self, source_mesh: TriangleMesh, target_mesh: TriangleMesh, parameters: dict): Tuple[int, RegistrationResult]
- deformable_registration(self, source_pcd: PointCloud, target_pcd: PointCloud, bcpd_parameters: dict): TriangleMesh
- preprocess_point_cloud(self, pcd: PointCloud, downsampling_distance_threshold: float, normals_estimation_radius: float, fpfh_estimation_radius: float, max_nn_normals: int, max_nn_fpfh: int): Tuple[PointCloud, Feature]
- ransac_pcd_registration(self, source_pcd_down: PointCloud, target_pcd_down: PointCloud, source_fpfh: Feature, target_fpfh: Feature, distance_threshold: float, fitness_threshold: float, max_iterations: int): RegistrationResult
- postprocess_meshes(self, deformed: TriangleMesh, target_mesh: TriangleMesh, parameters: dict): Tuple[vtk.vtkPolyData, vtk.vtkPolyData]

Figure 3.7: SlicerBoneMorphing UML diagram

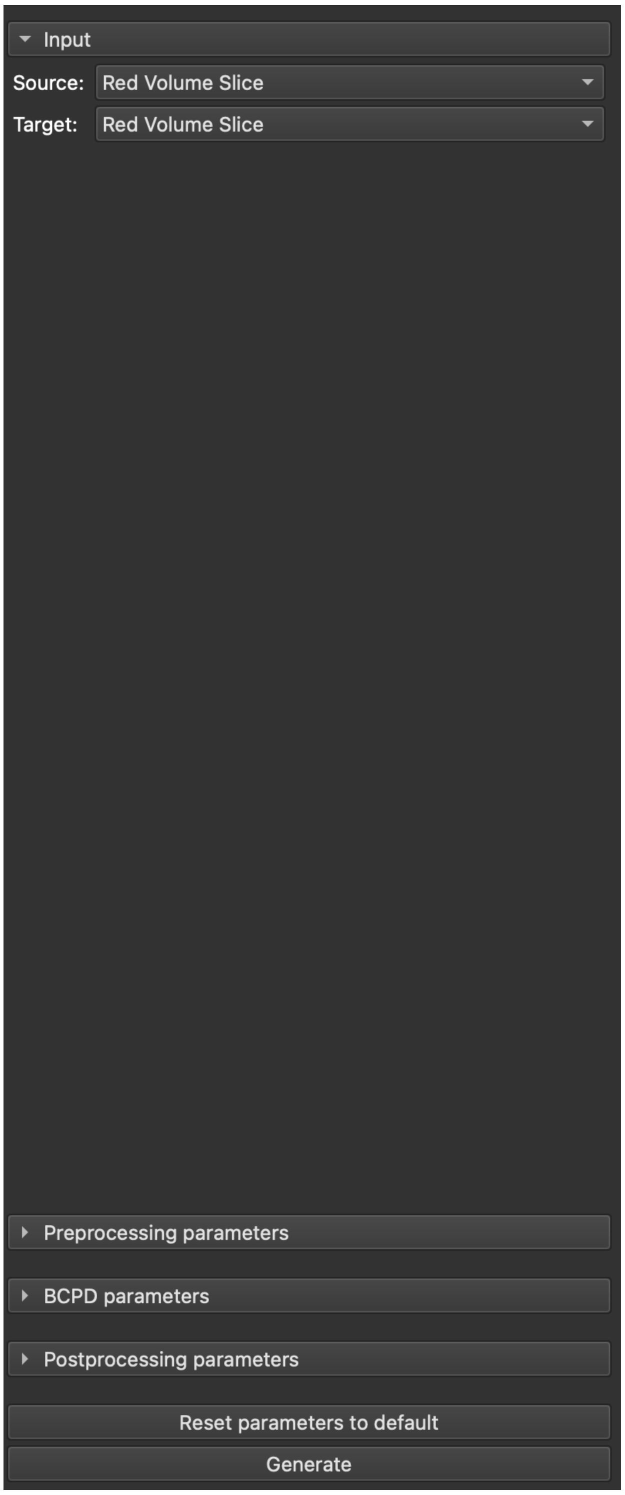


Figure 3.8: Main user interface of the SlicerBoneMorphing module

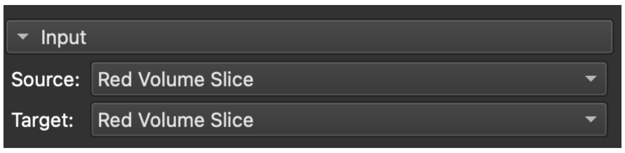

Input
Source: Red Volume Slice
Target: Red Volume Slice

Figure 3.9: SlicerBoneMorphing UI input section

Downsampling. In this stage we apply voxel downsampling on the point cloud based on the distance threshold set by the user. The downsampling step is included to mitigate the computational complexity of some of the methods discussed in section 1.2. The downsampling "groups" the points into voxels based on the "Downsampling distance threshold" (the size of the voxels determined automatically by Open3D). Each occupied voxel then generates exactly one point by averaging all points inside. If downsampling is not desirable, the user can configure the "Downsampling distance threshold" to be lower than the lowest distance between two adjacent points.

Normals estimation and Fast Point Feature Histogram. To compute the Fast Point Feature Histogram (recall section 2.2.5), we need to estimate the normals of the point clouds. For that we will use the *open3d.geometry.PointCloud.estimate_normals* method. This method computes the normal by finding adjacent points and calculating the principal axis of the adjacent points using covariance analysis. The required parameters are **radius** and **max number of neighbours**. These values can be adjusted by the user using the "Normals estimation radius" and "Normals estimation max neighbours", respectively.

Using the estimated normals of the point clouds, we can compute the Fast Point Feature Histogram using the *open3d.pipelines.registration.compute_fpfh_feature*. The required parameters are again a **radius** and **max number of neighbours**, though they do not need to be the same values as for the normals estimation.

RANSAC and ICP registration. With the histograms calculated, we can proceed to the registration process. Firstly, the RANSAC registration is performed as the global registration process to roughly align the models. Open3D again offers a concrete method *open3d.pipelines.registration.registration_ransac_based_on_feature_matching*. Secondly, ICP is performed on the pre-aligned models to limit the amount of rigid registration but refine the alignment as much as possible.

For RANSAC, three different parameters are available for the user to adjust: "Max iterations", "Distance threshold" and "Fitness threshold". "Max iterations" is the maximum number of iterations may be reached. "Distance threshold " is the **maximal** threshold for two points that can be considered corresponding. "Fitness threshold" is the **minimal** fitness for the registration to be considered a fit. It can also be interpreted as a minimum percentual alignment match between the models.

For ICP there is only a single adjustable parameter "ICP Distance threshold" and it represents the **maximum** distance in which a neighbour is being searched for.

3.4.5.3 BCPD Section

As discussed in section 2.2.3, the BCPD's implementation is publicly available on O. Hirose's GitHub. The source code is written in C and has been seemingly obfuscated for the purposes of With no documentation available solely for the source code, the module uses a manually pre-built binary executable. The binary version used by the SlicerBoneMorphing was built by the author of this paper for Windows 10 (x86), Linux distributions (x86) and MacOS (ARM/x86) using the Makefile provided in the repository. For MacOS, only the x86 version of the binary is available since Slicer as of version 5.6.1 runs in the Rosetta translation layer, however it has been tested by the author that the module is compatible with Apple M-chip based computers chips as well. The binaries are distributed as a part of the module, therefore no user interaction is required.

A thought has also been given to creating a shared library rather than using the BCPD as a **black-box** binary, and by utilizing the Python-C bindings, using the internal C functions. This has also been tested by the author and did initially work, however slight compatibility issues with several C libraries were encountered. Combined with the fact that the BCPD implementation was seemingly not created with such usage in mind, we have decided not to proceed in this manner any further.

Now, let us focus back on the SlicerBoneMorphing module. The BCPD section is internally represented by the *deformable_registration* method. The BCPD expects the input in a form of two text files (the source and target models, respectively) containing the coordinates for each vertex. Due to this requirement, the module exports the meshes as text files into the operating system's temporary file directory and passes the paths to the BCPD. Alongside the paths, a high number of parameters can be configured for the BCPD. Due to their relatively high count, their explanation will not be a part of this paper If necessary, please consult the user manual in the BCPD GitHub repository (link).

