2017-12-18

Full-body Shell Creation for CAD Virtual Humans including Tightly-Spaced, Enclosed Shells

Aung Thu Htet
Worcester Polytechnic Institute

Full-body Shell Creation for CAD Virtual Humans including Tightly-Spaced, Enclosed Shells

by

Aung Thu Htet

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Master of Science

In

Electrical and Computer Engineering

December 2017

APPROVED:

Prof. Sergey Makarov _____
Worcester Polytechnic Institute

Dr. Gregory Noetscher _____
US Army Natick Soldier Research, Development, and Engineering Center

Dr. Louis Chen _____
Bose Corporation

Dr. Janakinadh Yanamadala _____
MathWorks, Inc.

To my family and friends

Abstract

Computational human models have become essential in several different biomedical and electrical engineering research areas. They enable scientists to study, model, and solve complex problems of human body responses to various external stimuli including electromagnetic and radio-frequency signals.

This study describes the algorithms and procedures of creating multi-tissue full-body Computer-Aided Design (CAD) human models. An emphasis is made on full-body shells of variable thicknesses, e.g. skin, fat, and average body container shells. Such shells, along with internal organs, are useful for multiple high- and low-frequency simulations in a variety of applications.

Along with the creation of full-body models, an automatic algorithm to selectively decimate the meshes based on average surface curvature is developed. The algorithm will significantly reduce model size while keeping the same interpolation accuracy.

I defend:

1. Creation of Tightly-Spaced Enclosed Full-body Shells of CAD Human Male Model Using software tools ITK-snap, MeshLab, and MATLAB
2. Creation of Full-body CAD Human Female Model with Ear Canals of Variable Thicknesses
3. Development of Selective Decimation Algorithm based on Average Surface Curvature in MATLAB

Acknowledgement

Many thanks to Professor Sergey Makarov for his continuous guidance throughout this work. His vast knowledge, valuable advice, and dedication have been crucial to the successful outcome of this study.

I would also like to sincerely thank Dr. Gregory Noetscher of US Army Natick Soldier Research, Development, and Engineering Center. I am grateful for his critical advice on several major elements of this work, especially development of selective decimation algorithm based on surface curvature.

I would also like to thank fellow students I have worked with during this study - Janakinadh Yanamadala, Anh Le Tran, Harshal Tankaria, Edward Burnham, Nicholas Maino, and Patrick Lacroix. I had a chance to learn much from them including MATLAB, SpaceClaim, among various other tools and methodologies.

Table of Contents

List of Figures

List of Tables

List of Published and Accepted Papers

1. G. M. Noetscher, A. T. Htet, N. Maino, P. Lacroix, "The Visible Human Project Male CAD Based Computational Phantom and its use in Electromagnetic Simulations," 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, July. 11-15, 2017.

2. J. Yanamadala, G. Noetscher, M. Piazza, A. Helderman, N. Thang, T. Dolma, T. Trinh, J. Zhang, M. Islam, S. Xie, V. Rathi, S. Maliye, H. Win, A. Tran, X. Jackson, P. Carberry, A. Htet, M. Kozlov, S. Louie, A. Nazarian, S. Makarov "VHP-Female v. 2.0 Full-Body Computational Phantom: ANSYS HFSS Performance Metrics in Application to Antenna Radiation and Scattering," 39th Antenna Applications Symposium, Monticello, IL, Sep. 22- 24, 2015, pp. 453-461

3. J. Yanamadala, G. M. Noetscher, V. K. Rathi, S. Maliye, H. A. Win, A. L. Tran, X. J. Jackson, A. T. Htet, M. Kozlov, A. Nazarian, S. Louie, S. N. Makarov, "New VHP-Female v. 2.0 Full-Body Computational Phantom and Its Performance Metrics Using FEM Simulator ANSYS HFSS," 37th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society, Milano, Italy, Aug. 25-29, 2015.

4. J. Yanamadala, J. Elloian, G. M. Noetscher, A. T. Htet, S. N. Makarov, F. Scire-Scappuzzo, A. Pascual-Leone, "FEM-Compatible visible human female segmentation model and its application to in-body and on-body antenna studies," 37th Antenna Applications Symposium, Monticello, IL, Sep. 17-19, 2013.

5. G. M. Noetscher, A. T. Htet, X. Dai, S. N. Makarov, "Library of MATLAB modules for solving and visualizing electromagnetics problems," 2013 ASEE Northeast Section Conference, Norwich University, Mar. 14-16, 2013

6. G. M. Noetscher, A. T. Htet, J. M. Elloian, S. N. Makarov, F. Scire-Scappuzzo, A. PascualLeone, "Detecting in vivo changes of electrical properties of cerebrospinal fluid using microwave signals from small coil antennas – numerical simulations," IEEE Signal Processing in Medicine and Biology Symposium, Dec. 1st, 2012.

CHAPTER 1

OVERVIEW OF FULL-BODY COMPUTATIONAL CAD MODELS

1.1. Introduction

Virtual humans are computational models that are useful in numerical simulations of how the human body responds to external stimuli. According to the National Institutes of Health, "modeling can expedite research by allowing scientists to conduct thousands of simulated experiments by computer in order to identify the actual physical experiments that are most likely to help the researcher find the solution to the problem being solved"[1]. In fact, human body modeling has become essential in several areas of study and markets such as development of medical devices, automotive safety, electromagnetics studies and radiology studies [2].

Computation and numerical simulations with the virtual human models require a combination of mathematics, physics, anatomy, physiology, and computer science [2]. Currently, simulations of human body responses are limited by several factors such as anatomical accuracy, computational efficiency, and cross-platform compatibility. In this work, tools and methodologies for creating anatomically correct, and numerically efficient virtual human body models are studied with a particular emphasis on the creation of full-body models.

1.2. Overview of Full-Body Human Models

Currently available human models are listed in the following table from IEEE [3], along with country, type of the model, resolution, etc.

Table 1. 1. Major Anatomical Full-Body Human Models for CEM and Radiological Simulations (After 2004) [3]

Entity/Country	Model Name	G/A/H/W	Da	TYPE	RES, mm^2	FV	D
IT'IS Found. Switzerland	Glenn	m/84/173/61.1	N	V	0.5x0.5x1.0h 0.9x0.9x2b	N	Y
IT'IS Found. Switzerland	FATS	m/37/182/119	N	V	0.5x0.5x1.0h 0.9x0.9x2b	N	Y
IT'IS Found. Switzerland	DUKE	m/34/177/70.3	N	V/S	0.5x0.5x1.0h 0.9x0.9x2b	Y	Y

IT'IS Found. Switzerland	ELLA	f/26/163/57.3	N	V/S	0.5x0.5x1.0h 0.9x0.9x2b	Y	Y
IT'IS Found. Switzerland	LOUIS	m/14/168/49.7	N	V	0.5x0.5x1.0h 0.9x0.9x2b	N	Y
IT'IS Found. Switzerland	BILLIE	f/11/149/34.0	N	V/S	0.5x0.5x1.0h 0.9x0.9x2b	Y	Y
IT'IS Found. Switzerland	EARTHA	f/8/136/29.9	N	V	0.5x0.5x1.0h 0.9x0.9x2b	N	Y
IT'IS Found. Switzerland	DIZZY	m/8/137/25.4	N	V	0.5x0.5x1.0h 0.9x0.9x2b	N	Y
IT'IS Found. Switzerland	THELONIOUS	m/6/115/18.6	N	V/S	0.5x0.5x1.0h 0.9x0.9x2b	Y	Y
IT'IS Found. Switzerland	ROBERTA	f/5/109/17.8	N	V	0.5x0.5x1.0h 0.9x0.9x2b	N	Y
IT'IS Found. Switzerland	NINA	f/3/92/13.9	N	V	0.5x0.5x1.0h 0.9x0.9x2b	N	N
IT'IS Found. Switzerland	CHARLIE	f/8w/na/4.3	N	V	0.5x0.5x1.0h 0.9x0.9x2b	N	N
China Acad. of Tel. Res.	CHINESE MALE	m/35/172/64	Y[1]	V/S	1x1x1	N	N
China Acad. of Tel. Res.	CHINESE FEMALE	f/22/162/54	Y[1]	V/S	1x1x1	N	N
Huazhong Univ, China	CDH M2	m/166	Y[2]	V/S	0.1x0.1x0.2	N	N
Huazhong Univ, China	CHINESE REF. MAN	m/166	Y	S	2x2x2	N	Y
Natl. Inst. of Inform. and Comm. Technol., Japan	NAGAOKA MAN (TARO)	m/22/173/65	N	V	2x2x2	N	N
Natl. Inst. of Inform. and Comm. Technol., Japan	NAGAOKA WOMAN	f/22/160/53	N	V	2x2x2	N	N
ETRI, Korea Hanyang University	KOREAN MAN	m/21/176/67	N	V/S	1x1x1 (head) 3x3x3 (body)	N	N
ETRI, Korea Hanyang University	KOREAN CHILD	m/7/122.4/25.5	N	V	1x1x3	N	N

ETRI, Korea Hanyang University	KOREAN WOMAN	f/26/161/54	N	V	2x2x2	N	N
Nat. Radiological Protection Board, UK	NAOMI	f/23/163/60	N	V	2x2x2	N	N
REMCOM, PennState	Male/Female	m/f	Y	V	5x5x5 (both)	N	Y
Helmholtz Zentrum Munchen, Germany CST AG, Germany	BABY	f/8w/57/4.2	N	V	0.85x0.85x4	N	N
	CHILD	f/7/115/21.7	N	V	1.54x1.54x8	N	N
	DONNA	f/40/176/79	N	V	1.88x1.88x10	N	N
	EMMA	f/26/170/81	N	V	0.98x0.98x10	N	N
	GUSTAV	m/38/176/69	N	V	2.1x2.1x8	N	N
	LAURA	f/43/163/51	N	V	1.88x1.88x5	N	N
	HUGO, posable	m/38/187/113	Y	V	1x1x1	N	Y
U Texas Austin, USA	AUSTIN MAN	m/38/187/113	Y	V	1x1x4	Y	Y
U Texas Austin, USA	AUSTIN WOMAN	f/60/162/88	Y	V	1x1x4	Y	Y
Duke University Medical Center, USA	XCAT FAMILY	Orig: f/60/162/88 m/38/187/113	Y	NURBS &S[4]	variable	Y	Y
NEVA EM LLC, WPI, USA	VHP-Female (NELLY)	f/60/162/88	Y	FEM	variable		
NEVA EM LLC, WPI, USA	VHP-MALE	m/38/180/90.3	Y	FEM	variable		

Abbreviations and comments:

G/A/H/W – Gender/Age/Height/Weight; Da – Original image dataset made available for independent evaluation (Y/N); TYPE (V – voxel; S – surface-based model, but without proven FEM meshability; FEM – surface-based FEM-meshable model); RES – Lowest image resolution (before or after post-processing) of the model declared by the provider (h=head, b=body); FV - Free version for available (Y/N); D – Deformable/posable (Y/N);

Among the models in the table, the *virtual population* from IT'IS Foundation Switzerland [4]-[6] is currently most comprehensive. Fig. 1 shows the major members of the family.

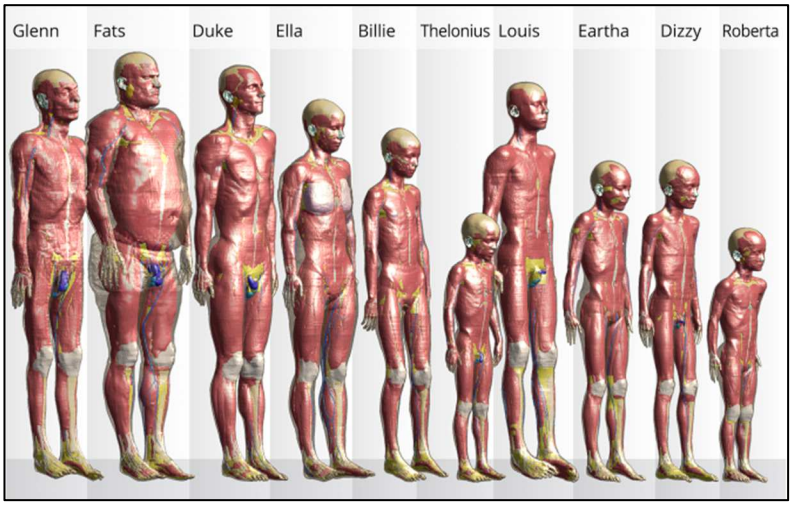

Figure 1: Members of Virtual Population [4]-[6]

A majority of the models in the table are voxel models or surface-based models, and are not compatible with FEM (Finite Element Method).

Other models such as male model from Zygote Media Group, Inc. shown in Fig.2 are available. However, these types of models are for educational purposes, illustrations or computer games and are not based on actual subjects [2].

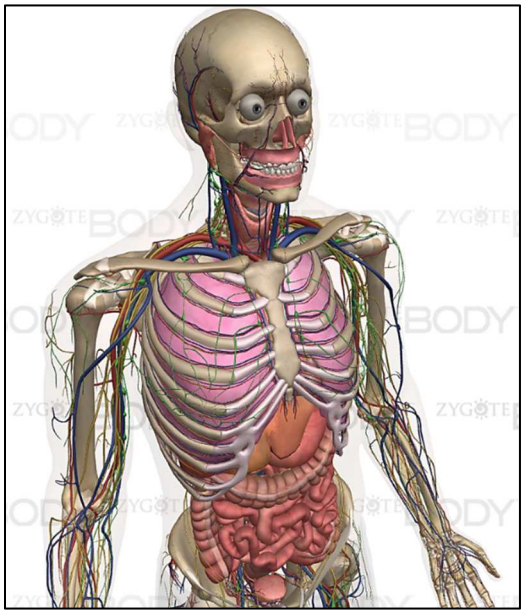

Figure 2: Full-body Male Model from Zygote Media Group, Inc. [7][7]

VHP-Female model, in contrast, are based on real subjects, anatomically accurate and compatible for simulations with Finite Element Method Fig. 3 shows the VHP-Female CAD model with several major tissues.

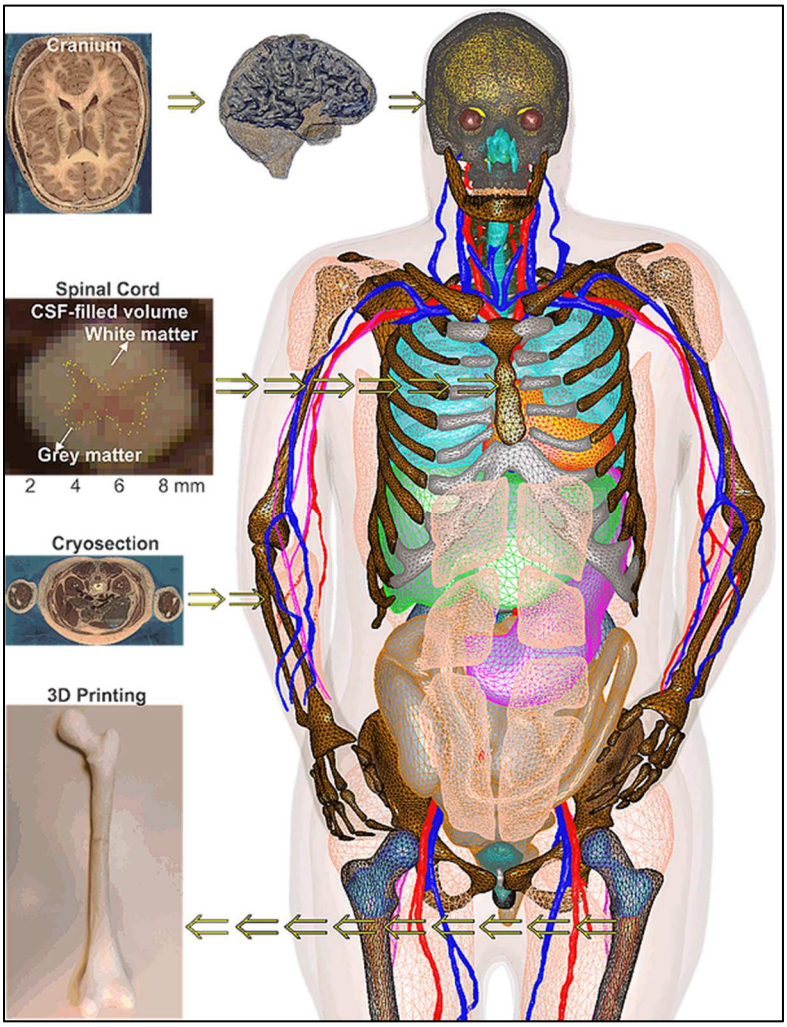

Figure 3: VHP-Female v. 3.0 CAD model [8]-[13] with about 250 individual parts; some tissues have been removed for visual clarity.[2]

References

[1]. National Institute of Biomedical Imaging and Bioengineering, NIH: "Computational Modeling," July 2013. Available: https://www.nibib.nih.gov/science-education/science-topics/computational-modeling

[2]. Makarov SN, Noetscher GM, Yanamadala J, Piazza MW, Louie S, Prokop A, Nazarian Ara, Nummenmaa A. Virtual Human Models for Electromagnetic Studies and Their Applications

[3]. IEEE Int. Committee on Electromagnetic Safety: Technical Committee 34: List of human phantoms. Available: http://grouper.ieee.org/groups/scc34/sc2/wg2/available%20human%20models.doc

[4]. A. Christ, W. Kainz, E. G. Hahn, K. Honegger, M. Zefferer, E. Neufeld, W. Rascher, R. Janka, W. Bautz, J. Chen, B. Kiefer, P. Schmitt, H. P. Hollenbach, J. Shen, M. Oberle, D. Szczerba, A. Kam, J. W. Guang, and N. Kuster, "The Virtual Family - development of surface-based anatomical models of two adults and two children for dosimetric simulations," Phys. Med. Biol., vol. 55, pp. 23-38, 2010.