As expected, the BCPD returns the deformed generated model in a file form, thus a reverse process of importing takes place. We also create a clone of the result model which we merge directly with the source model to show the user the exact fit of the result. In other words, there are two models imported into the scene.

3.4.5.4 Postprocessing Section

After obtaining the resulting deformed fully generated model, we give the user an option to apply a bit of postprocessing if necessary as the part of the pipeline. In postprocessing, the module offers two operations - ***mesh simplification*** and ***mesh smoothening***. The settings for these methods corresponds to parameters “Clustering scaling” and “Smoothing iterations”

Mesh simplification. Mesh simplification is a process where we want to represent a high-density mesh by a low density while keeping the overall shape aspects. In other words, it can be understood as form of **downscaling**. For this we utilize the *open3d.geometry.simplify_vertex_clustering* which **averages** out each set of vertices. The amount of scaling is controlled by the user. If set to 1.0 (default value), no simplification is applied.

Mesh smoothing. Mesh smoothing is, as the name implies, a method to smooth the surface of a mesh. There are various methods available in Open3D. In our case we firstly apply a “rough” smoothing with the *open3d.geometry.filter_smooth_simple* method and then apply a second smoothing process using the *open3d.geometry.filter_smooth_taubin*[Tau95] method. The amount of smoothing is also controlled by the user. If set to 0 (default value), no smoothing is applied.

Results 4

In this section we will take a look into the results of the reconstruction process. To recall, 8 female and 10 male models (dataset B) were used for the testing of the reconstruction. All of the meshes needed to be **cropped** in their proximal head area to test the reconstruction. The cropping was a straightforward process and the source code can be found in Appendix C.

4.1 Mean Model Results

Firstly, we will take a look into the resulting mean models created using the DeCA Slicer module, as discussed in section 3.3.2. The resulting mean models are depicted in Figures 4.1 and 4.2.

Notably, the resulting mean models are very smoothed out, which is not a bad property, however the proximal humeral neck and the distal trochlea areas do seem slightly oversmoothed. The cause of is not completely known, however it is suspected that A) the source models (dataset A) were already smoothed out too much in order to achieve a better mean model. On the other hand, this is a sensibly fixable issue if enough highly detailed models are gathered.

4.2 Reconstruction Results

Now, we will focus on the result of the reconstruction process itself (with information from 4.1 in mind).

To test the functionality of our pipeline, a testing scenario was created. Instead of using the mean model, simply a model of a full humerus and a partial (unrelated) counterpart were taken. These sources are depicted in Figure 4.3. The models were chosen as a duo which visually did not resemble each other in terms of their anatomical properties while being as detailed as possible. Figure 4.4 shows the result of the reconstruction.

In the testing scenario, the results seem very promising. The reconstructed bone remains highly detailed in the generated proximal area with a bit of widening of

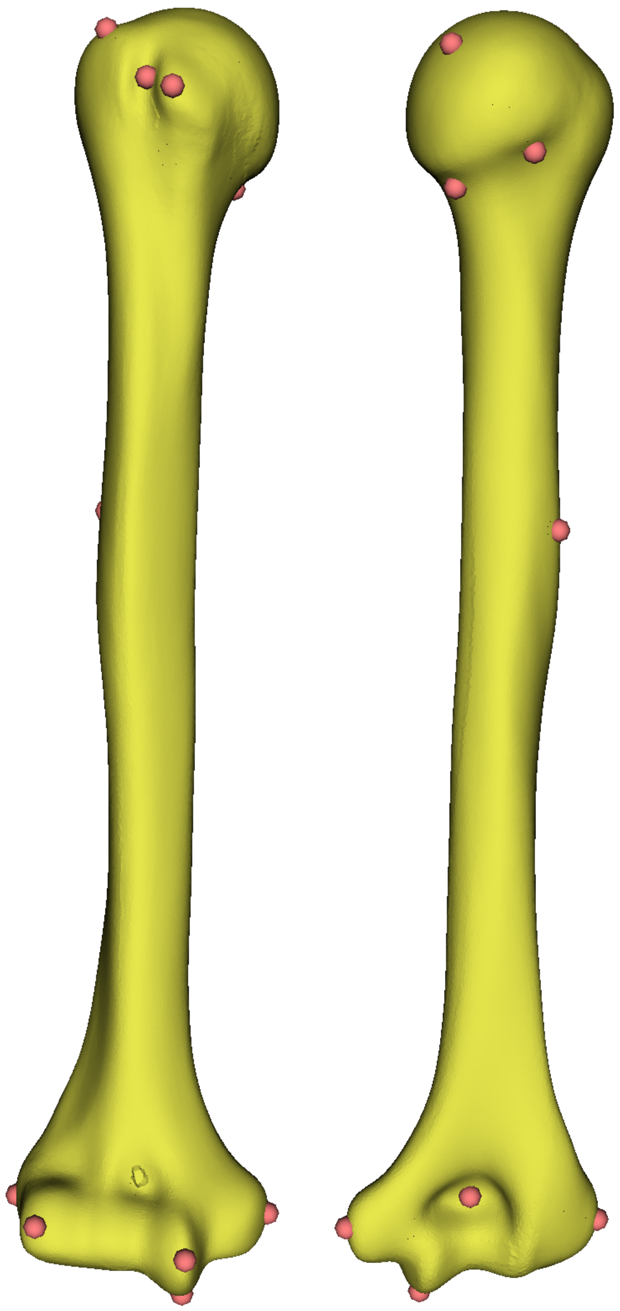


Figure 4.1: Male statistical shape model created using the DeCA module with its anatomical landmarks highlighted

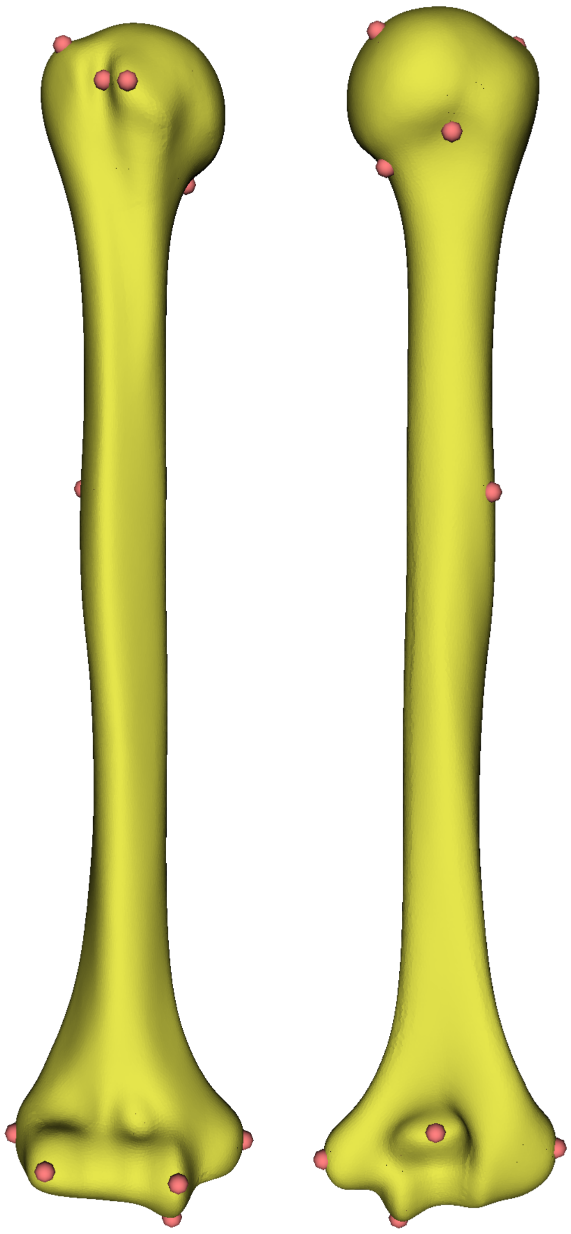


Figure 4.2: Female statistical shape model created using the DeCA module with its anatomical landmarks highlighted