[5]. M.-C. Gosselin, E. Neufeld, H. Moser, E. Huber, S. Farcito, L. Gerber, M. Jedensjo, I. Hilber, F.-D. Gennaro, and B. Llyod, "Development of a New Generation of High-Resolution Anatomical Models for Medical Device Evaluation: The Virtual Population 3.0," Phys. Med. and Biol., vol. 59, pp. 5287-5303, 2014.

[6]. The Virtual Population. High-Resolution Anatomical Models for Computational Life Sciences. SPEAG AG, Flyer, EuCAP 2016, Davos, Switzerland, April 10-15 2016, 2 p.

[7]. Zygote Media Group, Inc. ZygoteBody™. Online: https://zygotebody.com/about.

[8]. J. Yanamadala, V. K. Rathi, S. Maliye, H. A. Win, A. L. Tran, M. K. Kozlov, G. M. Noetscher, A. Nazarian, and S. N. Makarov, "Segmentation of the Visible Human Project® (VHP) Female Cryosection Images within MATLAB® Environment," 23rd Int. Meshing Roundtable (IMR23), London, England, Oct. 12-15, 2014. Available: http://www.imr.sandia.gov/papers/imr23.html

[9]. J. Yanamadala, G. M. Noetscher, V. K. Rathi, S. Maliye, H. A. Win, A. L. Tran, X. J. Jackson, A. T. Htet, M. Kozlov, A. Nazarian, S. Louie, and S. N. Makarov, "New VHP-Female V.2.0 Full-Body Computational Phantom and Its Performance Metrics Using FEM Simulator ANSYS HFSS," 37th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC 2015), Milano, Italy, Aug. 25-29, 2015, pp. 3237-3241.

[10]. G. M. Noetscher, J. Yanamadala, M. Kozlov, S. Louie, A. Nazarian, and S. Makarov, "VHP-Female v3.0 FEM/BEM Computational Human Phantom," 24th Int. Meshing Roundtable (IMR24), Austin, TX, Oct. 12-14, 2015.

[11]. J. Yanamadala, G. M. Noetscher, S. Louie, A. Prokop, M. Kozlov, A. Nazarian, and S. N. Makarov, "Multi-Purpose VHP-Female Version 3.0 Cross-Platform Computational Human

Model," 10th European Conf. on Antennas and Propagation (EuCAP16), Davos, Switzerland, April 2016, pp. 1-5.

[12]. H. Tankaria, X. J. Jackson, R. Borwankar, G. N. K. Srichandhru, A. L. Tran, J. Yanamadala, G. M. Noetscher, A. Nazarian, S. Louie, and S. N. Makarov, "VHP-Female Full-Body Human CAD Model for Cross-Platform FEM Simulations – Recent Development and Validations," 38th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC 2016), Orlando, FL, Aug. 16-20, 2016.

[13]. G. M. Noetscher, J. Yanamadala, S. Louie, A. Nazarian, and S. N. Makarov, "Computational Human Model VHP-Female Derived from Datasets of the National Library of Medicine," IEEE Pulse, April 27th 2016. Available: http://pulse.embs.org/march-2016/creating-a-computational-human-model/

CHAPTER 2

METHODS OF CAD MODEL CREATION

2.1 Introduction

The CAD models are created using the data made available by the Visible Human Project® (VHP) established in 1989 by the U.S. National Library of Medicine. The purpose of the project is to create a detailed three-dimensional representation of the male and female bodies.

Both male and female datasets contain stacks of cryosectional images acquired at the axial plane of the bodies. These axial cryosectional image stacks obtained from the Visible Human Project are segmented using manual or semi-automatic methods and using several different software tools such as Insight Toolkit segmentation software (ITK-snap), and MATLAB. The initial segmentation then goes through a series of mesh calibration procedures so that the final models are numerically feasible in simulation without compromising the anatomical accuracy.

The image segmentation procedures as well as mesh processing operations are described in this chapter.

2.2 Segmentation

For segmenting the model, the Insight Toolkit segmentation software (ITK-snap) is primarily used. In order to import the image stacks from Visible Human Project into the tool, the images are first converted into a compatible binary file using an image-processing tool ImageJ.

Once the dataset is imported, the segmentation process can be either semiautomatic or manual in the tool. However, in many instances, semiautomatic segmentation requires manual intervention. In order to segment a tissue or structure manually, the polygon tool available in the Insight Toolkit is used to trace the boundary of the desired structure. The definition of the mesh structure can be controlled by placing the desired number of points in the polygon. Usually, regions of higher curvature require a greater number of points in order to accurately represent the definition in the area.

The segmentation results in a coarse mesh, which can be exported into several file formats. The resulting meshes are usually exported into STL file format since the format is compatible in a variety of mesh processing software tools.

Fully manual segmentation requires a lot of time and effort whereas fully automatic segmentation is complex and error-prone. Semi-automatic segmentation guided by

manual segmentation proves to be an optimal solution for creating initial models. The following sub-sections provides a detailed description of manual segmentation and semi-automatic segmentation.

2.2.1. Manual Segmentation [1]

Computational phantoms or virtual humans are created via a set of 3D mathematical algorithms commonly called *image segmentation*. Segmentation is one of the most studied problems in the field of biomedical image analysis. Consider one body image (a slice in the xy-plane) in Fig. 9a which shows a cross-section of a human leg including the patella [2][2]. The complete stack of images continues in the z-direction. A skilled operator traces the patella boundary with a set of discrete points in the xy-plane (a polygon) shown by crosses in Fig 9a. A z-coordinate corresponding to the global coordinate system is added. Then, another cross-sectional image is traced, and all 3D points are collected image by image. The result is a complete patella boundary in three dimensions given in the form of a point cloud shown in Fig. 9b. This process is known as *manual segmentation*, still the "gold standard" of image segmentation. Other tissues are segmented similarly. The inner volume of the point cloud is either empty or can be filled with a set of (uniformly distributed) inner nodes. In the latter case, we arrive at a volumetric *voxel model* of a tissue, which is a *typical final result* of image segmentation. A large number of such volumes may exist, either interconnected or separate.

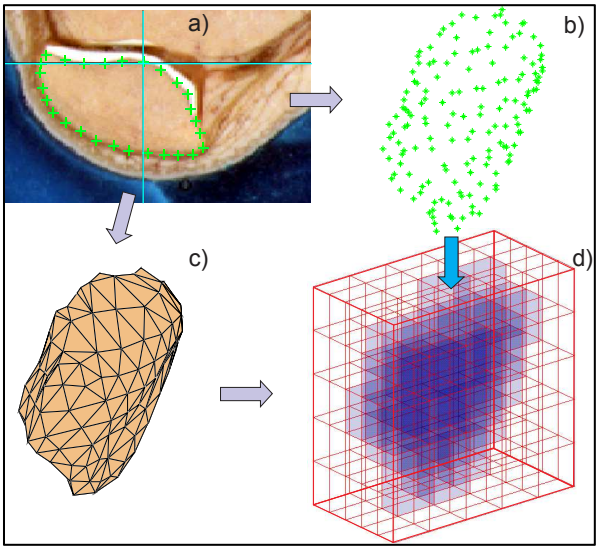

Figure 4: Image of a patella with a traced boundary; b) – resulting point cloud; c) – patella CAD model; d) – patella voxel model.[1]

2.2.2 Semi-Automatic Segmentation [1]

Manual segmentation of datasets is extremely labor intensive and expensive. Depending on the images and segmentation quality, this effort is often measured by many man-months or years for a single model [3]. Therefore, most research has focused on semi-automatic or fully automatic approaches. The basic segmentation algorithm is a threshold or pixel contrast method [4], which is similar to manual tracing of the boundaries of 2D object slices outlined above. A more elaborate example is an active contour segmentation technique [5]. For medical image segmentation review, please see [6]-[8]. There is a plethora of image segmentation software packages [9].

A popular and powerful open-source semi-automatic image segmentation tool is ITK-SNAP [11] from University of Pennsylvania, which includes a multi-modality segmentation capability and machine learning to differentiate tissue classes based on texture, location, and intensity.

2.2.3 Automatic Segmentation

One software package for automatic segmentation is FreeSurfer Software package developed at the Athinoula A. Martinos Center for Biomedical Imaging, Massachusetts General Hospital. It is an imaging software package specializing in segmentation and labeling of parts of brain. Currently available automatic segmentation of the software includes subcortical segmentation, and hippocampal subfields segmentation [12].

In surface-based segmentation of the white matter, the tool first strips the skull [14]. Voxels are classified as white matter based on their intensity. Cutting planes are also chosen to separate hemispheres and to separate out the white matter. The result is refined based on intensity gradients between white matter and gray matter. The following figure shows the skull-stripped brain (left), and the segmented white matter (right).[13]

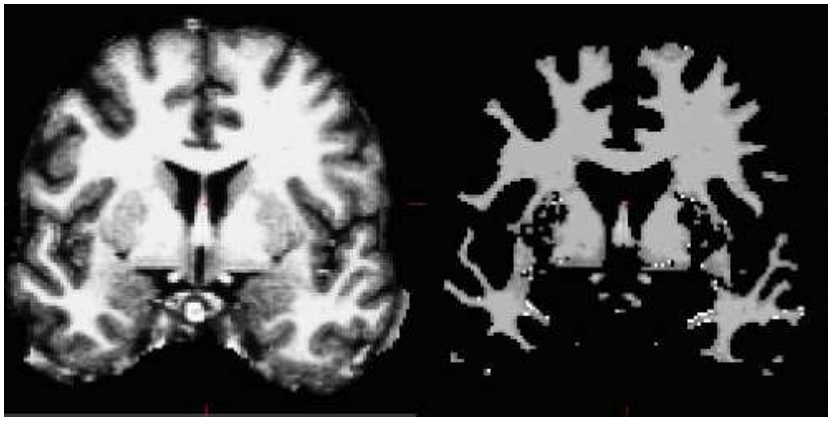

Figure 5: FreeSurfer Segmentation of White Matter[13]

The FreeSurfer also includes a visualization tool 'FreeView' that can load several volumes at once. The volumes are visualized at high resolution as in the following figure.

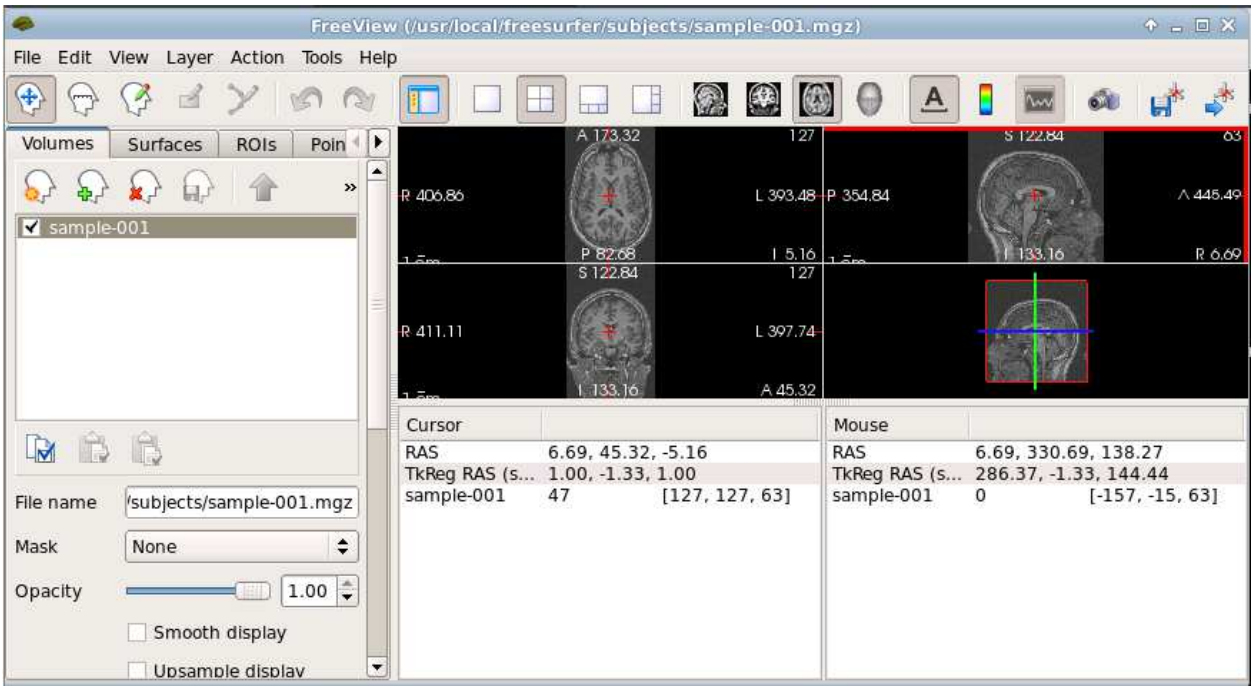

Figure 6: FreeSurfer FreeView Visualization Tool

Another software suite that can accomplish automatic segmentation is Slicer software program, developed by the MIT AI Lab and the Surgical Planning Lab at Brigham & Women's Hospital. It can achieve automatic segmentation of various brain structures using expectation-maximization algorithm [15]

2.2.4. SimNIBS[16]

SimNIBS, which stands for simulation of Non Invasive Brain Stimulation, is a software package that incorporates automatic segmentation of FreesSurfer and simulation capabilities using Finite Element Method libraries. SimNIBS includes a command line segmentation tool 'mri2mesh', which utilizes FreeSurfer to automatically segment MRI images and generates meshes of head tissues such as skin, CSF (Cerebrospinal Fluid), skull, white matter, and gray matter.

For example, the following command does automatic segmentation of subject 'example_model', running all steps of reconstruction, using head.nii MRI images.

```
mri2mesh  --all  example_model head.nii
```

The result is a .msh file that includes all the tissues, which can be used in simulation with SimNIBS.

```
example_model.msh
```

2.2.4.1. Electric Field Calculation Using SimNIBS

SimNIBS can be used to calculate electric field by either Transcranial Direct Current Simulation (tDCS) Transcranial Magnetic Simulation (TMS). This section describes the TMS electric field calculation using an example model from SimNIBS.

After downloading and installing SIMNIBS, SimNIBS GUI is invoked using the command:

```
simnibs_gui
```

In the GUI, the example mesh ‘almi5.msh’ is loaded.

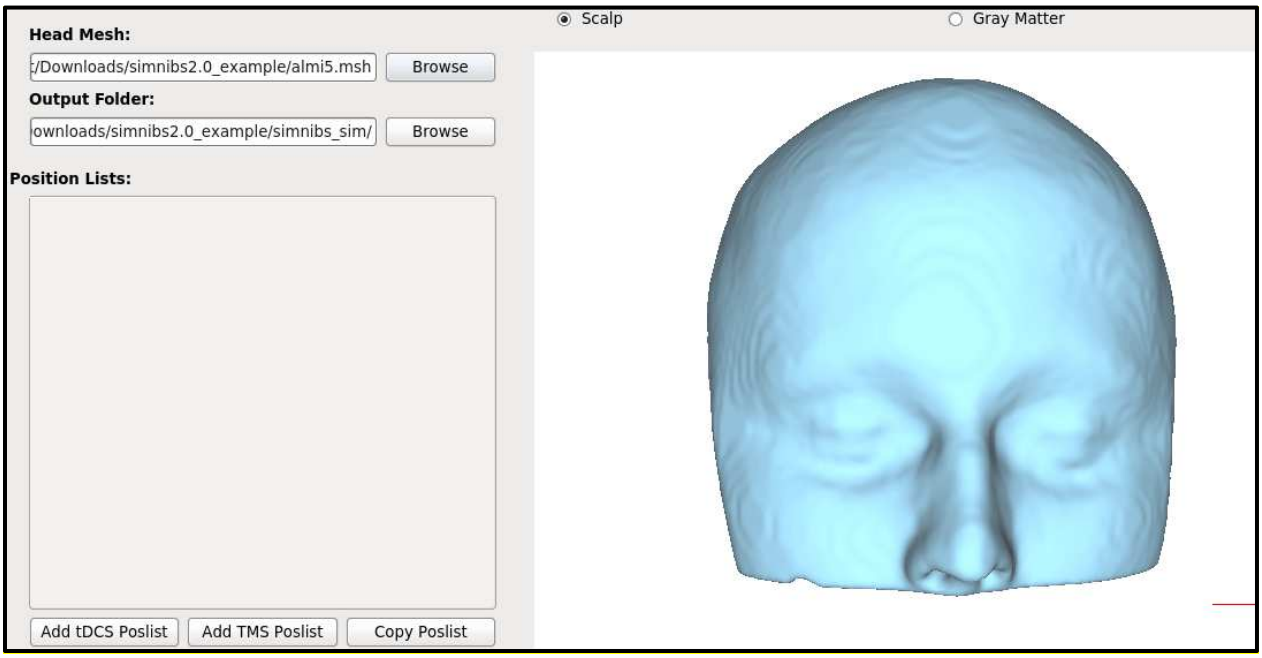

Figure 7: Example Mesh Loaded in SimNIBS GUI. Original Drawing using SimNIBS GUI

A TMS coil is added by specifying the position and direction reference points as in the following figure.

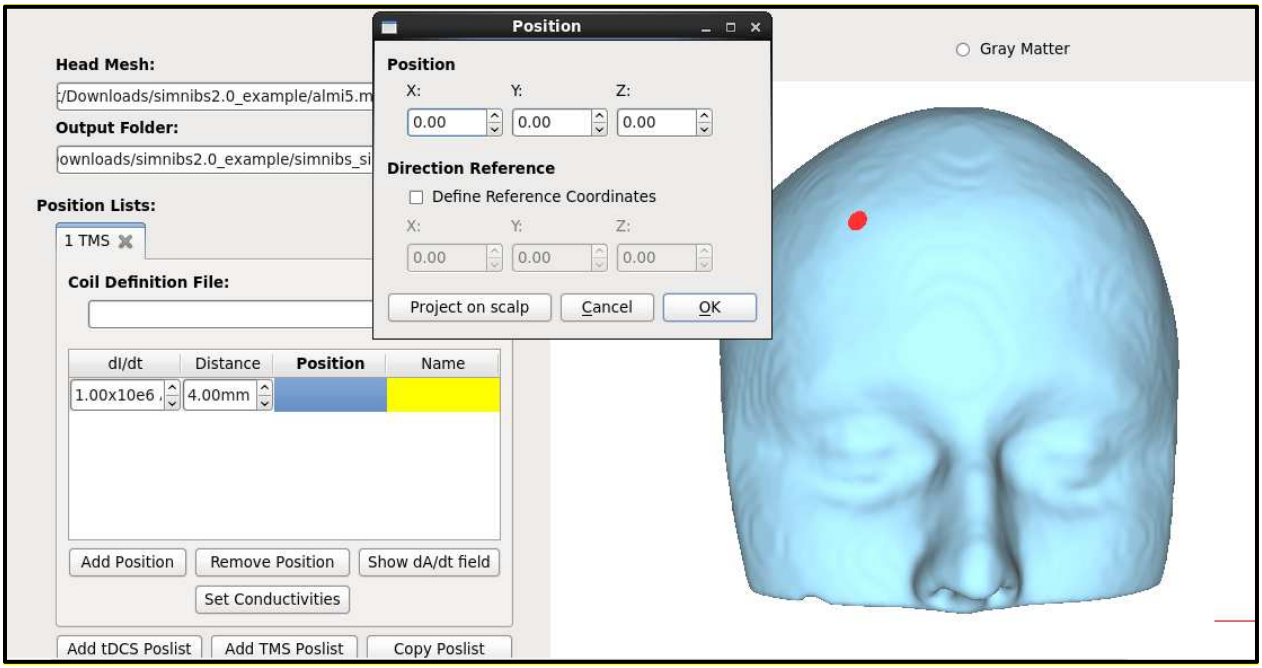

Figure 8: Adding TMS Position and Direction Reference to SimNIBS Model. Original Drawing using SimNIBS GUI

Next, a coil definition file is specified.

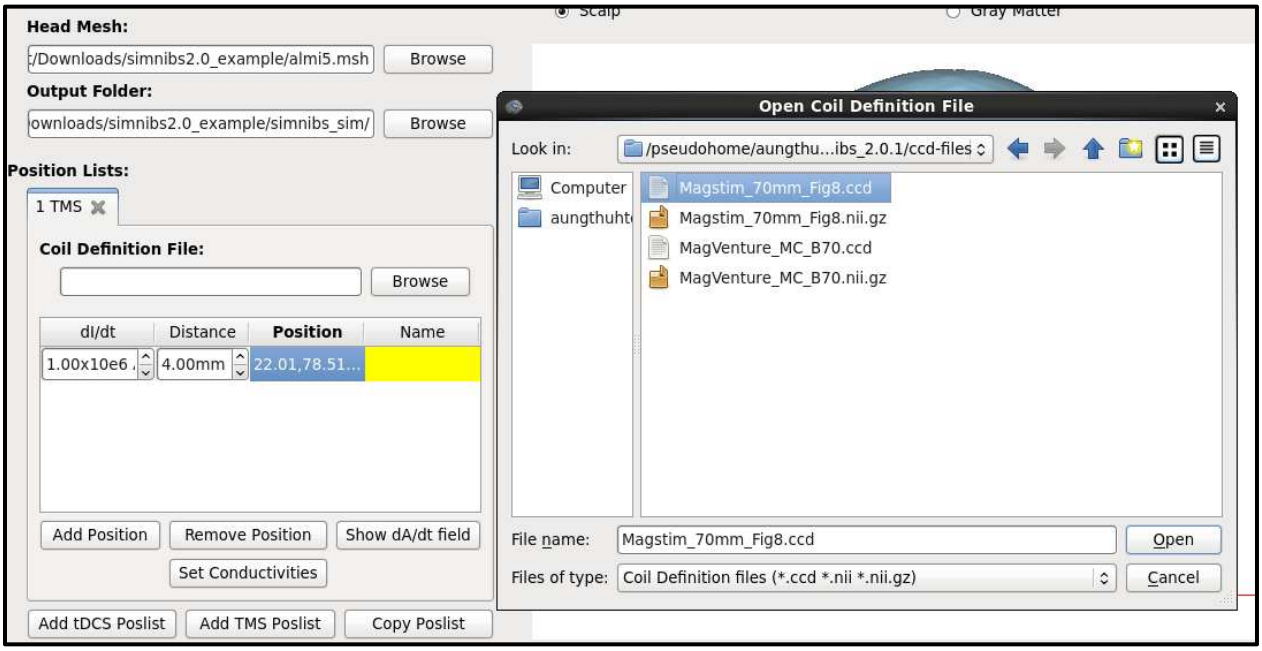

Figure 9: Adding Coil Definition File to SimNIBS. Original Drawing using SimNIBS GUI

Simulation is run and after the simulation finishes, a final .msh file is generated. The result msh file is viewed using gmsh tool included in the SimNIBS software package.

```
gmsh result.msh
```

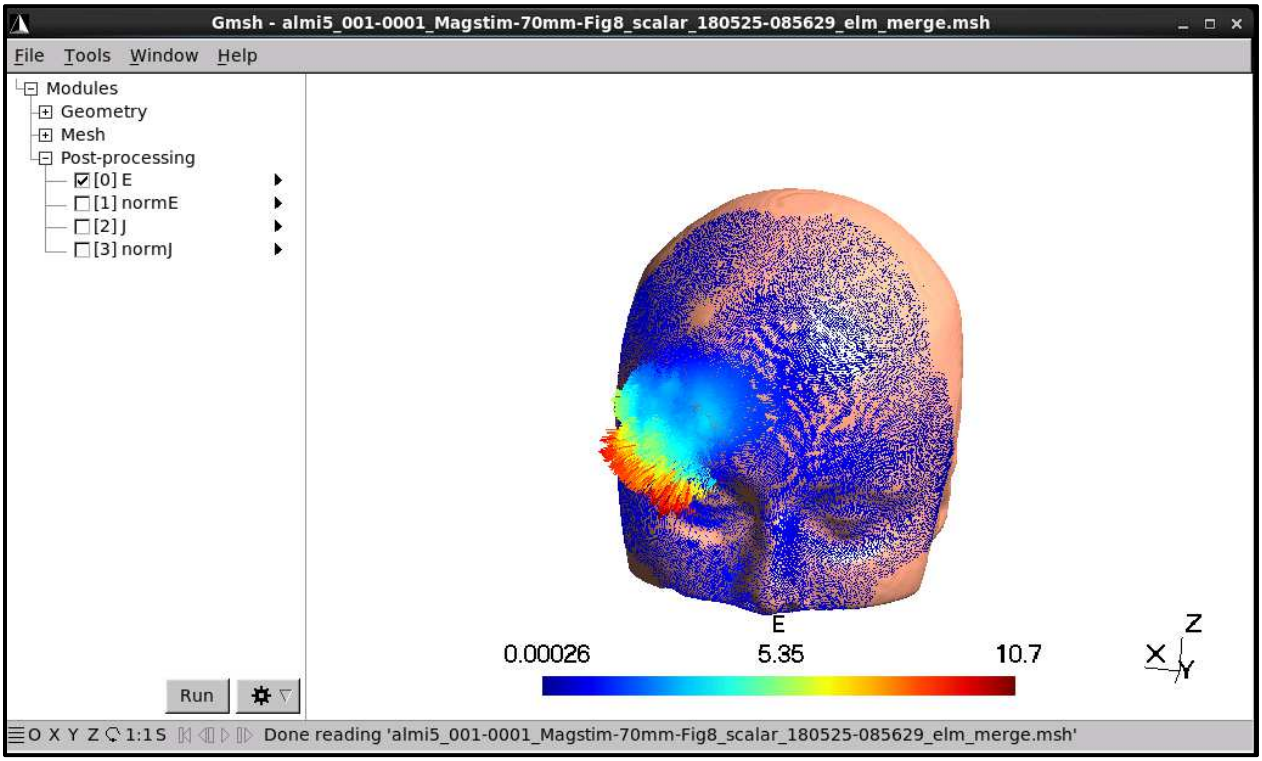

Figure 10: SimNIBS Simulation Result Viewed using Gmsh. Original Drawing using Gmsh Tool

2.2.4.2. Importing SimNIBS Simulation Data into MATLAB

The resulting electric field data can be exported as generic txt file and can be imported into MATLAB for further analysis.

The following picture shows the electric field data re-plotted in MATLAB. The head model is saved as mesh file, and the electric field data from SimNIBS is saved as txt file.

The head model is saved as stl mesh file as follows. The stl file is converted into MATLAB mat format.

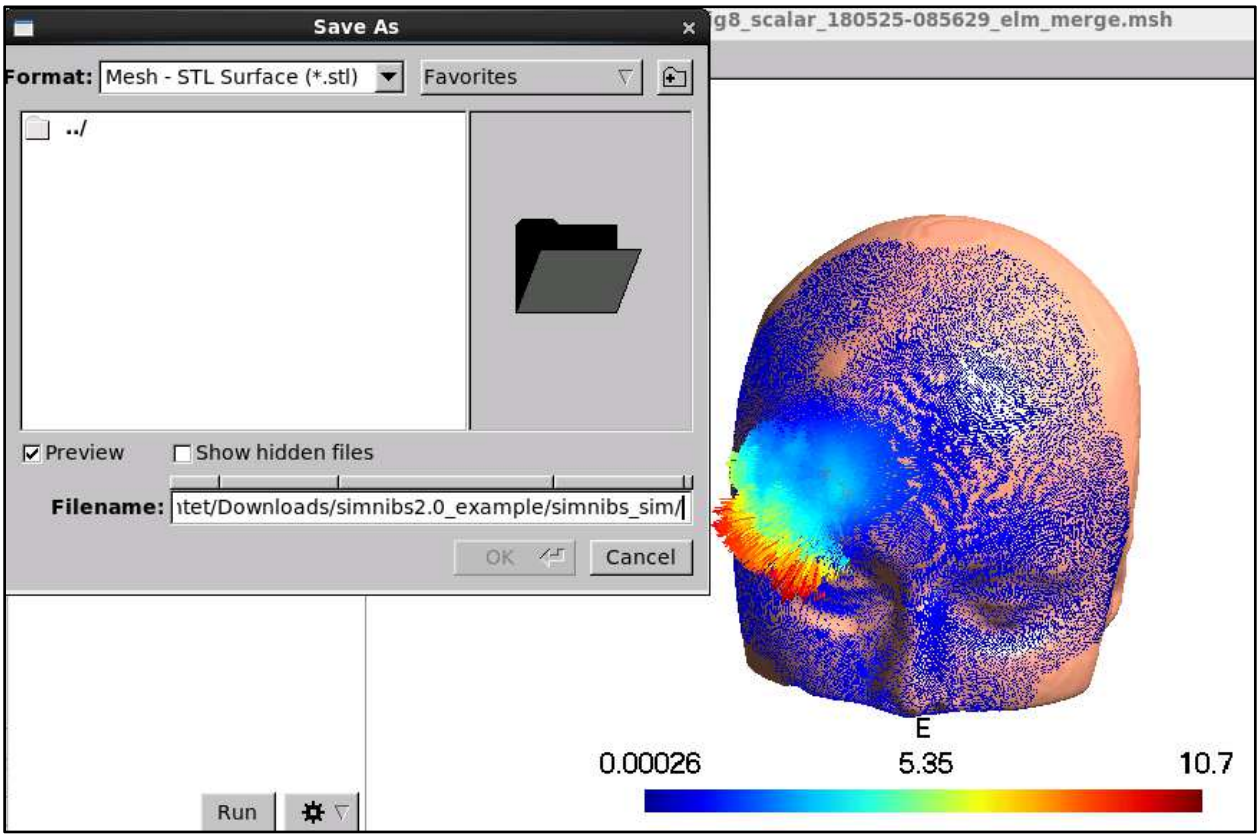

Figure 11: Saving STL File of Mesh in Gmsh. Original Drawing using Gmsh Tool

The electric field data from SimNIBS is imported as a matrix of values. Each row consists of the coordinate points of the vertices of each triangle or tetrahedron of the model, along with the electric field vector at each vertex. For example, the picture below shows x,y,z coordinates at column 5,6,7 respectively, and the corresponding electric field value at colum 8,9, and 10.

```
Matrix = dlmread(E_field.txt);
```

Matrix

1768715x54 double

	4	5	6	7	8	9	10
1	0	-1.4304	12.7242	-14.9285	-0.0542	-0.0200	-0.0633
2	1	-1.0719	22.3991	-17.2698	-0.0229	-0.0309	-0.0963
3	2	0.2383	20.2806	-17.8919	-0.0215	-0.0432	-0.1499
4	3	-1.8882	21.6187	-17.6378	0.0017	-0.0203	-0.0807

Figure 12: Electric Field Data Imported as Matrix in MATLAB

The average electric field value at each triangle of the mesh is calculated and plotted in MATLAB.