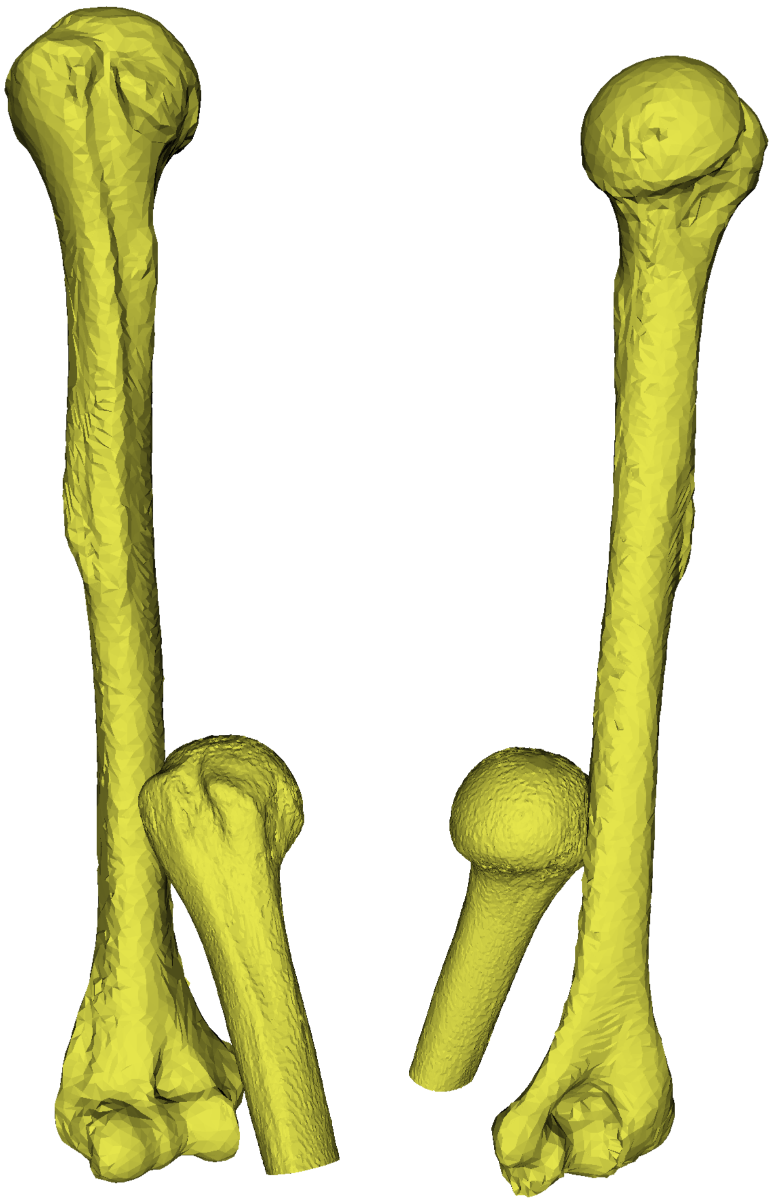


Figure 4.3: An “ideal” testing scenario testing source models from different angles

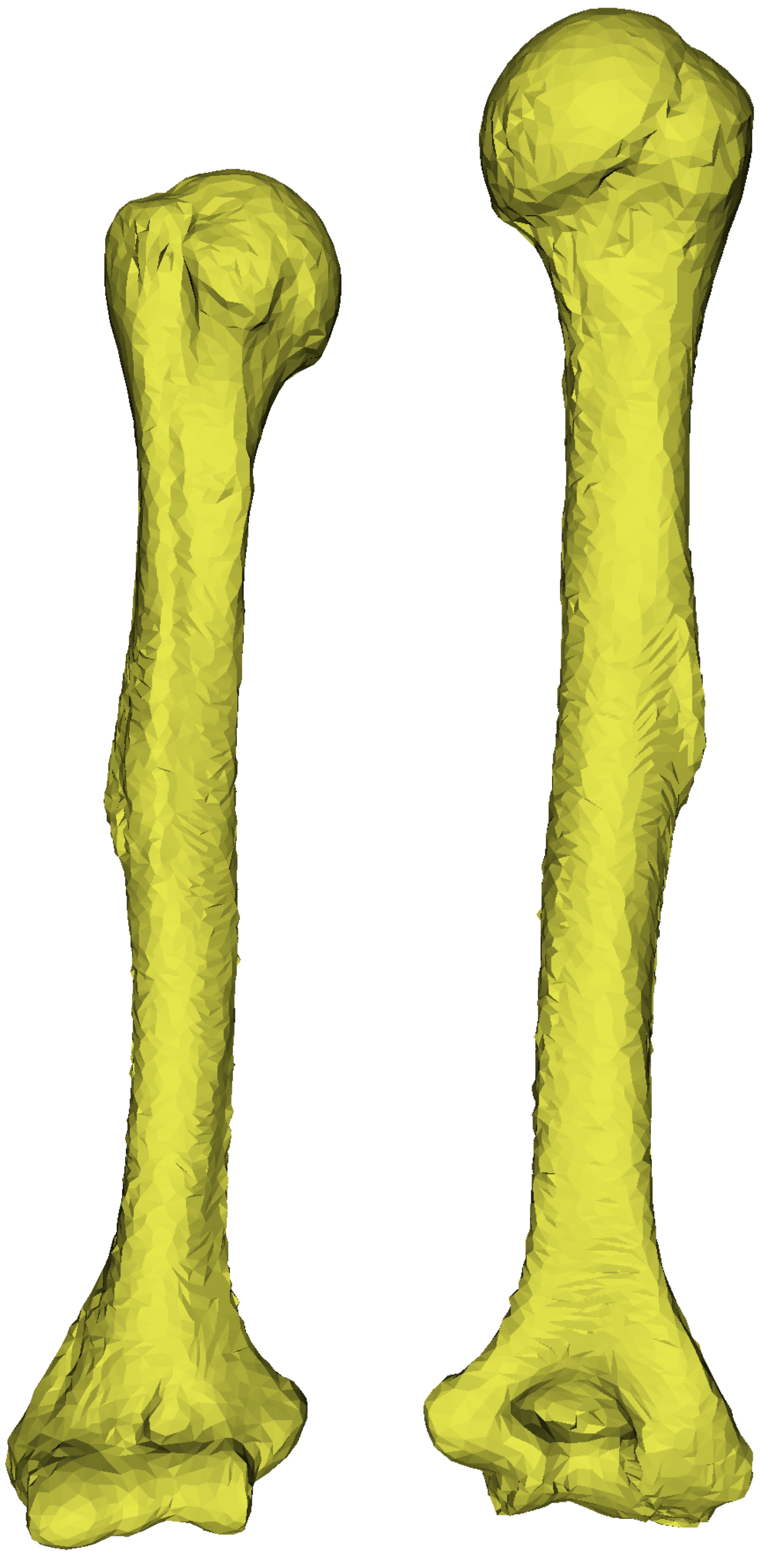


Figure 4.4: An “ideal” testing scenario result model from different angles

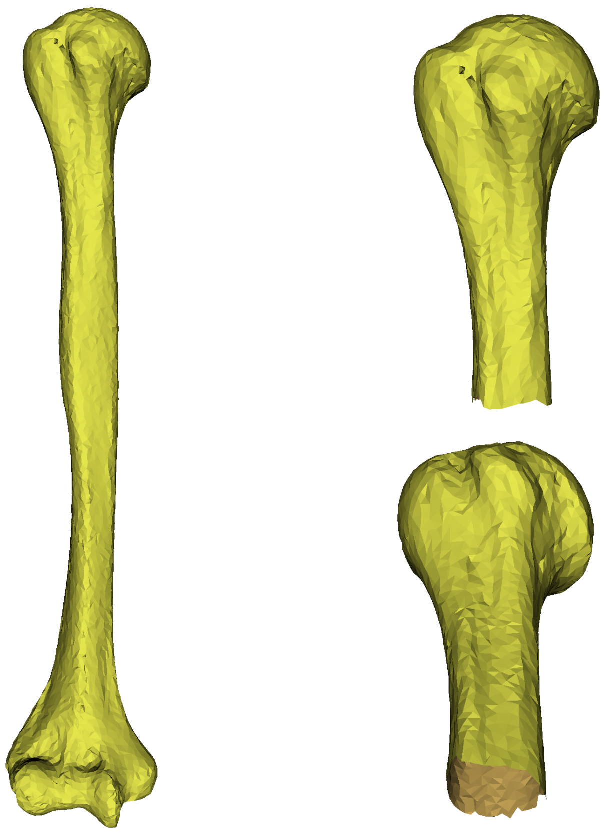


Figure 4.5: Testing model “O45_001_F_47_R_Humerus” with its cropped distal part

the distal part and the distal articular surface being slightly “squished”. Also, the model’s fit within the humeral shaft also contains a protrusion just like in the source model, which shows that the source model’s aspects are, in fact, being projected on the reconstructed models as anticipated. This scenario is good for testing out the reconstruction, however the results are not representative of a usual case since the mean model captures the overall variation of the humerus.