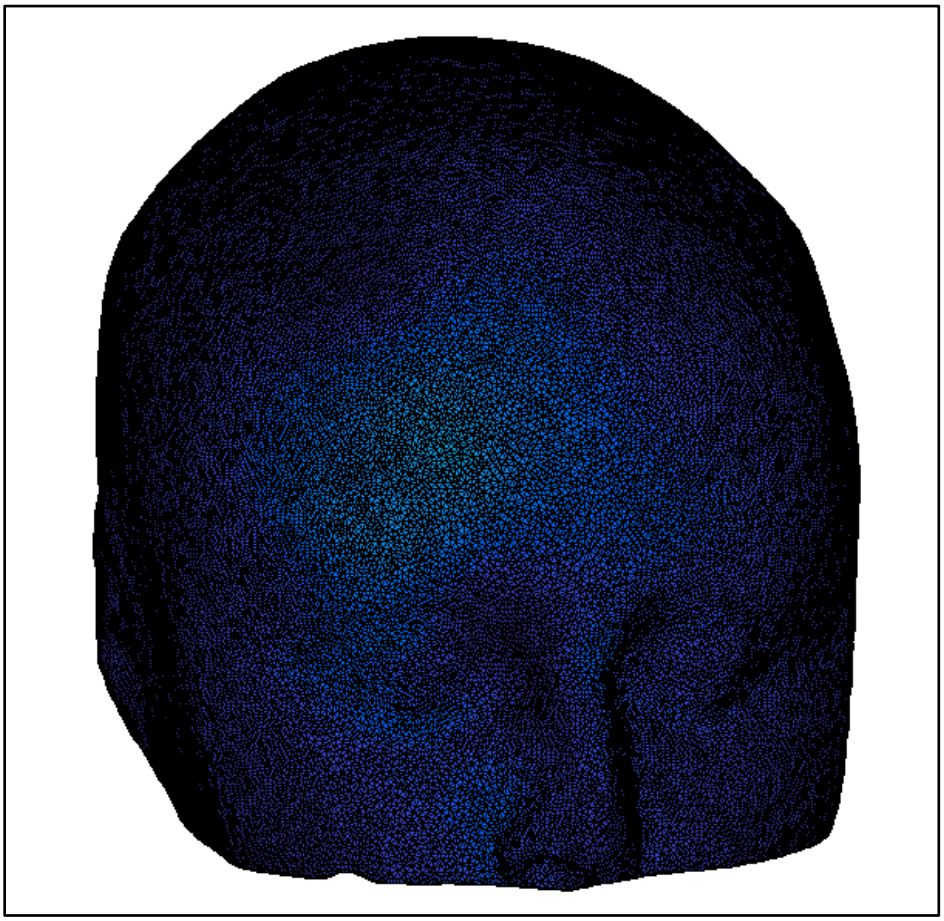

Figure 13: Electric Field Data Plotted in MATLAB. Original Drawing using SimNIBS Example Head Model

2.2.4.3. Importing Coil Model Into MATLAB

The coil model used in SimNIBS is of ccd file type. The file is an ASCII file formatted so that there are 6 columns of values, where the first three columns contains x,y,z coordinate values of the vertices, and the next three columns contains information about dipole moment vector.

The columns of values are imported into, and plotted in MATLAB. A coil model can be reconstructed in MATLAB using the information from ccd file, and used in simulations.

```
Matrix = dlmread('coil.ccd');
plot3(Matrix(:,1), Matrix(:,2), Matrix(:,3), 'o');
```

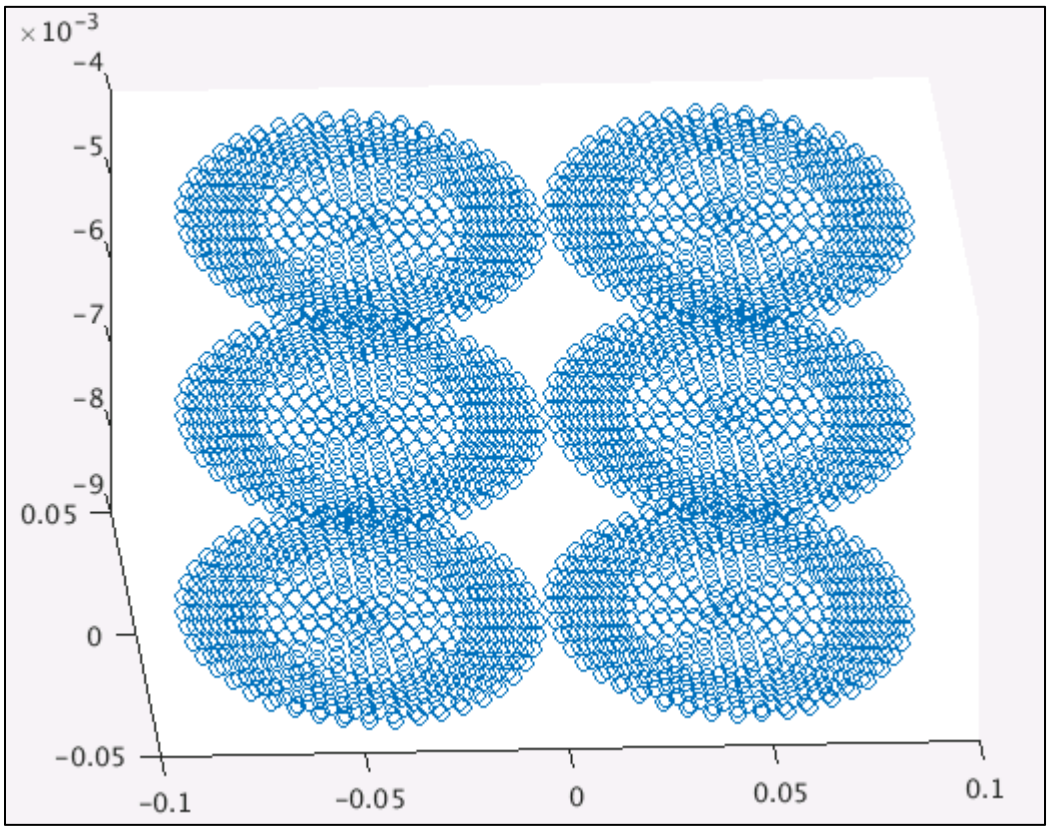

Figure 14: SimNIBS Coil Model Visualized in MATLAB. Original Drawing using SimNIBS Coil.

2.3 Decimation

The initial segmentation result is a coarse, and sometimes very dense, mesh, which is not computationally practical in simulation environment. To employ the model in a reasonable simulation time, the number of vertices in the mesh is reduced while maintaining the original topology. This decimation procedure is accomplished primarily in mesh processing software tool MeshLab. Among several available decimation schemes, quadric edge collapse decimation method is utilized. The method can be applied to reduce the mesh point clouds by a specified percentage or to a target number of triangles. Although typically, the whole model is decimated, it is occasionally necessary to decimate some regions selectively. For example, in a human model, regions of lower definitions such as torso may need more decimation than the regions of higher definitions such as face or fingers. This selective decimation can be achieved in MeshLab although it requires selecting the region of interest manually. Fully automatic selective decimation is developed in this study and described in Chapter 3.

The figure below shows the decimation of the fingers without compromising the definitions. The decimated mesh has significantly fewer number of triangles.

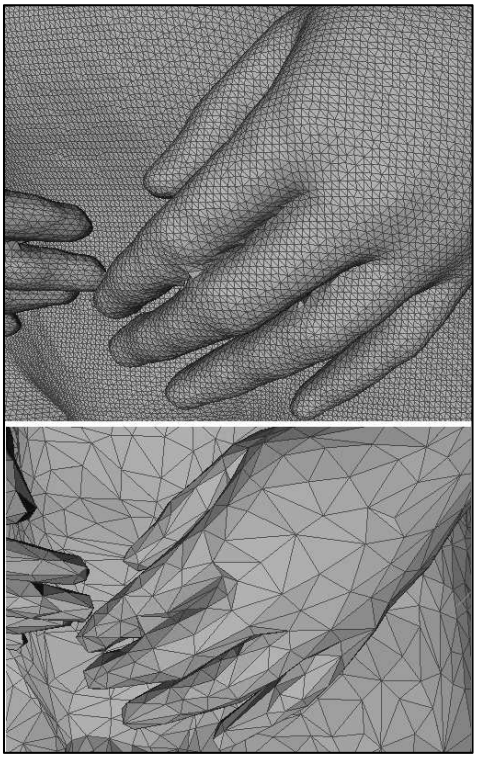

Figure 15: Original Mesh and Decimated Mesh. Original drawing

2.4 Mesh Processing

The decimated mesh may have triangles with bad quality and other common geometrical issues such as several number of triangles being attached to a single node. In order to resolve these issues, a variety of mesh processing algorithms may need to be applied.

To describe the quality of a triangle, the following sub-sections have been excerpted from my advisor's book "Low-Frequency Electromagnetic Modeling for Electrical and Biological Systems Using MATLAB" [10].

2.4.1. Triangle Quality and Mesh Quality [10]

The triangular mesh should be used to obtain a solution to differential or integral equations. The long narrow triangles are generally not desirable since electric charge and current distributions may significantly vary along their lengths. Such a feature is in contradiction with the general idea of surface discretization, where we typically assume that the electric (or other) parameters are approximately constant for every small triangle.

The best triangle is an equilateral triangle, with all three triangle angles equal to 60 degrees, i.e. $\alpha = \beta = \gamma = 60°$. In reality, the mesh cannot consist of only equilateral triangles. The *quality of the triangular element* is therefore introduced, which is essentially a measure of the deviation from an equilateral triangle. One common quality measure is the ratio between the radius of the inscribed circle (times two), r_{in}, and the radius of the circumscribed circle, r_{out}, – see Fig. 16.

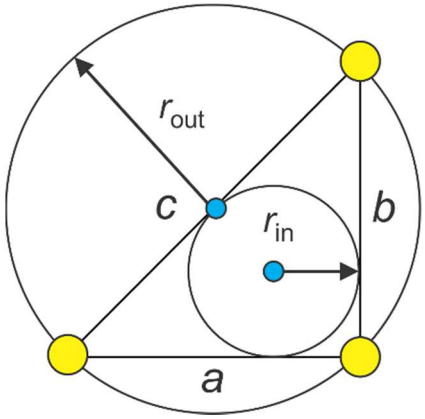

Figure 16: Radii of the inscribed circle (largest circle contained in the triangle) and the circumscribed circle (smallest circle containing the triangle), respectively, for a right-angled isosceles triangle. r_{in} is called the inradius and r_{out} is the circumradius.

The quality factor q is expressed by:

$$q \equiv \frac{2r_{in}}{r_{out}} = \frac{(b+c-a)(c+a-b)(a+b-c)}{abc} \tag{2.4}$$

where a, b, c are the triangle sides. Another useful expression is:

$$q = \frac{2r_{in}}{r_{out}} = 2(\cos\alpha + \cos\beta + \cos\gamma - 1)$$

2.4.2. Smoothing

Smoothing is applied to resolve triangles with low quality, and is accomplished via several smoothing algorithms available in MeshLab. The smoothing algorithm mainly utilized is Laplacian smoothing. A section about Laplacian Smoothing is included in Chapter 3, along with its weaknesses.

The figure below shows the triangles with bad quality before smoothing and the triangles with better quality after smoothing.

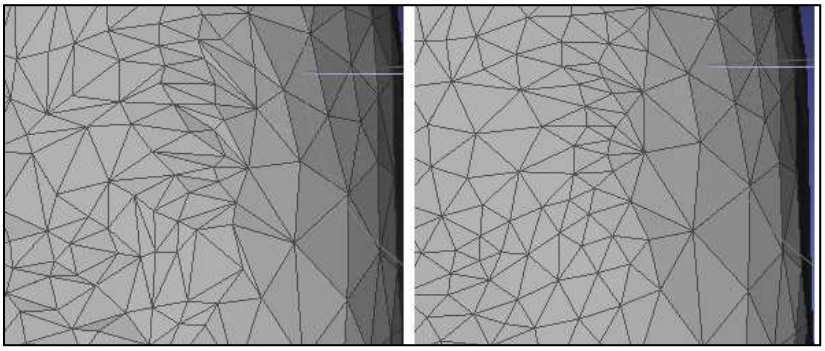

Figure 17: Mesh Smoothing. Original drawing

2.5 Validation of Models

After executing all mesh-processing procedures, the models need to meet specific criteria so that they become true CAD models fit for valid simulations.

The following subsections describe such criteria and how they are checked.

2.6.1. Specific Conditions for CAD Models [10]

The following two conditions are required for a true CAD human body model: (1) a 3D triangular mesh representing a solid object must not have holes; (2) the surface of a well-behaved triangular mesh in 3D must satisfy one critical condition, which is the so-called *manifold* condition. A mesh is *2-manifold* if every node of the mesh has a disk-shaped neighborhood of triangles. This neighborhood can be continuously deformed to an open disk. Every edge of a 2-manifold mesh is a manifold edge with only two attached triangles. All other meshes are non-manifold meshes and are not suitable for FEM (Finite Element Method) analysis. Figure 18 gives examples of a manifold edge, a non-manifold edge and, a non-manifold mesh with a non-manifold node.

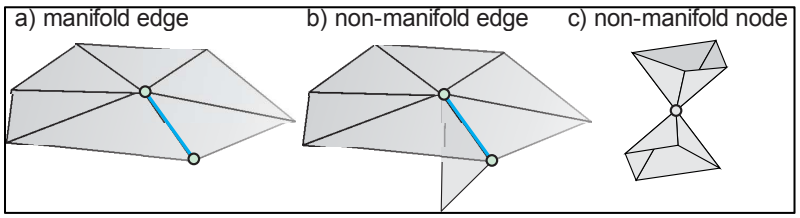

Figure 18:. a) – Examples of a manifold edge; b) – non-manifold edge, and c) – non-manifold node. [10]

2.6.2. Intersections [10]

In addition to the 2 conditions to CAD models, (1) must be watertight and (2) manifoldness, the CAD models must not intersect each other to be fit for simulations.

There are three types of triangle intersections as described below and the mesh must be free of all kinds of intersections.

Consider a master mesh X and a slave mesh Y. Both meshes are 2-manifold. For any triangle from the master mesh there exist three different intersection cases as shown in Fig. 19. Cases #1 and #3 in Fig. 19 would become equivalent if we could treat the master and slave meshes as one set of triangles.[10]

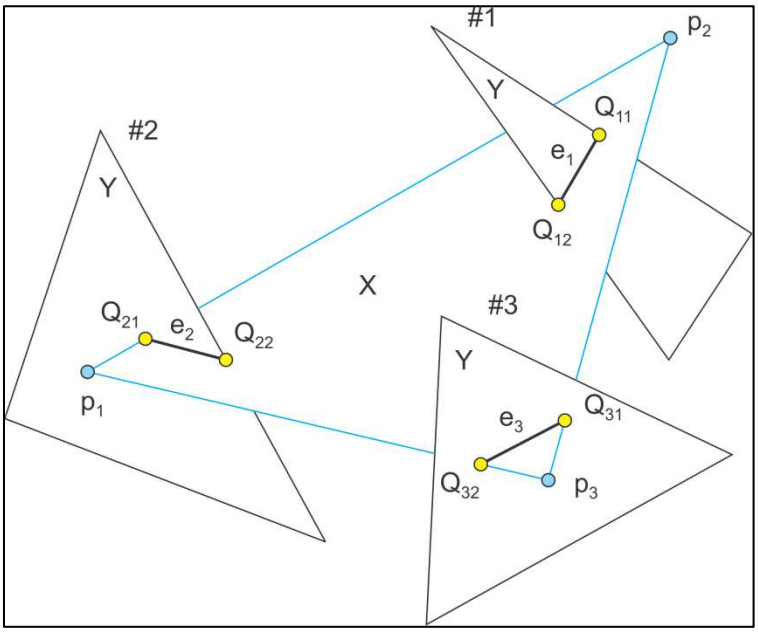

Figure 19: Three types of intersection of a triangle from a master mesh X with various triangles of a slave mesh Y. Cases #1 and #3 are equivalent if we treat the master and slave meshes as one set of triangles. Original Drawing from R.[10]

References

[1]. Makarov SN, Noetscher GM, Yanamadala J, Piazza MW, Louie S, Prokop A, Nazarian Ara, Nummenmaa A. Virtual Human Models for Electromagnetic Studies and Their Applications

[2]. M. J. Ackerman, "The Visible Human Project," Proc. IEEE, vol. 86, no. 3, pp. 504-511, Mar. 1998. Available: https://www.nlm.nih.gov/research/visible/visible_human.html

[3]. K. O. Genc, P. Segars, S. Cockram, D. Thompson, M. Horner, R. Cotton, and P. Young, "Workflow for creating a simulation ready virtual population for finite element modeling," J. Med. Devices, vol. 7, no. 4, Dec. 2013.

[4]. M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," Journal of Electronic Imaging, vol. 13, no. 11, pp. 46-165, Jan. 2004.

[5]. P. A. Yushkevich, J. Piven, H. C. Hazlett, R. G. Smith, S. Ho, J. C. Gee, and G. Gerig, "User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability," Neuroimage, vol. 31, no. 3, pp.1116-1128, Jul. 2006.

[6]. P. A. Yushkevich, Y. Gao, and G. Gerig, "ITK-SNAP: An interactive tool for semi-automatic segmentation of multi-modality biomedical images," 38th Annu. Int. Conf. of the IEEE Eng. in Medicine and Biology Soc., Orlando, FL, Aug. 16-21, 2016.

[7]. D. L. Pham, C. Xu, and J. L. Prince, "Current Methods in Medical Image Segmentation," Annu. Review of Biomedical Eng., vol. 2, pp. 315-337, Aug. 2000. Available: http://www.annualreviews.org/doi/full/10.1146/annurev.bioeng.2.1.315

[8]. F. Zhao and X. Xie, "An Overview of Interactive Medical Image Segmentation," Ann. of the BMVA, vol. 2013, no. 7, pp 1–22, 2013. Available: http://www.bmva.org/annals/2013/2013-0007.pdf

[9]. Internet Analysis Tools Registry. Harvard Univ., Available: http://www.cma.mgh.harvard.edu/iatr/index.php

[10]. S. N. Makarov, G. N. Noetscher, and A. Nazarian, "Low-Frequency Electromagnetic Modeling for Electrical and Biological Systems Using MATLAB", Wiley, New York, June 2015, 648 p.,

[11]. P. A. Yushkevich, Y. Gao, and G. Gerig, "ITK-SNAP: An interactive tool for semi-automatic segmentation of multi-modality biomedical images," 38th Annu. Int. Conf. of the IEEE Eng. in Medicine and Biology Soc., Orlando, FL, Aug. 16-21, 2016.

[12]. FreeSurfer. Online: https://surfer.nmr.mgh.harvard.edu/fswiki/

[13]. FreeSurfer. Online: https://surfer.nmr.mgh.harvard.edu/fswiki/FreeSurferAnalysisPipelineOverview

[14]. F. Segonne, A.M. Dale, E. Busa, M. Glessner, D. Salat, H.K. Hahn, and B. Fischl, "A Hybrid Approach to the Skull Stripping Problem in MRI," Available: http://surfer.nmr.mgh.harvard.edu/ftp/articles/segonne05.pdf

[15]. Slicer. Online: https://www.slicer.org/w/images/2/24/AutomaticSegmentation_SoniaPujol.pdf

[16]. SimNIBS. Online: http://simnibs.de/

CHAPTER 3

DECIMATION BASED ON SURFACE CURVATURE

3.1 Introduction

In the process of decimation, it is crucial not to compromise the original topology of the mesh. Usually, meshes may have areas of varying curvatures across the surface. In such cases, decimation on the entire mesh through several iterations will undermine the quality of the mesh in the areas of high definitions since the triangles are uniformly decimated across the entire mesh. To avoid this, an algorithm to decimate based on surface curvature is devised. In this decimation method, curvature of the surface at each triangle is calculated and triangles with lowest curvature values are given priority in decimation.

As an introduction to mesh decimation, the following sub-section has been excerpted from my advisor's book "Low-Frequency Electromagnetic Modeling for Electrical and Biological Systems Using MATLAB" [1]

3.1.1 Mesh Decimation [1]

Quite often, a dense mesh in certain areas is not required. Depending on factors including the problem geometry, field gradient, and value of the error function, large triangles may provide an adequate solution, enabling reduction of the problem size and an increase in the speed of the calculation. The process of reducing mesh density in certain areas is known as *mesh decimation*. One popular algorithm for mesh decimation is the (incremental) *edge collapse* schematically shown in Fig. 20a (see, for example, Ref.[2]). Two nodes p_1, p_2 of an edge (usually the shortest edge in the area of interest) are replaced by one node p_{12} at the edge center and the mesh topology is updated. Yet another algorithm is the *vertex removal* schematically shown in Fig. 20b. Node p_1 and all triangles attached to this node is removed and the resulting hole is re-triangulated.

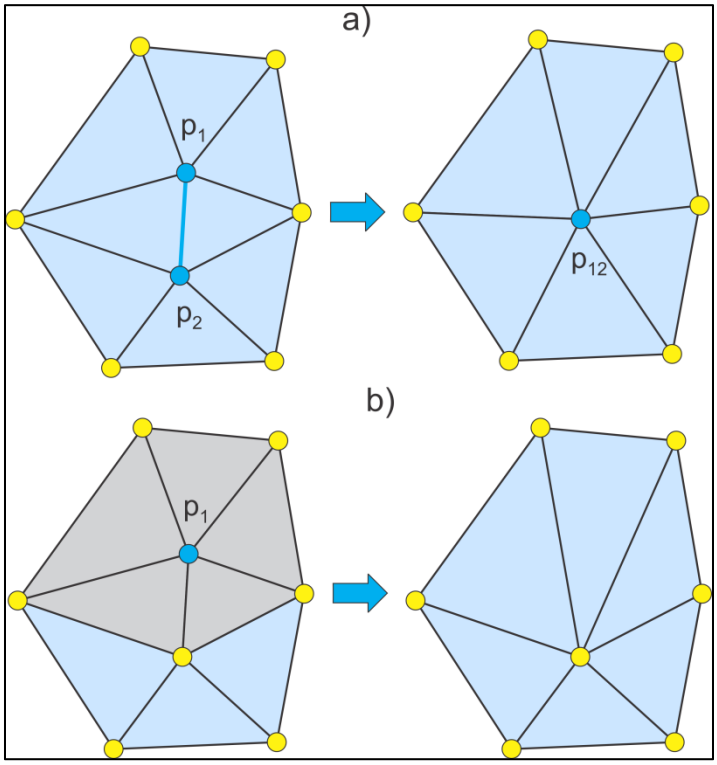

Figure 20: a) - Edge collapse method; b) – vertex removal. Original Drawing from Ref.[1]

3.2 Calculation of Surface Curvature

For the purpose of this decimation algorithm, surface curvature at a triangle is defined as the metric by which a normal vector of the triangle surface deviates from the normal vectors of its surrounding triangles.

This is illustrated in the following figure. For example, to find the surface curvature at the green triangle, the neighboring triangles in red are determined. The normal vectors of all triangles are calculated. The angle between the normal vector of the green triangle and that of each neighboring triangle in red is determined. The average angle is considered the surface curvature at the triangle.

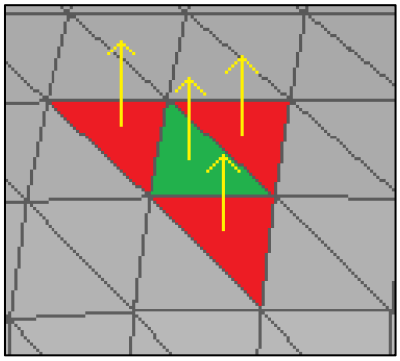

Figure 21: Surface Curvature at a Triangle Depicting Neighboring Triangles and the Normal Vectors. Original Drawing using Mesh Loaded in Meshlab

To determine surface curvature at each of the triangles in the mesh, the following steps are performed:

1. The normal vectors of all triangles in the mesh are computed first.
2. For each triangle, the neighboring triangles are determined.
3. The angle between the normal of the triangle and that of each attached triangle is calculated.
4. The average angle is determined for each triangle.

3.2.1 Calculation of Normal Vectors

The normal vector is defined as the normalized cross-product of two triangle edges.[1].

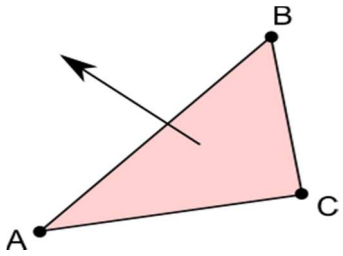

Figure 22: Normal of a Triangle. Drawing taken from (https://docs.unity3d.com/ScriptReference/Plane.html)

The normal vectors of a given mesh is calculated using the script `MeshNormals.m` in Chapter 2 of the Ref [1].

3.2.2 Determining Neighboring Triangles

Neighboring triangles are determined by manipulating the arrays of vertices and arrays of faces (triangulation matrix) of the mesh in MATLAB. The triangulation matrix is an identification of triangles based on the indices of the vertices matrix. Each triangle is identified by three indices of its vertices.

The following subsection describes the details of the triangulation matrix:

3.2.2.1 Arrays of Vertices and Faces [1]

Arrays of vertices and faces

A triangular surface mesh is the base of any surface representation including various numerical methods in electrical and biomedical engineering, computer graphics, etc. […] Our goal is to "cover" its surface with triangles – *simplexes* in 2D. Many ways of doing so exist. One such way is shown in Fig.23.

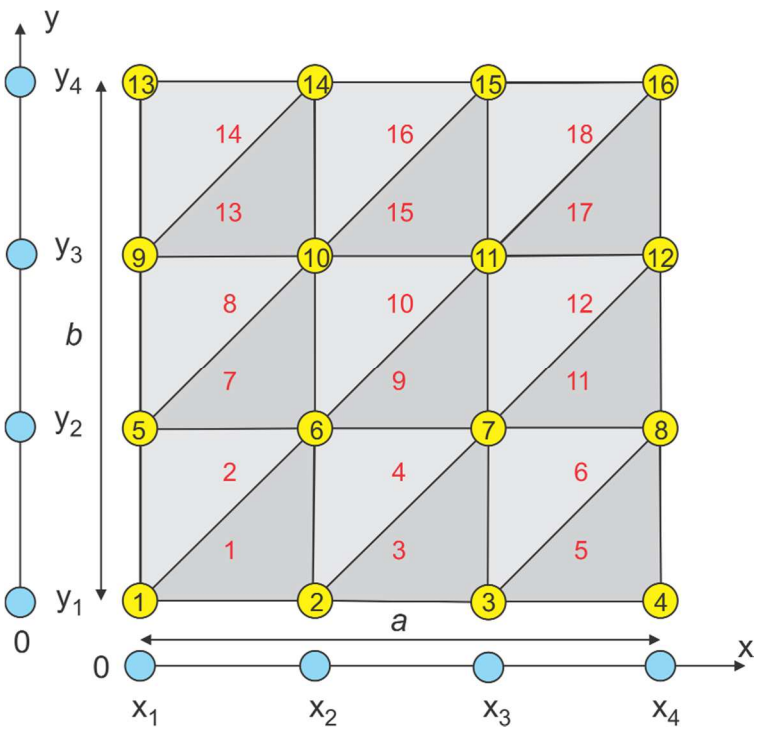

Figure 23: Mesh generation for a planar rectangle. Drawing from Ref. [1].

We first define uniformly spaced *x*-nodes and uniformly spaced *y*-nodes. Assume that there are N_x+1 nodal points along the *x*-axis and N_y+1 nodal points along the *y*-axis. In Fig.23, $N_x=3, N_y=3$. In a general case, one has

$$x_m = \left(\frac{m-1}{N_x}\right)a, \quad m = 1,...,N_x + 1$$

$$y_n = \left(\frac{n-1}{N_y}\right)b, \quad n = 1,...,N_y + 1$$

(2.1)

There is a common way of describing triangular meshes, which originates from old NASTRAN programs written in the 1970s and 1980s. In order to define a triangular mesh, we need the *array of vertices* (or nodes), *P*. This array consists of rows; every row includes three Cartesian coordinates of the corresponding nodal point. The row number in array *P* is simply the vertex number. Further, we need an *array of faces* (or triangles), *t*. This array also consists of rows; every row includes three integer numbers of triangle vertices; each such number is simultaneously the row number of the array *P*. The row number in the array *t* is the number of the triangular face. For the mesh shown in Fig. 23, there are 16 nodes and 18 triangles, giving:

$$P = \begin{bmatrix} x_1 & y_1 & 0 \\ x_2 & y_1 & 0 \\ x_3 & y_1 & 0 \\ x_4 & y_1 & 0 \\ x_1 & y_2 & 0 \\ x_2 & y_2 & 0 \\ x_3 & y_2 & 0 \\ x_4 & y_2 & 0 \\ x_1 & y_3 & 0 \\ x_2 & y_3 & 0 \\ x_3 & y_3 & 0 \\ x_4 & y_3 & 0 \\ x_1 & y_4 & 0 \\ x_2 & y_4 & 0 \\ x_3 & y_4 & 0 \\ x_4 & y_4 & 0 \end{bmatrix} \qquad t = \begin{bmatrix} 1 & 2 & 6 \\ 1 & 5 & 6 \\ 2 & 3 & 7 \\ 2 & 6 & 7 \\ 3 & 4 & 8 \\ 3 & 7 & 8 \\ ... & & \\ 11 & 15 & 16 \end{bmatrix} \Rightarrow t = \begin{cases} \text{odd row :} \\ [m, m+1, m+N_x+2] + (n-1)(N_x+1) \\ \\ \text{even row :} \\ [m, m+N_x+1, m+N_x+2] + (n-1)(N_x+1) \\ m = 1,...,N_x, \quad n = 1,...,N_y \end{cases}$$

(2.2)

Connectivity

The *triangular mesh* which covers the surface of the rectangle in Fig. 23 is uniquely characterized by the array of vertices, *P,* and the array of faces, *t*. Thus, the mesh is simply a collection of vertices and faces. The face information is a part of the c*onnectivity information* – constructing the faces means establishing the connectivity between different vertices (or nodes). Other (secondary) parts of the connectivity information may include an array of edges, *e*, array of edges attached to each node, array of triangles attached to each node (one or more triangles), array of triangles attached to each edge (one or two triangles), etc. Meaning of connectivity is wide: one may reply upon face connectivity data, edge connectivity data, etc.

3.2.2.1 Determining Neighboring Triangles

To determine the neighboring triangles of a particular triangle, the triangulation matrix is searched for any triangle that shares two common vertices of a given triangle. In other words, the triangles that share an edge with a particular triangle in the triangulation are considered neighboring triangles of that particular triangle.