As noted in section 4.1, the results presented below did not yield as promising results.

As an example, a comparison of results for the model “O45_001_F_47_Humerus” are shown (Figure 4.5). All of the testing models have yielded very similar behaviour. The resulting reconstruction of this model is shown in Figure 4.6.

When visually inspected, all of the results shared a couple of common aspects. Firstly, the calculation of the rigid pre-alignment did require slight manual adjustments of the module’s **preprocessing** parameters for every case separately. To be exact, out of all of the available parameters, only the **downsampling distance threshold, normals estimation radius** and **normals estimation max neighbours** (recall the preprocessing section 3.4.5.2) parameters needed a slight adjust-

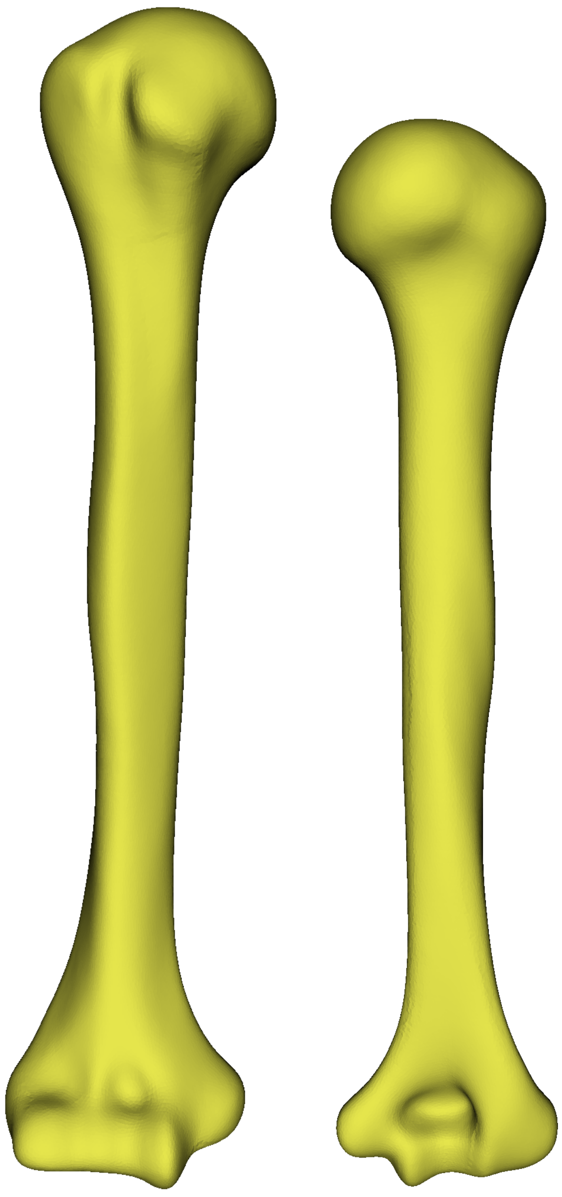


Figure 4.6: Result of the reconstruction of the model “O45_001_F_47_R_Humerus”

ment in their values. For each model, the exact values of these parameters are specified in a form of a text file inside the distributed archive. These adjustments were necessary as the cropped models were (in about 90% of the cases) being invalidly fitted to the distal end of the mean model (as depicted in Figure 4.7).

The cause of this error in rigid alignment was first presumed to be cropped models being non-uniform (jagged) and unclosed in the area of the humeral shaft cropping, these ends were confusing the RANSAC algorithm. To test this hypothesis, one of the models was cropped and then closed up using a modified version of the cropping script with the source code available in Appendix D. The resulting closed up mesh model is shown in Figure 4.8. However, even the closed up partial model yielded the same problems of invalid fitness. After adjusting the appropriate parameters, a better alignment fit was achieved, yet still to be improved (Figure 4.9).

All of the models also shared one common issue in terms of **length preservation**, or rather lackthereof. The main suspect is the difference in the amount of deformation of the distal humerus allowed by the settings of the BCPD algorithm and the actual amount of deformation needed for the proximal humeri to align properly. However this hypothesis remains to be investigated further and is therefore out of scope of this thesis.

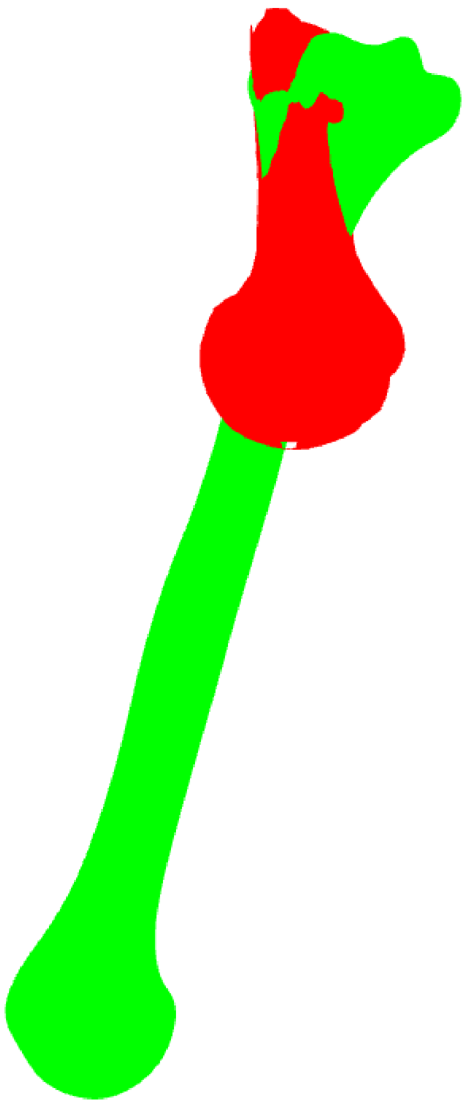


Figure 4.7: Example of an invalid rigid pre-alignment fit. Green model represents the mean model whereas the red model represents the partial target model

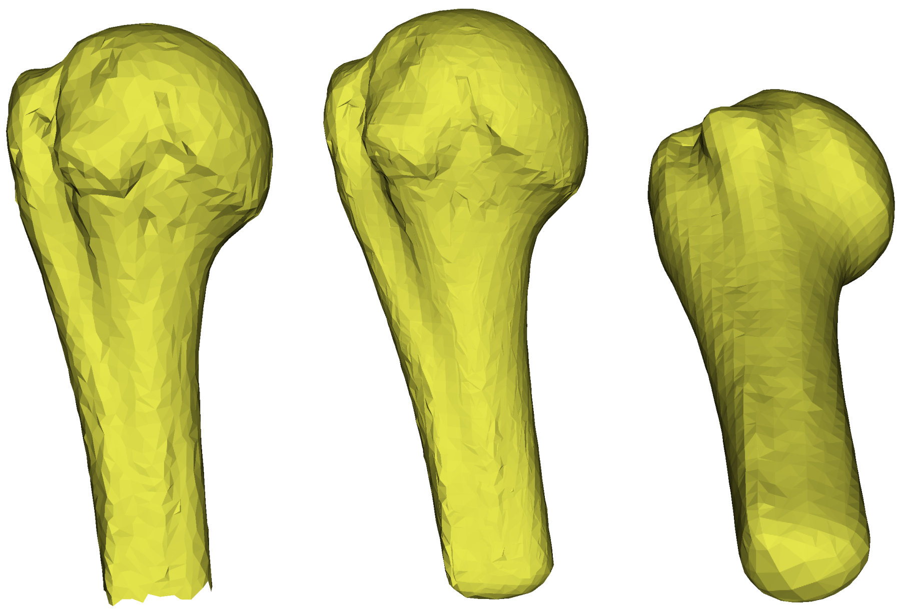


Figure 4.8: Cropping of the “U35_008_F_22_R_Humerus” model (left) and its “closed-up” version (center, right)