In the code below, the triangles that share the same edge are searched for each edge of a given triangle. The resulting triangles are stored in a cell for calculation of surface curvature.

```
    % find neighbors of each triangle
    N_T = cell(length(t),1);
    for i=1:length(t)
        % neighbor triangles of each triangle
        n_t = [];

        % triangles attached to first edge
        [a b]=find(t==t(i,1));
        [c d]=find(t==t(i,2));
        n_t = [n_t;intersect(a,c)];

        % triangles attached to second edge
        [a b]=find(t==t(i,2));
        [c d]=find(t==t(i,3));
        n_t = [n_t;intersect(a,c)];

        % triangles attached to third edge
        [a b]=find(t==t(i,1));
        [c d]=find(t==t(i,3));
        n_t = [n_t;intersect(a,c)];

        % keep neighboring triangles of each triangle
        N_T{i}=(unique(n_t))';
    end
```

3.2.2.2. Calculation of Angle between Normal Vectors

To calculate angle between normal vectors, MATLAB built-in function subspace is utilized.

MATLAB subspace function serves the purpose of finding the angle between two vectors. The average of the angles between all neighboring triangles is taken as curvature value.

The validity of the curvature calculation is demonstrated by the following cylinder, whose triangles are colored based on their respective curvature values. The triangles with lower curvature values are reflected by darker colors while those with higher curvature values are in lighter colors. Accordingly, the flat surface on the cap that instinctively has curvature value of zero is colored very dark. Towards the edge of the cap, the curvature values grow higher and therefore the color of the triangles become lighter.

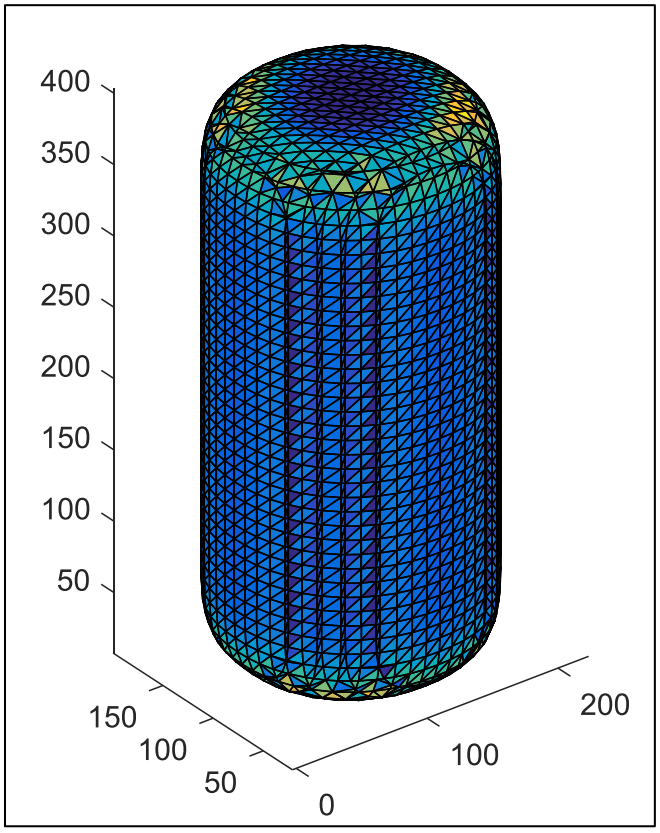

Figure 24: Demonstration of Curvature Calculation. Original Drawing using cylinder mesh taken from (https://www.mathworks.com/matlabcentral/fileexchange/32573-patch-curvature)

3.3 Decimation on Area of Lower Curvature

Once the curvature values of the triangles are determined, the triangle that has the lowest curvature is determined for decimation.

In each iteration of this decimation algorithm, the following steps are executed:

1. The triangle with the lowest curvature value is determined.
2. Lengths of the edges of the triangle are determined using Euclidean distance formula: given 2 vertices (x1,y1,z1) and (x2,y2,z2), distance between the 2 vertices D

$$D = \sqrt{(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2}$$

3. Shortest edge is determined.
4. The shortest edge is collapsed by merging the 2 vertices of the edge together as follows:
    a. Assign vertex2 to vertex1:Vertex1=Vertex2

b. In triangulation matrix, triangles with two same vertices are searched and deleted.

The following figure illustrates the decimation results of the algorithm. In Fig.25, the algorithm is applied 5 times on the VHP-Female bladder mesh, and the area with lower curvature (in darker color) is decimated. The resulting triangles have lower but acceptable quality.

However, applying the algorithm multiple times generates triangles with undesirable quality in the decimated area. As seen in Fig. 26, when decimation is executed 25 times, the area with lower curvature ends up having very few edges, resulting in low quality triangles.

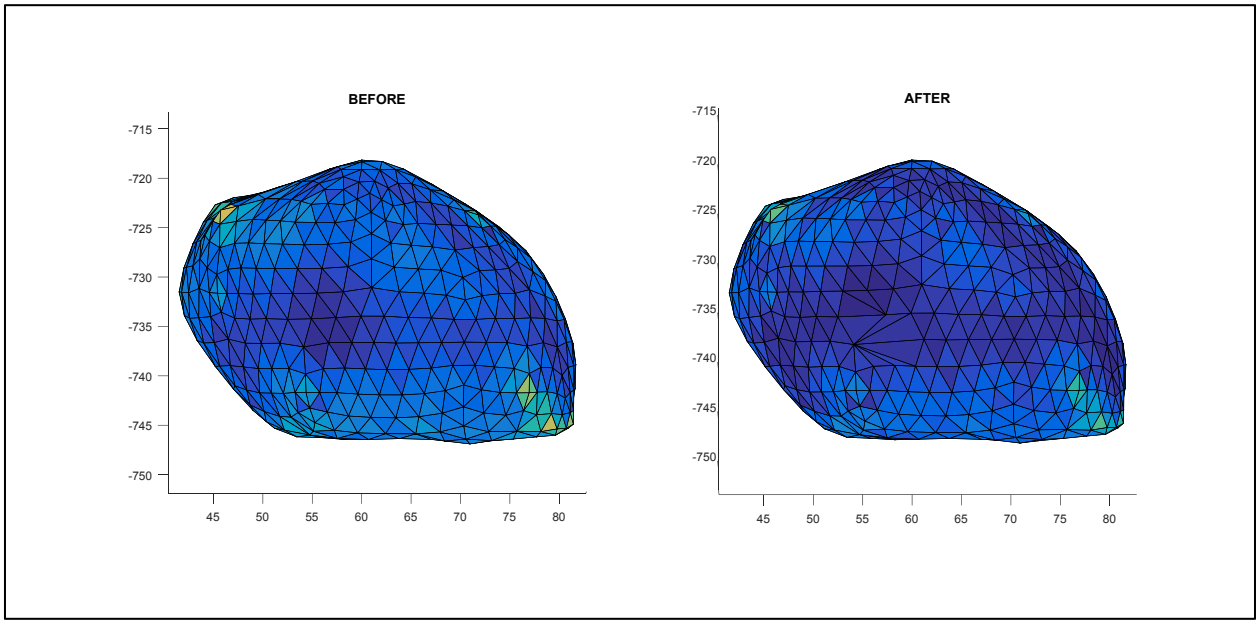

Figure 25: Decimation Result of Algorithm (5 iterations). Original Drawing

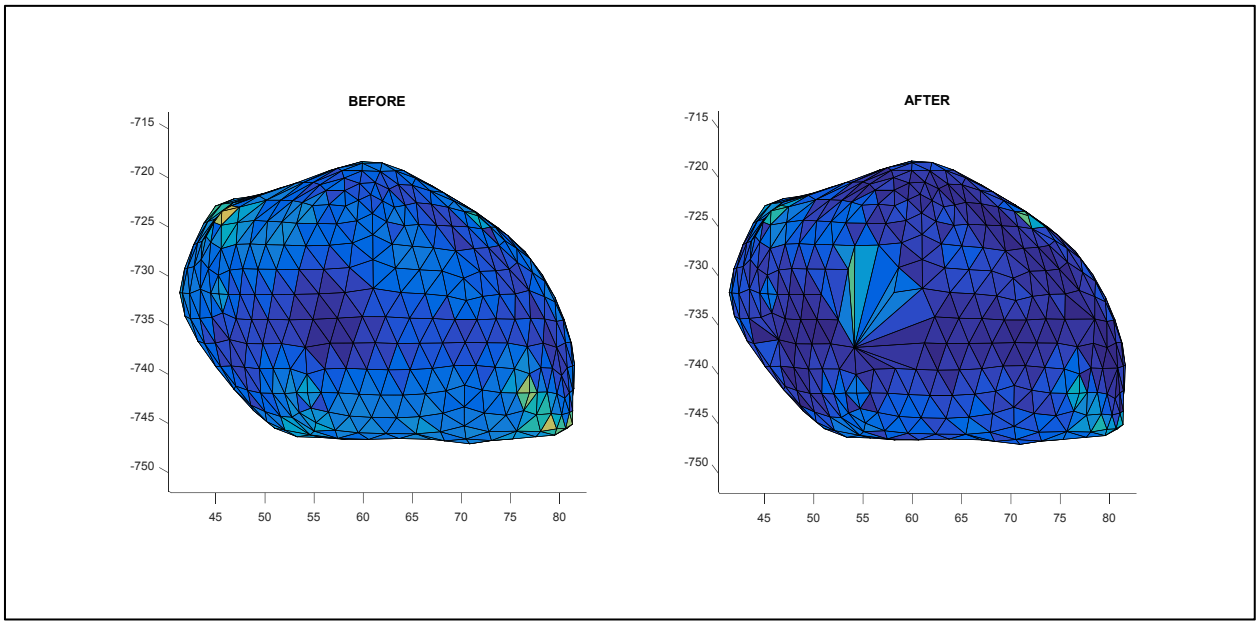

Figure 26: Decimation Result of Algorithm (25 iterations). Original Drawing

3.4 Local Laplacian Smoothing

To fix the issue of low quality triangles in decimated area, Laplacian smoothing algorithm is applied to the local nodes.

To provide an overview of the Laplacian smoothing algorithm used in the decimation algorithm, the following sub-sections have been excerpted from my advisor's book "Low-Frequency Electromagnetic Modeling for Electrical and Biological Systems Using MATLAB" [1].

3.4.1 Necessity of Mesh Smoothing [1]

Often, the (constrained) 2D Delaunay triangulation algorithm *for planar meshes* is readily available, in particular using the basic MATLAB platform. Then, a mesh for a 2D convex or any non-convex *complicated shape* represented by multiple polygonal boundaries might be created as follows:

- We select a *larger domain* of a simple shape that contains all boundaries (a rectangle, circle, etc.). We fill this domain with nodes distributed either randomly or, alternatively, as nodes of equilateral triangles of a certain size *excluding* the nodes outside the domain. The resulting node array is P.

- We analytically specify the *required* boundary nodes, P_b, and the corresponding boundary edges, C. The boundary nodes are put *up front* in the resulting node array, $P \to [P_b; P]$, in order to keep the connectivity in C. This is a very critical step.

- We apply the constrained Delaunay triangulation with constrains on C and select subdomains if necessary. Such subdomains may be, for example, electrodes attached to a conducting object.

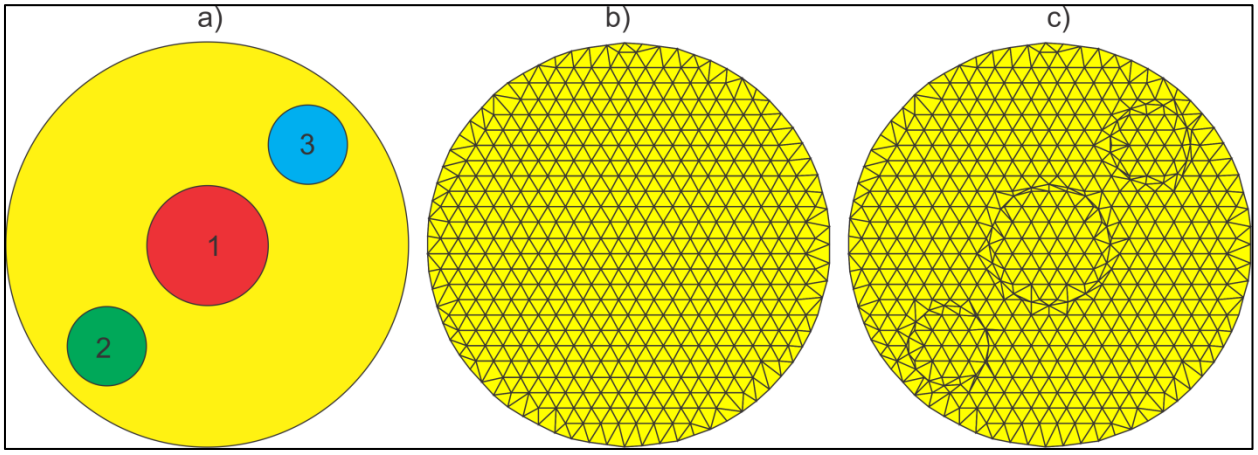

Figure 27: Problem geometry with non-intersecting boundaries and the initial meshes. Drawing from Ref.[1]

3.4.2 Topology-preserving Laplacian smoothing [1]

According to Ref.[3], "there are mainly three types of mesh improvement methods: (1) refinement or coarsening, (2) edge swapping, and (3) mesh smoothing. The refinement and the coarsening mainly try to optimize the mesh density, while edge swapping and mesh smoothing mainly aim to optimize the shape regularity. There are mainly two types of smoothing methods, namely Laplacian smoothing and optimization-based smoothing."

In its basic form, Laplacian smoothing implies moving each vertex to the arithmetic average of the neighboring vertices while keeping the boundary nodes and boundary connectivity (boundary edges) unchanged as shown in Fig. 28. In other words, a free vertex of the mesh is simply relocated to the centroid of the vertices connected to that vertex. In Fig. 28, the node p_5 is moved to the average of the neighboring points p_1, p_2, p_3, p_4. In this way, the local triangle quality greatly improves. Indeed, in order to perform the smoothing we should know the neighboring nodes (neighboring triangles) for every mesh node.

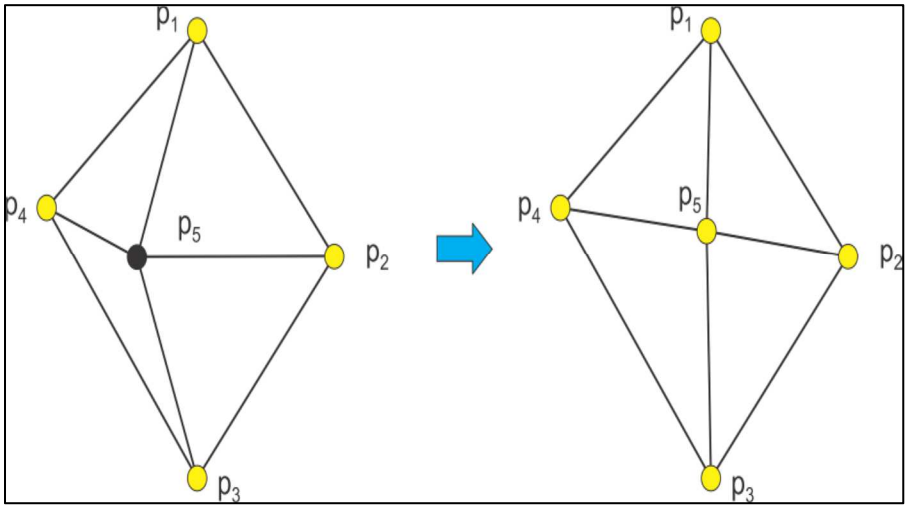

Figure 28: Concept of Laplacian smoothing. Drawing from Ref. [1]

Reference [4] provides a comprehensive overview of different Laplacian smoothing methods. Some of them are briefly listed below.

Standard Laplacian smoothing shown in Fig. 28:

$$p^* = \frac{1}{k} \sum_{p_j \in \Omega_i, p_j \neq p_i} p_j \tag{3.1}$$

where Ω_i is the "star" of the vertex p_i having k points and p^* is the new location of p_i. Note that this formulation can also be interpreted as a torsion-spring system where a central node in a star polygon is located at the centroid of the polygon balancing out the system to stay in (local) equilibrium.