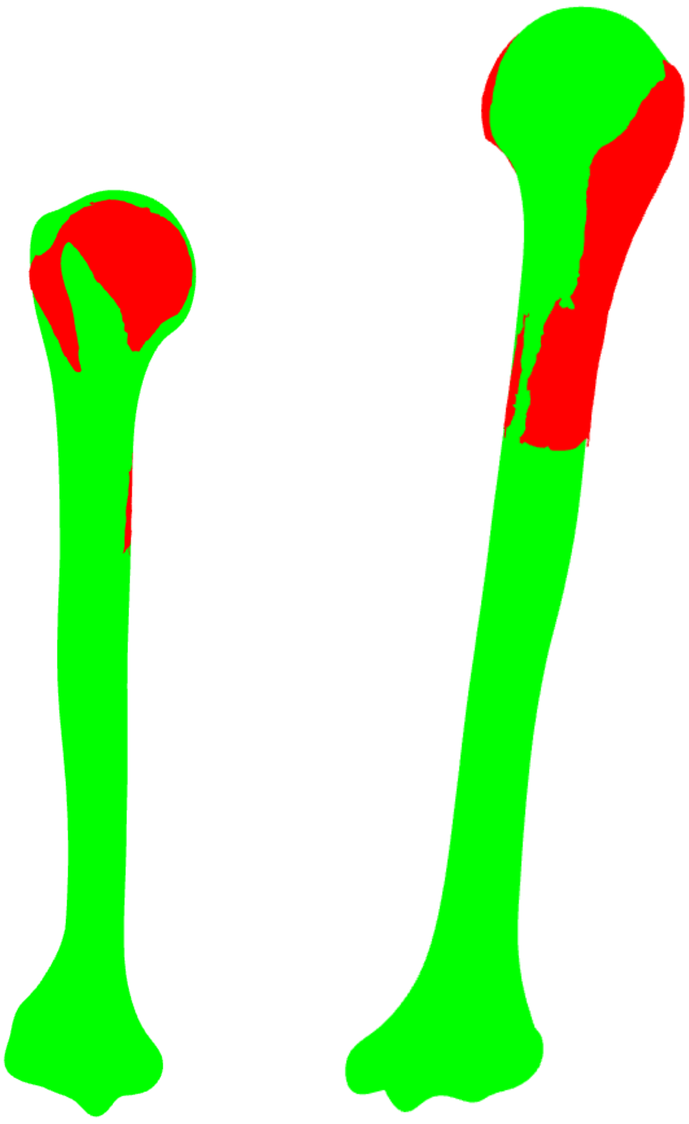


Figure 4.9: Example of an improved rigid alignment after adjusting some of the module's parameters

Conclusion and Discussion

5

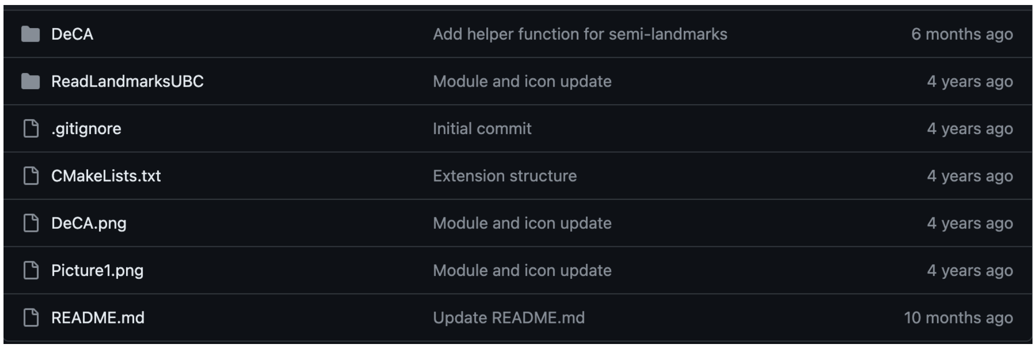
This thesis provides a new pipeline for the patient-specific humerus distal part reconstruction with analysis of possible approaches and the current state of research in this area. The reconstruction implementation is available as an open-source module into the 3D Slicer application, which can also be used in other cases in reconstruction missing morphology. Based on the testing, in ideal conditions the reconstructed modules appear to be of good quality. On the other hand, in other cases it is desirable to further improve the reconstruction results.

Fortunately, there's still a lot of room for improvement. One of the possible enhancement could be in form of a hyperparameter search, i.e. test many different parameter settings of the module and empirically choose the best one. Next, the rigid alignment could be calculated by utilizing important anatomical landmarks rather than automatic inference of fitness. Additionally, the process of selection of the base model for the statistical mean model creation can be optimized (as noted in section 3.3.2). In terms of the length preservation issue, possible solutions include the use of deformable statistical shape models and/or constraining deformations in the distal target mesh.

DeCA Module Installation

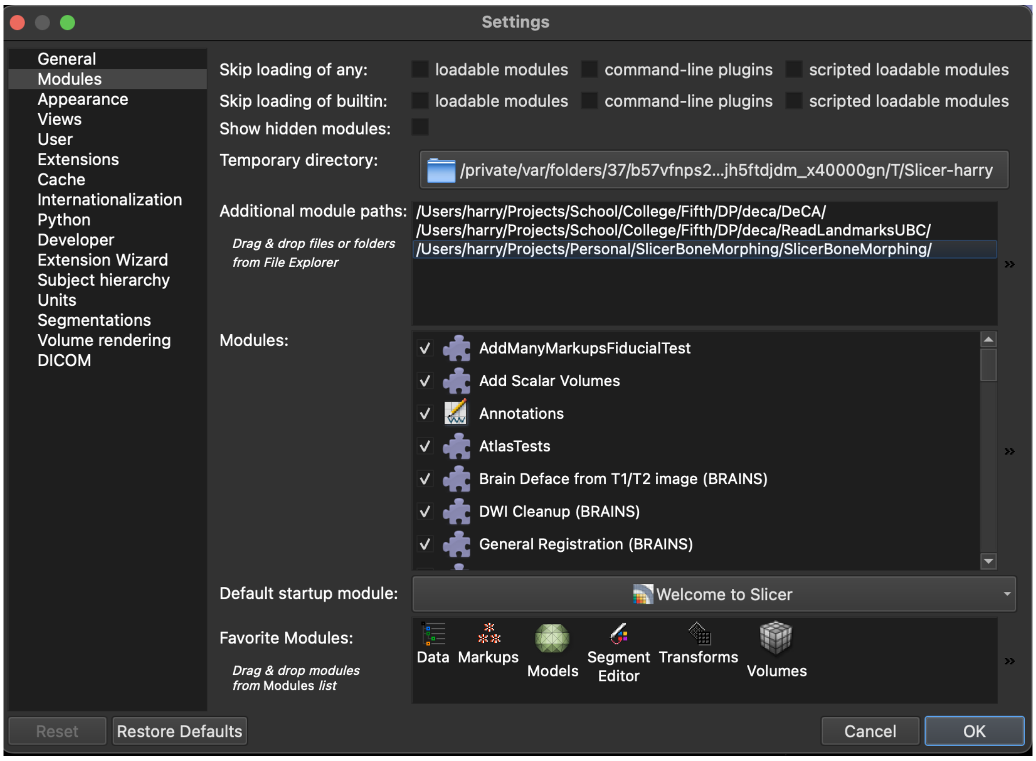
A

Installing the DeCA module into Slicer is a very straight forward process. Firstly clone the GitHub repository wherever you want to on your disk and navigate to it. In the cloned repo, there will be two main folders: **DeCA** and **ReadLandmarksUBC** (Figure A.1). We will be using these two folders shortly so do not close the file viewer yet. Next, open up Slicer, navigate to the **Edit** menu→**Application Settings**. In the settings open up the **Modules** tab and pay attention to the **Additional module paths** section. Now drag & drop the mentioned two folders into this window. The "Additional module paths" section should now contain both of the absolute paths to the DeCA module (Figure A.2). Now restart Slicer and if the installation is successful, you should now have two new modules available in the Modules overview in Slicer (Figure A.3).



DeCA	Add helper function for semi-landmarks	6 months ago
ReadLandmarksUBC	Module and icon update	4 years ago
.gitignore	Initial commit	4 years ago
CMakeLists.txt	Extension structure	4 years ago
DeCA.png	Module and icon update	4 years ago
Picture1.png	Module and icon update	4 years ago
README.md	Update README.md	10 months ago

Figure A.1: DeCA module file structure


Settings

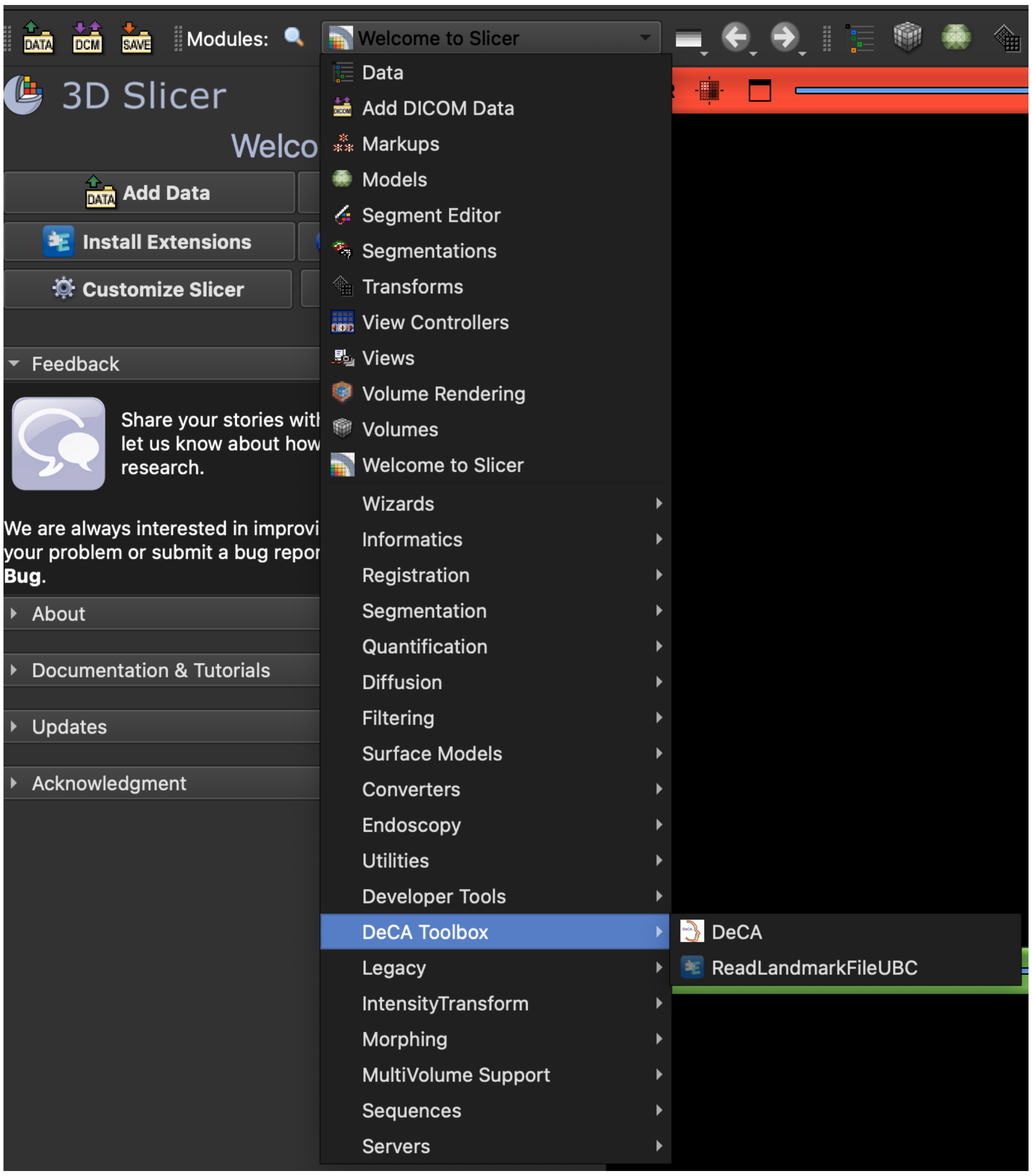
General
Modules
Appearance
Views
User
Extensions
Cache
Internationalization
Python
Developer
Extension Wizard
Subject hierarchy
Units
Segmentations
Volume rendering
DICOM

Skip loading of any: ☐ loadable modules ☐ command-line plugins ☐ scripted loadable modules
Skip loading of builtin: ☐ loadable modules ☐ command-line plugins ☐ scripted loadable modules
Show hidden modules: ☐
Temporary directory: /private/var/folders/37/b57vfnps2...jh5ftdjdm_x40000gn/T/Slicer-harry
Additional module paths:
/Users/harry/Projects/School/College/Fifth/DP/deca/DeCA/
/Users/harry/Projects/School/College/Fifth/DP/deca/ReadLandmarksUBC/
/Users/harry/Projects/Personal/SlicerBoneMorphing/SlicerBoneMorphing/
Drag & drop files or folders from File Explorer

Modules:
✓ AddManyMarkupsFiducialTest
✓ Add Scalar Volumes
✓ Annotations
✓ AtlasTests
✓ Brain Deface from T1/T2 image (BRAINS)
✓ DWI Cleanup (BRAINS)
✓ General Registration (BRAINS)

Default startup module: Welcome to Slicer
Favorite Modules: Data Markups Models Segment Editor Transforms Volumes
Drag & drop modules from Modules list

Figure A.2: Slicer Additional module paths section



Share your stories wit
let us know about how
research.

We are always interested in improvi
your problem or submit a bug repor
Bug.

Figure A.3: Slicer module list with DeCA installed

SlicerBoneMorphing User Manual

B

B.1 Installation

Supported platforms

- Linux (x86_64)
- Windows (x86_64)
- MacOS (both x86_64 and ARM; Slicer runs through Rosetta on ARM-based Macs)

Steps:

1. Download the latest ZIP package from GitHub Releases
2. Extract the ZIP contents to your desired folder
3. Open up 3D Slicer and go to Edit→Application Settings
4. In the **modules** section, add the extracted contents' path to "Additional Module Paths"
5. Restart 3D Slicer

DISCLAIMER! After restarting, the installation process will begin. If there are any Python modules not available in Slicer, they will be installed, so the startup will take SIGNIFICANTLY MORE amount of time. Do not be scared, this is intended behaviour.

B.2 Module Description

This section will describe the basics of the module for the user.