Lumped Laplacian smoothing,

$$p^* = \frac{1}{3} p_i + \frac{2}{3} \frac{1}{k} \sum_{p_j \in \Omega_i, p_j \neq p_i} p_j \tag{3.2}$$

Centroid Voronoi Tessellation (CVT) smoothing utilizing attached triangle centers t_j and areas A_j,

$$p^* = \sum t_j A_j / \sum A_j \tag{3.3}$$

Weighted Centroid of Circumcenters (WCC) smoothing utilizing attached triangle circumcenters c_j and areas A_j,

$$p^* = \sum c_j A_j / \sum A_j \tag{3.4}$$

Equally-weighted versions of Eqs (3.3), (3.4) may be used.

Laplacian Smoothing with Re-triangulation. Iterative Algorithm [1]

After the creation of the initial mesh, the Laplacian smoothing of any type is applied to all free nodes except the boundary nodes. After that, the constrained Delaunay triangulation with the constraints is defined. Then, the Laplacian smoothing is applied again and the process repeats itself iteratively. One measure of convergence is the control of the resulting mesh quality (defined in Chapter 2). The process may be stopped when the triangle quality ceases to increase or it oscillates about a certain value. Note that the Laplacian smoothing does not necessarily converge except for the algorithm ***Lumped Laplacian smoothing,***

Lumped Laplacian smoothing [5].

There are two techniques to calculate new positions, p^*. The first method is to modify *all* positions by one step. This method is called the *simultaneous version*. The second variant is to update the new positions of p^* immediately. This variant is called the *sequential version*. In this latter case, a position p^* may not solely depend on the "set" of old positions but can also depend on previously calculated new positions. The Laplacian algorithm ***Standard Laplacian smoothing*** has the property that the limits of the two techniques are the same if they exist [5].

Applied to the mesh shown in Fig. 27c, algorithms ***CVT*** and ***WCC*** will show the best performance. They reach the highest mesh quality in a shortest number of iterations. Algorithm ***WCC*** is slightly faster than algorithm ***CVT***. Fig. 29 shows two meshes. One is

the initial mesh and another is the mesh obtained at 9th iteration using the iterative method described above with algorithm ***WCC***. The minimum mesh triangle quality increases from 4×10^{-5} to 0.69, which is a remarkable result given the simplicity of the algorithm. The usefulness of the Laplacian smoothing thus speaks for itself.

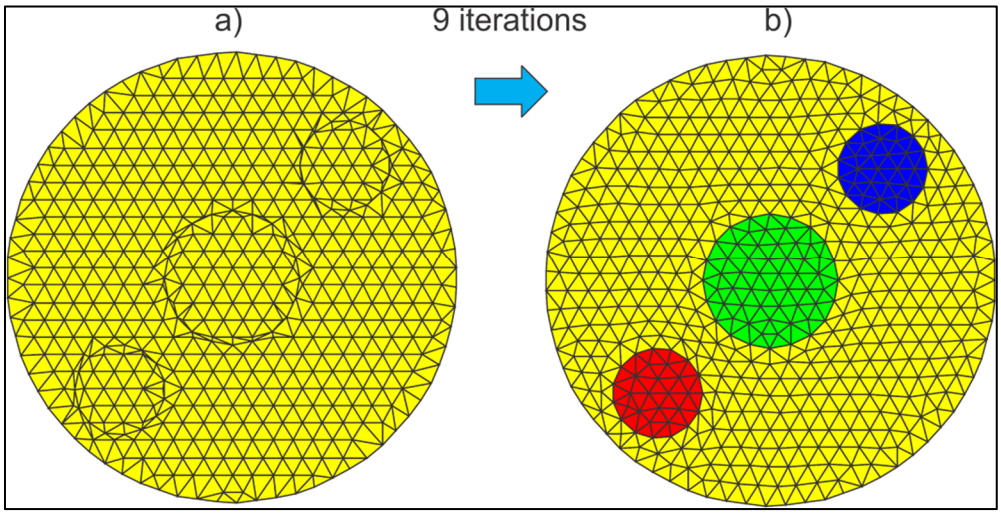

Figure 29: Results of Laplacian smoothing with algorithm ***WCC*** after 9th iteration. Drawing from Ref. [1]

3.4.3 Weaknesses of Laplacian Smoothing[1]

The Laplacian smoothing algorithm described in previous subsections has the following weaknesses:

- It is a *local* algorithm with fixed boundary nodes; therefore it is not able to provide high-quality uniform meshes from an arbitrary set of given nodes. If the nodes were initially concentrated in one local area, they will stay there forever.
- For the same reason, it is not well suited for creating high-quality, non-uniform meshes with different triangle sizes from a given set of data points;
- The proper handling of boundary nodes is an important subject. The method of fixed nodes used in this subsection is definitely not the best; it only works when the length of a boundary edge is approximately the best anticipated edge length for a given mesh. Also, intersecting boundaries require special care.

Laplacian smoothing in three dimensions

An undesirable effect is also the obvious "shrinkage" of 3D triangular surface meshes after Laplacian smoothing; the entire 3D mesh actually becomes smaller that it is in reality after several iterations[5]. A simple algorithm to avoid the shrinkage has been

developed[5]. The idea is to push the modified points toward the previous points and the original points of the mesh.

3.4.4 Local Laplacian Smoothing Results on Selective Decimation Algorithm

In each iteration of the algorithm, the nodes attached to the deleted vertex of the collapsed edge is determined. Laplacian smoothing on these nodes are then performed. The result shown in Fig. 30 shows the Laplacian smoothing performed on the nodes attached to decimated nodes. In contrast to decimation result without smoothing, the triangles now have much more acceptable quality.

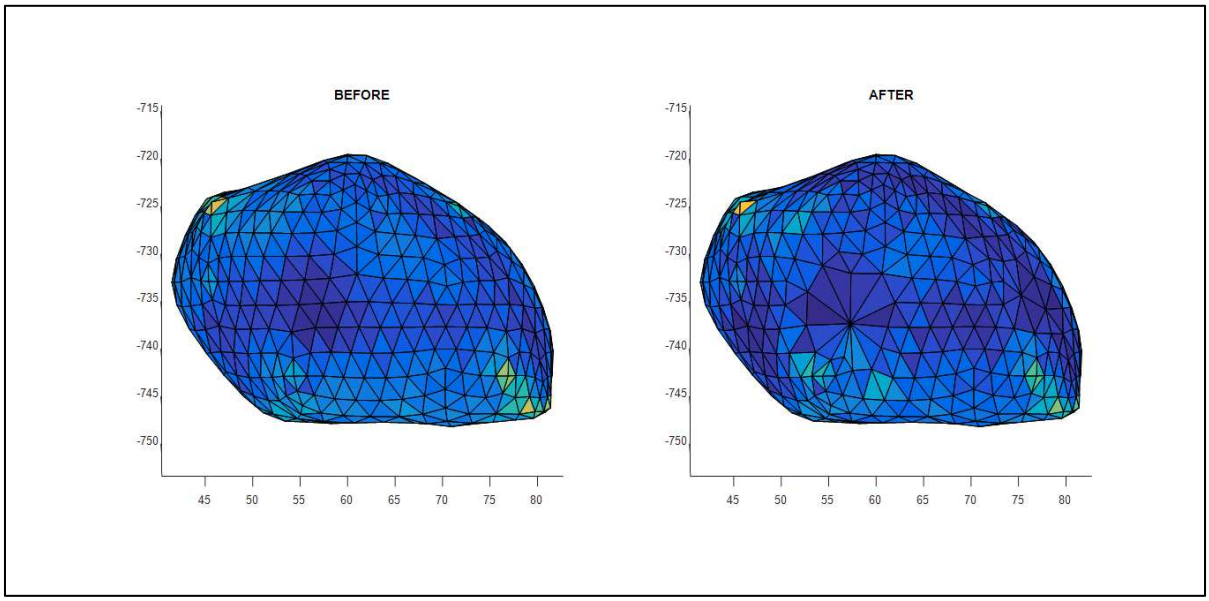

Figure 30: Decimation of Lower Curvature Area (25 iterations) with Local Laplacian Smoothing on Attached Nodes of Deleted Vertex. Original Drawing

Although the low quality triangles could be resolved using local Laplacian Smoothing, the result is not ideal since there are still many triangles attached to a single node.

In order to achieve a more decent smoothing results, smoothing is applied to more nodes in the surrounding region. This is realized by finding the attached nodes two times. The nodes attached to the deleted node are determined first. Again, the attached nodes of the resulting nodes are determined in order to apply Laplacian smoothing to a wider surrounding region.

The function to find attached nodes are called two times in the MATLAB code.

```
    % find attached nodes to the merged node
    lp_nodes = find_att_nodes(t,deleted_vertex);
    tmp_lp_nodes = lp_nodes;
    for i=1:length(tmp_lp_nodes)
        lp_nodes = [lp_nodes, find_att_nodes(t, tmp_lp_nodes(i))];
    end
```

This eliminates the issue of multiple triangles attached to a single node as shown in the following figure.

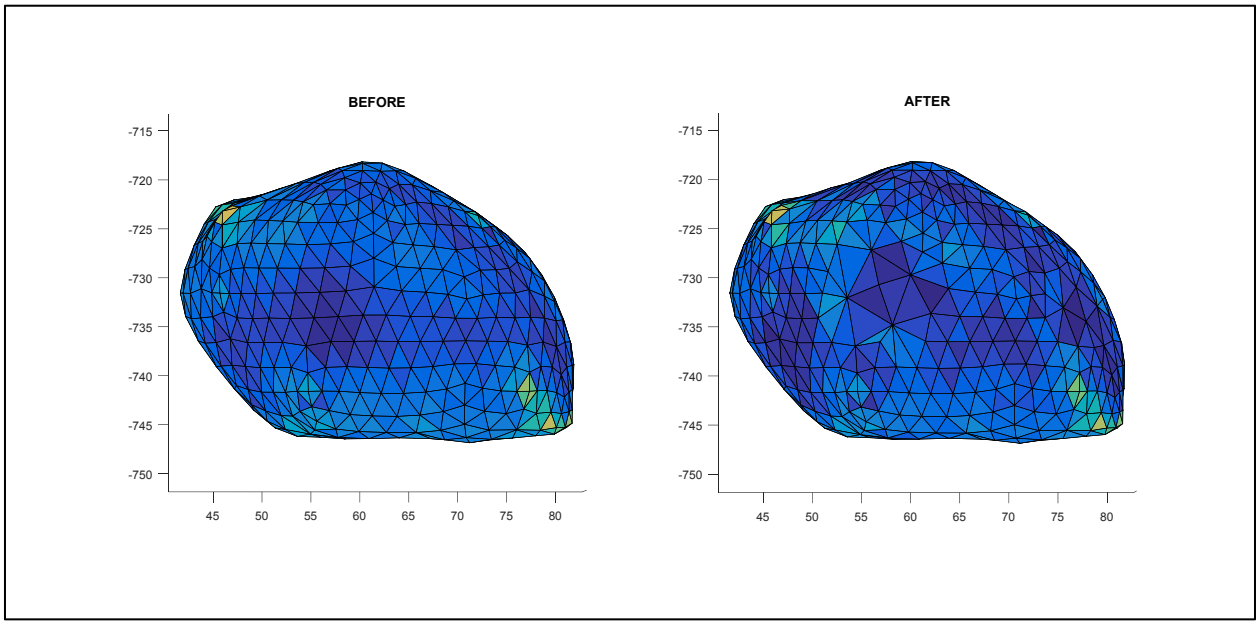

Figure 31: Decimation of Lower Curvature Area (25 iterations) with Local Laplacian Smoothing on More Nodes. Original Drawing

References

[1]. S. N. Makarov, G. N. Noetscher, and A. Nazarian, "Low-Frequency Electromagnetic Modeling for Electrical and Biological Systems Using MATLAB", Wiley, New York, June 2015, 648 p.,

[2]. J. R. Shewchuk, Delaunay Refinement Mesh Generation, PhD thesis, Computer Science Department, Carnegie Mellon University, May 18, 1997, Pittsburgh, PA.

[3]. L. Chen, "Mesh smoothing schemes based on optimal Delaunay triangulations," In Proc. 13th International Meshing Roundtable, Williamsburg, VA, Sandia National Laboratories, SAND #2004-3765C, pp.109-120, Sep. 19-22, 2004. Online: http://www.imr.sandia.gov/papers/abstracts/Ch317.html

[4]. H. Erten, A. Ungor, and C. Zhao, "Mesh Smoothing Algorithms for Complex Geometries," in: B. W. Clark (ed.), Proc. 18th Int. Meshing Roundtable, Springer-Verlag, Berlin-Heidelberg, 2009, pp. 175-193.

[5]. J. Vollmer, R. Mencl, and H. Müller, "Improved Laplacian smoothing of noisy surface meshes," Computer Graphics Forum, vol. 18 no. 3, Sep. 1999, pp. 132-138.

CHAPTER 4

METHODS OF FULL-BODY SHELL CREATION

4.1 Introduction

In creating initial CAD human models, standard semiautomatic and manual segmentation methods are used [1]-[7]. After initial segmentation, there are several mesh processing procedures such as decimation, and deformation to be performed so that the model can be deployed in a reasonable simulation time. During the initial segmentation as well as mesh conditioning procedures, errors can be introduced in the model. These errors give rise to the intersection in the individual meshes as well as among different CAD objects.

These intersections could be resolved using various intersection resolution methods [8]-[11]. However, these methods usually result in significantly high number of low quality triangles around the intersection area. This is undesirable because higher number of triangles increases the simulation time. Another issue is that with standard intersection resolving methods, coincident faces are left in contact regions, which are typically not defined to be imported. There are specific operations that could be performed to import these faces but these often result in incompatibility issues [12].

To avoid the intersection problems and CAD import errors, a thin gap is introduced between any two tissue objects [13]-[16]. To do so, nodes around the intersection region are moved outwards until the intersection issues are resolved.

Given normal vectors $\boldsymbol{n}_i$ of N triangular facets surrounding vertex $\boldsymbol{P}$, the nodal movement is performed step by step and in the form

$$\boldsymbol{P} = \boldsymbol{P} + d\frac{\boldsymbol{n}_0}{|\boldsymbol{n}_0|^2}, \boldsymbol{n}_0 = \sum_{i=1}^{N} A_i^{-1}\boldsymbol{n}_i \; / \sum_{i=1}^{N} A_i^{-1} \tag{4.1}$$

where A_i are triangle areas and d is a small increment on the order of 0.1 mm. This method ensures topology-preserving expansion/shrinkage of a cone as shown in the following figure [17].

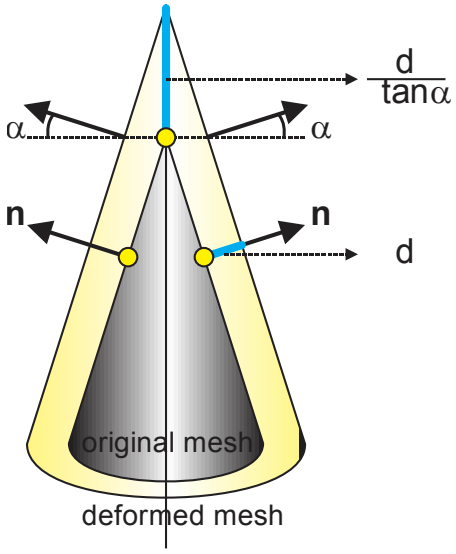

Figure 32: Local mesh shrinkage and/or expansion according to Eq. (4.1). Drawing from Ref [17]

4.2 Skin Shell

The skin shell is achieved via methods described in Chapter 2 after several iterations of mesh conditioning procedures such as segmentation, decimation, and deformation.