B.2.1 Input Section

This section is self-explanatory. Here, you can choose two input models:

- Source = the mean model, i.e. a full humerus
- Target = partial model to be reconstructed

B.2.2 Preprocessing

Before the generation process, we want to preprocess the model. First of all is the option of downsampling. For this you can configure the threshold for downsampling by the following parameter:

- **Downsampling distance threshold**
  - If set to 0.0, no downsampling is performed

After the downsampling, we compute the normals of the point clouds. The computation needs a radius for which the normals are calculated and maximum number of neighbours. These can be adjusted with the following parameters:

- **Normals estimation radius** - maximum radius in which points are considered neighbouring
- **Normals estimation max neighbours** - maximum number of neighbours taken into account

Also, we need to calculate a (Fast) point feature histogram in order to encode the local geometric properties of the models. This method uses the following parameters:

- **FPFH search radius** - maximum radius in which points are considered neighbouring
- **FPFH max neighbours** - maximum number of neighbours taken into account

B.2.2.1 Registration

At this moment we have our models preprocessed and ready for the next step, which is the registration. Here we calculate the rigid alignment of the models in order to pre-align them. The concrete method we use is called **RANSAC** (random sample consensus). The behaviour of this algorithm can be adjusted by the following parameters:

- **Max iterations**
- **Distance threshold** - maximum distance in which corresponding points are considered reachable
- **Fitness threshold** - the lowest fitness of the models to be accepted (section 3.4.5.2)

The computed fit by the RANSAC algorithm is a bit "raw". To improve it further, we perform the **ICP** (Iterative closest points) algorithm. This can be tuned by the following parameter:

- **ICP Distance threshold** - maximum distance in which points are considered neighbouring

B.2.3 Reconstruction

Since we now have a preprocessed meshes and with defined transformations from the source to the target, we can proceed to the reconstruction section. For the reconstruction we use the **BCPD** (Bayesian coherent point drift) algorithm. Now, the BCPD allows for very fine adjustments of its behaviour using lots of different parameters. For the exact description of their effects, please refer to the official documentation here.

You do NOT have to perform any kind of installation process, the BCPD and its geodesic variant are already pre-built and preconfigured for immediate use in this module.

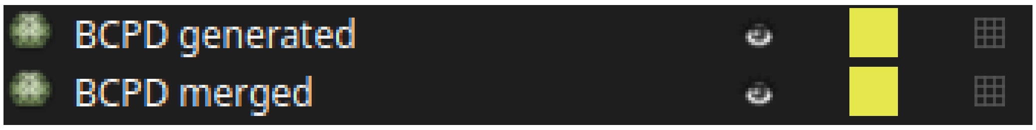
Not implemented options:

- Terminal output
- File output

B.2.4 Postprocessing

After the model is reconstructed, we include a postprocessing section to slightly modify the result, if necessary. For these, we let you modify the following parameters:

- **Clustering scaling**
  - Scaled size of voxel for within vertices that are clustered together
  - If set to 1.0, no scaling is applied



BCPD generated

BCPD merged

Figure B.1: Slicer scene model list with the imported reconstructed model

- **Smoothing iterations** - Number of iterations of mesh smoothing
  - If set to 0, no smoothing is applied

After the whole process is done, both the generated mesh (source transformed into target, standalone) and the merged mesh (generated meshes merged with the target; "combined model") are import back into the current Slicer scene (Figure B.1).

Mesh Cropping Script C

Source code C.1: Code snippet for mesh cropping

```python
def crop(mesh_filename: str, gender: str):
    mesh = o3d.io.read_triangle_mesh(os.path.join(
   MESH_SOURCE_DIR, gender, mesh_filename))

    # Compute the oriented bounding box
    obb = mesh.get_oriented_bounding_box()

    # Determine the major axis and its extent
    extents = np.asarray(obb.extent)
    major_axis_index = np.argmax(extents)
    major_axis_extent = extents[major_axis_index]

    # Calculate the extent for the cropping volume (one-third
    of the major axis)
    cropped_extent = extents.copy()
    cropped_extent[major_axis_index] /= 3

    # Compute the center for the proximal cropping volume
    # Inverting the direction of the adjustment to align with
    the correct proximal end
    cropped_center = obb.center + obb.R[:, major_axis_index]
   * (major_axis_extent / 3)

    # Create the cropping volume (a smaller OBB for the
   proximal third)
    cropped_obb = o3d.geometry.OrientedBoundingBox(
   cropped_center, obb.R, cropped_extent)
    proximal_mesh = mesh.crop(cropped_obb)
    proximal_mesh.compute_vertex_normals()

    o3d.visualization.draw_geometries([proximal_mesh])
    o3d.io.write_triangle_mesh(mesh=proximal_mesh, filename=
   os.path.join(MESH_OUTPUT_DIR, gender, mesh_filename))
```

Mesh Cropping and Closing Script

D

Source code D.1: Code snippet for mesh cropping and closing

```python
def crop(mesh_filename: str, gender: str):
    mesh = o3d.io.read_triangle_mesh(os.path.join(MESH_SOURCE_DIR, gender, mesh_filename))

    # Compute the oriented bounding box
    obb = mesh.get_oriented_bounding_box()

    # Determine the major axis and its extent
    extents = np.asarray(obb.extent)
    major_axis_index = np.argmax(extents)
    major_axis_extent = extents[major_axis_index]

    # Calculate the extent for the cropping volume (one-third of the major axis)
    cropped_extent = extents.copy()
    cropped_extent[major_axis_index] /= 3

    # Compute the center for the proximal cropping volume
    # Inverting the direction of the adjustment to align with the correct proximal end
    cropped_center = obb.center + obb.R[:, major_axis_index] * (major_axis_extent / 3)

    # Create the cropping volume (a smaller OBB for the proximal third)
    cropped_obb = o3d.geometry.OrientedBoundingBox(cropped_center, obb.R, cropped_extent)

    # Crop the mesh to get the proximal third
    proximal_mesh = mesh.crop(cropped_obb)

    pcd2 = o3d.geometry.PointCloud()
    pcd2.points = proximal_mesh.vertices
    pcd2.colors = proximal_mesh.vertex_colors
    pcd2.normals = proximal_mesh.vertex_normals

    proximal_mesh = o3d.geometry.TriangleMesh.
create_from_point_cloud_poisson(pcd=pcd2, depth=6, width
=1, linear_fit=False)[0]

    proximal_mesh.compute_vertex_normals()

    o3d.visualization.draw_geometries([proximal_mesh])
    o3d.io.write_triangle_mesh(mesh=proximal_mesh, filename=
os.path.join(MESH_OUTPUT_DIR, gender, mesh_filename))
```

Bibliography

[Abl+18] ABLER, Daniel et al. A statistical shape model to predict the premorbid glenoid cavity. *Journal of shoulder and elbow surgery*. 2018, vol. 27, no. 10, pp. 1800–1808. Available from DOI: 10.1016/j.jse.2018.04.023.

[Ali+21] ALIAJ, Klevis; HENNINGER, Heath B.; SULKAR, Hema; KOLZ, Christopher. *Biplane fluoroscopy derived humerus and scapula kinematics during arm elevation and rotation*. 2021. Available from DOI: 10.5281/zenodo.4536684.

[Amb+19] AMBELLAN, Felix; LAMECKER, Hans; VON TYCOWICZ, Christoph; ZACHOW, Stefan. *Advances in Experimental Medicine and Biology*. Statistical Shape Models: Understanding and Mastering Variation in Anatomy. 2019. Available from DOI: 10.1007/978-3-030-19385-0\\{_},5=.

[Bei+15] BEICHEL, Reinhard R et al. *Data From QIN-HEADNECK*. The Cancer Imaging Archive, 2015. Available from DOI: 10.7937/K9/TCIA.2015.K0F5CGLI.

[Bes92] BESL P.J., McKay D. Neil. A method for registration of 3-D shapes. 1992. Available also from: https://ieeexplore.ieee.org/document/121791.

[BB] BIOMEDICAL IMAGING, National Institue of; BIOENGINEERING. *Magnetic Resonance Imaging (MRI)*. Available also from: https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri.

[Bor23] BOROWY, Christopher S. *Sonography physical principles and instrumentation*. 2023-03-20. Available also from: https://www.ncbi.nlm.nih.gov/books/NBK567710/.

[Cla+13] CLARK, Kenneth W. et al. The Cancer Imaging Archive (TCIA): Maintaining and operating a public information repository. *Journal of digital imaging*. 2013, vol. 26, no. 6, pp. 1045–1057. Available from DOI: 10.1007/s10278-013-9622-7.

[D20] D, Walter Ledermann. Leyendo a Spallanzani hoy en día. *Revista Chilena De Infectologia*. 2020, vol. 37, no. 1, pp. 64–68. Available from DOI: `10.4067/s0716-10182020000100064`.

[Dec+21] DECKERS, Claudia et al. Midterm MRI Follow-Up of untreated enchondroma and atypical cartilaginous tumors in the long bones. *Cancers*. 2021, vol. 13, no. 16, p. 4093. Available from DOI: `10.3390/cancers13164093`.

[FLJ21] F PAULO, Soraia; LOPES, Daniel; JORGE, Joaquim. 3D Reconstruction from CT Images Using Free Software Tools. In: 2021, pp. 135–157. ISBN 978-3-030-61904-6. Available from DOI: `10.1007/978-3-030-61905-3_8`.

[Fed+12] FEDOROV, Andriy et al. 3D Slicer as an image computing platform for the Quantitative Imaging Network. *Magnetic resonance imaging*. 2012, vol. 30, no. 9, pp. 1323–1341. Available from DOI: `10.1016/j.mri.2012.05.001`.

[Fed+16] FEDOROV, Andriy et al. DICOM for quantitative imaging biomarker development: a standards based approach to sharing clinical data and structured PET/CT analysis results in head and neck cancer research. *PeerJ*. 2016, vol. 4, e2057. Available from DOI: `10.7717/peerj.2057`.

[FB81] FISCHLER, Martin A.; BOLLES, Robert C. Random sample consensus. *Communications of the ACM*. 1981, vol. 24, no. 6, pp. 381–395. Available from DOI: `10.1145/358669.358692`.

[FA] FOOD, U.S.; ADMINISTRATION, Drug. *Radiography*. Available also from: `https://www.fda.gov/radiation-emitting-products/medical-x-ray-imaging/radiography`.

[Har24] HARRIS, Tom. *How x-rays work*. 2024. Available also from: `https://science.howstuffworks.com/x-ray.htm`.

[HH13] HAYNES, Heather; HOLMES, William M. *Geomorphological Techniques (Online Edition)*. The emerging use of Magnetic Resonance Imaging (MRI) for 3D analysis of sediment structures and internal flow processes. British Society for Geomorphology, 2013. Geomorphological Techniques (Online Edition). Chap. 1, Sec. 5.4 - Imaging sediment structures.

[Her23] HERMENA, Shady. CT-Scan image production procedures. 2023. Available also from: `https://www.ncbi.nlm.nih.gov/books/NBK574548/`.

[Hir21] HIROSE, Osamu. A Bayesian formulation of coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*. 2021, vol. 43, no. 7, pp. 2269–2286. Available from DOI: `10.1109/tpami.2020.2971687`.

[Hir22] HIROSE, Osamu. Geodesic-Based Bayesian coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*. 2022, pp. 1–18. Available from DOI: 10.1109/tpami.2022.3214191.

[Hua+21] HUANG, Yichen; ROBINSON, Dale L.; PITOCCHI, Jonathan; LEE, Peter Vee Sin; ACKLAND, David C. Glenohumeral joint reconstruction using statistical shape modeling. *Biomechanics and modeling in mechanobiology (Internet)*. 2021, vol. 21, no. 1, pp. 249–259. Available from DOI: 10.1007/s10237-021-01533-6.

[Jol86] JOLLIFFE, Ian T. *Springer series in statistics*. Principal component analysis and factor analysis. 1986. Available from DOI: 10.1007/978-1-4757-1904-8_7.

[JAF18] JU, Zhang; ACKLAND, David C.; FERNANDEZ, Justin. Point-cloud registration using adaptive radial basis functions. *Computer methods in biomechanics and biomedical engineering*. 2018, vol. 21, no. 7, pp. 498–502. Available from DOI: 10.1080/10255842.2018.1484914.

[LC87] LORENSEN, William E.; CLINE, H. E. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*. 1987, vol. 21, no. 4, pp. 163–169. Available from DOI: 10.1145/37402.37422.

[Lüt+16] LÜTHI, Marcel; JUD, Christoph; GERIG, Thomas; VETTER, Thomas. *Gaussian Process Morphable Models*. 2016. Available also from: https://arxiv.org/abs/1603.07254.

[Mel14] MELTON, J. Trends in Fracture Incidence: A Population-Based Study Over 20 Years. 2014, pp. 582–585. Available also from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3929546/#S6title.

[Myr10] MYRONENKO A., Song X. Point Set Registration: Coherent Point Drift. 2010. Available also from: https://ieeexplore.ieee.org/document/5432191.

[Pat23] PATEL, Maulik. Neonatal distal humeral fracture - ultrasound. *Radiopaedia.org*. 2023. Available from DOI: 10.53347/rid-159640.

[Pol+17] POLTARETSKYI, Sergii et al. Prediction of the pre-morbid 3D anatomy of the proximal humerus based on statistical shape modelling. *The Bone & Joint journal*. 2017, vol. 99-B, no. 7, pp. 927–933. Available from DOI: 10.1302/0301-620x.99b7.bjj-2017-0014.

[Pub21] PUBLISHING, Harvard Health. *Radiation risk from medical imaging*. 2021-09. Available also from: https://www.health.harvard.edu/cancer/radiation-risk-from-medical-imaging.

[Rad86] RADON, Johann. On the determination of functions from their integral values along certain manifolds. *IEEE Transactions on Medical Imaging*. 1986, vol. 5, no. 4, pp. 170–176. Available from DOI: `10.1109/TMI.1986.4307775`.

[RM23] ROLFE, Sara; MAGA, A. Murat. *Lecture Notes in Computer Science*. DECA: A Dense Correspondence Analysis Toolkit for shape analysis. 2023. Available from DOI: `10.1007/978-3-031-46914-5\_,21=`.

[Rol+21] ROLFE, Sara et al. SlicerMorph: An open and extensible platform to retrieve, visualize and analyse 3D morphology. *Methods in ecology and evolution*. 2021, vol. 12, no. 10, pp. 1816–1825. Available from DOI: `10.1111/2041-210x.13669`.

[RBB09] RUSU, Radu Bogdan; BLODOW, Nico; BEETZ, Michael. Fast Point Feature Histograms (FPFH) for 3D registration. *IEEE*. 2009. Available from DOI: `10.1109/robot.2009.5152473`.

[Rus+08] RUSU, Radu Bogdan; MÁRTON, Zoltán-Csaba; BLODOW, Nico; BEETZ, Michael. Learning informative point classes for the acquisition of object model maps. *IEEE*. 2008. Available from DOI: `10.1109/icarcv.2008.4795593`.

[Tau95] TAUBIN, G. Curve and surface smoothing without shrinkage. 1995, pp. 852–857. Available from DOI: `10.1109/ICCV.1995.466848`.

[Tib96] TIBSHIRANI, Robert. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B. Methodological*. 1996, vol. 58, no. 1, pp. 267–288. Available from DOI: `10.1111/j.2517-6161.1996.tb02080.x`.

[Ume91] UMEYAMA, S. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1991, vol. 13, no. 4, pp. 376–380. Available from DOI: `10.1109/34.88573`.

[Vla+18] VLACHOPOULOS, Lazaros et al. Restoration of the Patient-Specific anatomy of the proximal and distal parts of the humerus. *The Journal of bone and joint surgery. American volume*. 2018, vol. 100, no. 8, e50. Available from DOI: `10.2106/jbjs.17.00829`.

[Wei+20] WEI, Hang et al. 2-Step Sparse-View CT Reconstruction with a Domain-Specific Perceptual Network. *arXiv (Cornell University)*. 2020. Available from DOI: `10.48550/arxiv.2012.04743`.

[Xie04] XIE Z, Farin GE. Image registration using hierarchical B-splines. 2004. Available also from: `https://ieeexplore.ieee.org/document/1260760`.

[Zha+22] ZHANG, Charles; PORTO, Arthur; ROLFE, Sara; KOCATULUM, Altan; MAGA, A. Murat. Automated landmarking via multiple templates. *PloS one*. 2022, vol. 17, no. 12, e0278035. Available from DOI: `10.1371/journal.pone.0278035`.

[Zha94] ZHANG, Zhengyou. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*. 1994, vol. 13, no. 2, pp. 119–152. Available from DOI: `10.1007/bf01427149`.

[ZH] ZHUO WANG, Ming-Yue Lv; HUANG, Yao-Xiong. Effects of Low-Dose X-Ray on Cell Growth, Membrane Permeability, DNA Damage and Gene Transfer Efficiency. [N.d.]. Available also from: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7597563/`.

List of Figures

List of Tables

List of Listings