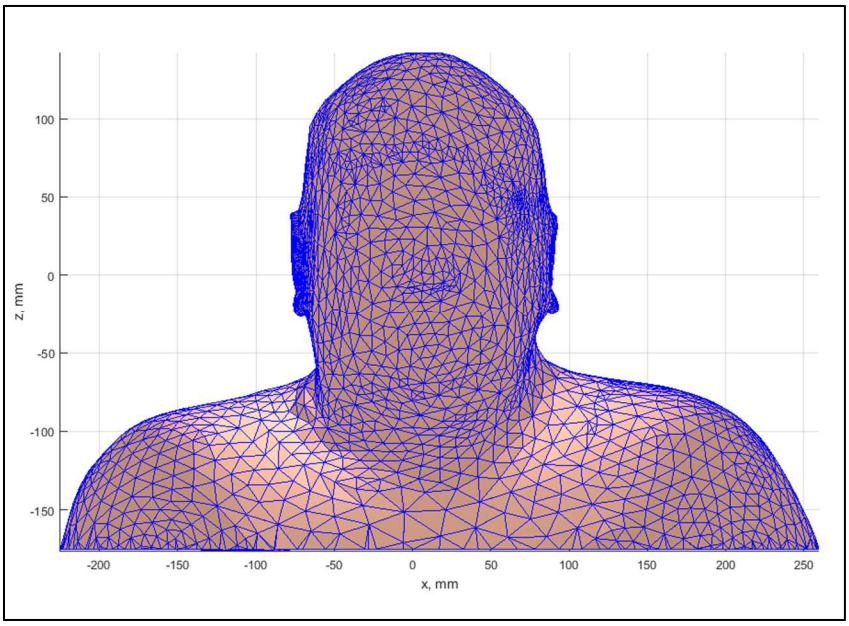

Figure 33: VHP-Female Skin Shell. Original Drawing

4.3 Fat Shell

Applying the concept of nodal movements, the skin shell is shrunken 1mm The vertices are moved inwards in the direction of the normal vector. The following figure shows the original skin shell in red and shrunken fat shell in green with a thickness of 1mm in between.

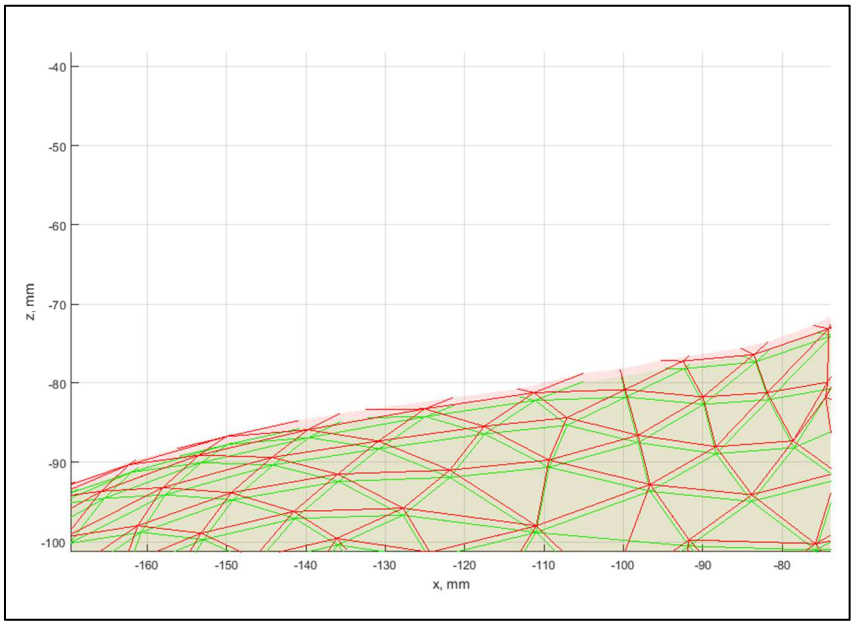

Figure 34: Skin Shell and Fat Shell with 1mm thickness. Original Drawing

4.4 Average Body Shell

Similarly, the fat shell is shrunken 2mm to create the average body shell. The following figure shows the fat shell in green and average body shell in blue with a thickness of 2mm in between.

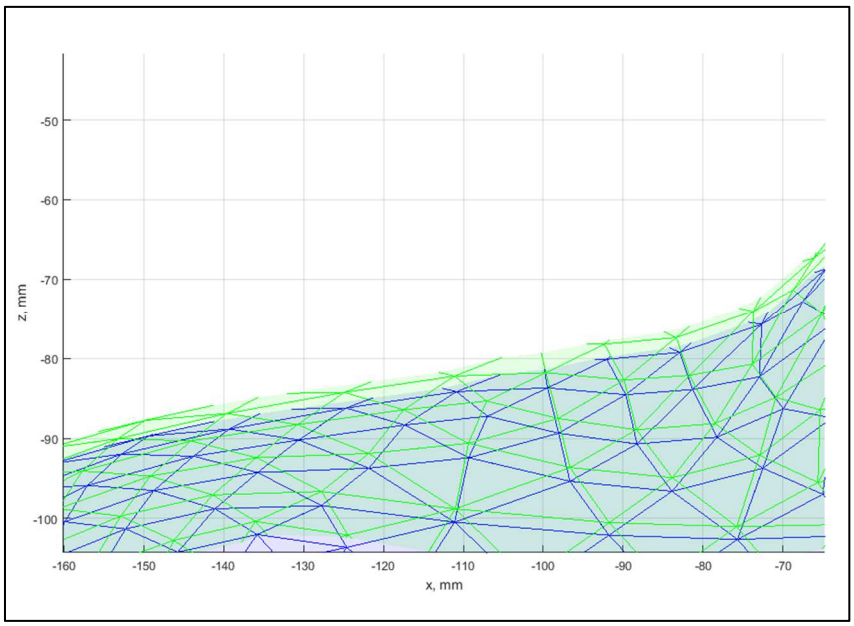

Figure 35: Fat Shell and Average Body Shell with 2mm thickness. Original Drawing

4.5 Skin, Fat, and Average Body Shells

The three layers of skin, fat and average body are combined. The following figure shows the skin, fat, and average body shells in red, green, and blue respectively.

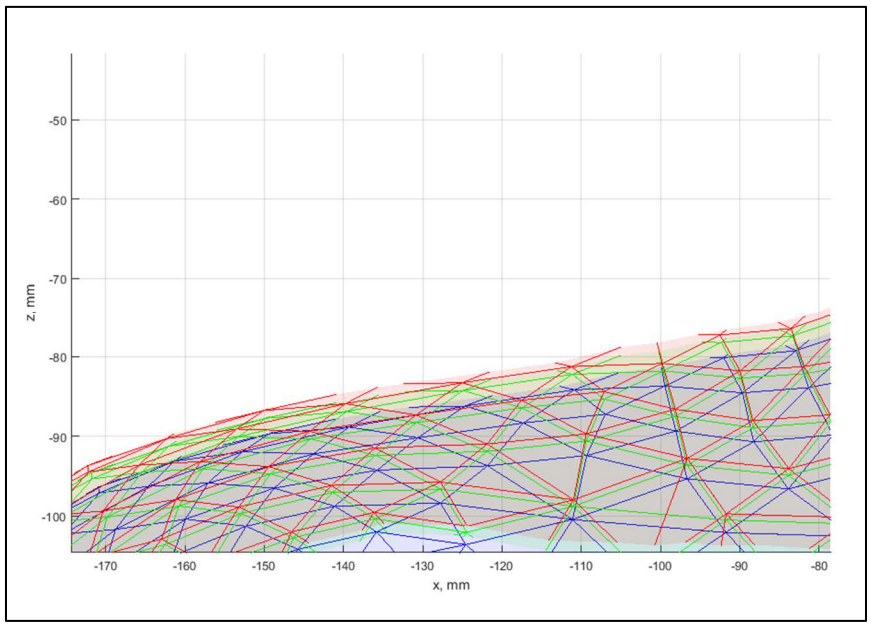

Figure 36: Skin Shell, Fat Shell and Average Body Shell. Original Drawing

References

[1]. Sezgin M, Sankur B. Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging. January 2004; 13(11): 46-165.

[2]. Yushkevich PA, Piven J, Hazlett HC, Smith RG, Ho S, Gee JC, Gerig G. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. Neuroimage. 2006; 31(3): 1116-1128.

[3]. Yushkevich PA, Gao Y, Grig G. ITK-SNAP: an interactive tool for semi-automatic segmentation of multi-modality biomedical images. 38th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC 2016), Orlando, FL, Aug. 16-20 2016.

[4]. Pham DL, Xu C, Prince JL. Current Methods in Medical Image Segmentation. Annual Review of Biomedical Engineering. August 2000; 2: 315-337. PMID: 11701515.

[5]. Zhao F, Xie X. An Overview of Interactive Medical Image Segmentation. Annals of the BMVA, 2013; 2013(7): 1–22.

[6]. Internet Analysis Tools Registry. Harvard Univ., Available: http://www.cma.mgh.harvard.edu/iatr/index.php

[7]. Young PG, Beresford-West TBH, Coward SRL, Notarberardino B, Walker B, Abdul-Aziz A. An efficient approach to converting 3D image data into highly accurate computational models. Philos Trans A Math Phys Eng Sci. 2008 September 13; 366(1878): 3155-73. PMID 18573757.

[8]. Lo SH. Automatic mesh generation over intersecting surfaces. Int. J. Numerical Methods Eng. 1995 March 30; 38(6): 943-954.

[9]. Lo SH, Wang WX. A fast robust algorithm for the intersection of triangulated surfaces. Engineering with Computers. 2004 March; 20(1): 11-21.

[10]. Elsheikh AH, Elsheikh M. A reliable triangular mesh intersection algorithm and its application in geological modelling. Engineering with Computers. 2014 January; 30(1): 143-157.

[11]. Coelho LC, Gattass M, De Figueiredo LH. Intersecting and Trimming Parametric Meshes on Finite-Element Shells. Int. J. for Numerical Methods in Engineering. 1998 October 47(4): 777-800.

[12]. Gammon M. CAD Clean-up for Meshing. What could possibly go wrong?. Short Course. 23rd Int. Meshing Roundtable, London, UK, pp. 1-70. Oct. 12-15, 2014.

[13]. Yanamadala J, Noetscher GM, Rathi V, Maliye S, Win HA, Tran AL, Jackson XJ, Htet AT, Kozlov M, Nazarian A, Louie S, Makarov S. New VHP-Female V. 2.0 Full-Body Computational Phantom and Its Performance Metrics Using FEM Simulator ANSYS HFSS. 37th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC 2015), Milano, Italy, Aug. 25-29 2015, pp. 3237-3241.

[14]. Noetscher GM, Yanamadala J, Kozlov M, Louie S, Nazarian A, Makarov S. VHP-Female v3.0 FEM/BEM Computational Human Phantom. 24th Int. Meshing Roundtable (IMR24), Austin, TX, Oct. 12-14, 2015.

[15]. Yanamadala J, Noetscher GM, Louie S, Prokop A, Kozlov M, Nazarian A, Makarov S. Multi-Purpose VHP-Female Version 3.0 Cross-Platform Computational Human Model. 10th European Conf. on Antennas and Propagation (EuCAP16), Davos, Switzerland, April 2016 pp. 1-5.

[16]. Tankaria H, Jackson XJ, Borwankar R, Srichandhru GNK, Tran AL, Yanamadala J, Noetscher GM, Nazarian A, Louie S, Makarov SN. VHP-Female Full-Body Human CAD Model for Cross-Platform FEM Simulations – Recent Development and Validations. 38th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC 2016), Orlando, FL, Aug. 16-20 2016.

[17]. Makarov SN, Noetscher GM, Yanamadala J, Piazza MW, Louie S, Prokop A, Nazarian Ara, Nummenmaa A. Virtual Human Models for Electromagnetic Studies and Their Applications

CHAPTER 5

FULL-BODY SHELLS FOR VHP-MALE CAD MODEL

5.1 Introduction

Using the mesh expansion and shrinkage methods described in Chapter 4, full-body shells of VHP-Male CAD model are created. Starting with the manually segmented VHP-Male skin shell, the mesh is shrunken 1mm and 3mm respectively to obtain the fat shell and average body shell.

The shrinkage of the shells results in various geometrical problems such as non-manifoldness and intersections. In order to be truly CAD models suitable for simulations as described in Chapter 2, these issues need to be resolved and shells need to be validated.

These geometrical problems appear not only in individual shells but also between two tightly spaced enclosed shells, especially around the regions of high definitions such as ears, fingers and toes.

These issues are resolved using ANSYS 3-D modeling tool 'SpaceClaim'. This chapter describes the procedure to resolve various geometrical errors in full-body shells, especially triangle intersections.

5.2 Mesh Healing in ANSYS SpaceClaim

ANSYS SpaceClaim features easy and intuitive manual manipulation of the vertices, edges, and triangles in the mesh. This enables effective real-time resolution of geometrical problems.

For this study, the SpaceClaim Facets Tool is heavily used to detect and resolve errors. Features most often used in the creation of VHP-Male shells are:

1. Detection and resolution of non-manifoldness
2. Detection and resolution of holes (mesh not being watertight)
3. Detection and resolution of intersections
4. Detection and resolution of multiple pieces

5.2.1 Resolving Non-manifoldness and Holes

Non-manifold edges are detected in the SpaceClaim Facets tool using 'Check Mesh' button as shown in the figure below. The areas of non-manifoldness are marked with red dots in the mesh as in the Fig. 38.

The issue can be resolved using ‘Auto Fix’ button.

Figure 37: SpaceClaim Check Mesh. Original Drawing in SpaceClaim

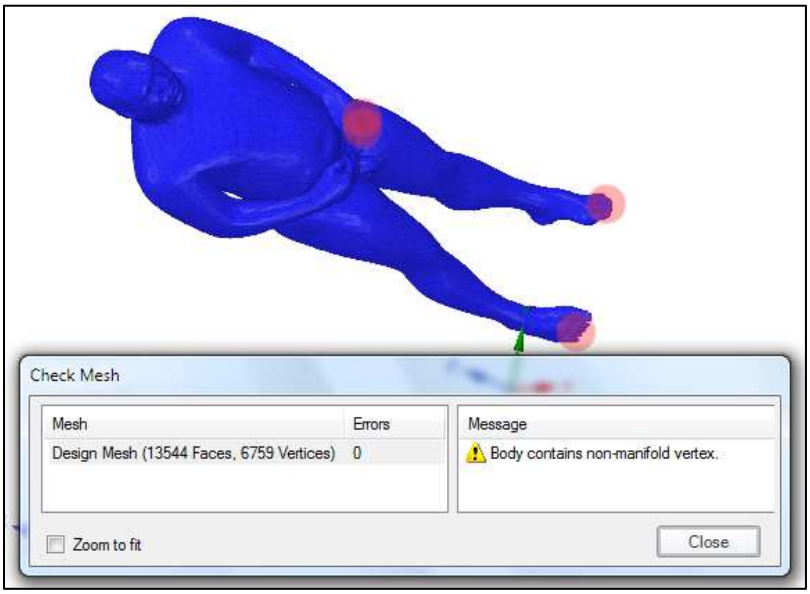

Figure 38: Non-Manifoldness Detection in SpaceClaim. Original Drawing in SpaceClaim

To manually resolve the non-manifoldness, the non-manifold areas marked in red dots are zoomed in. The triangles with non-manifold vertices are usually hidden inside the shell so the triangles on top need to be removed.

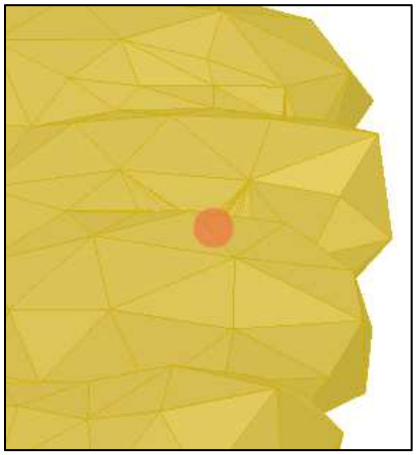

Figure 39: Marked Non-Manifold Triangle Zoomed In. Original Drawing in SpaceClaim

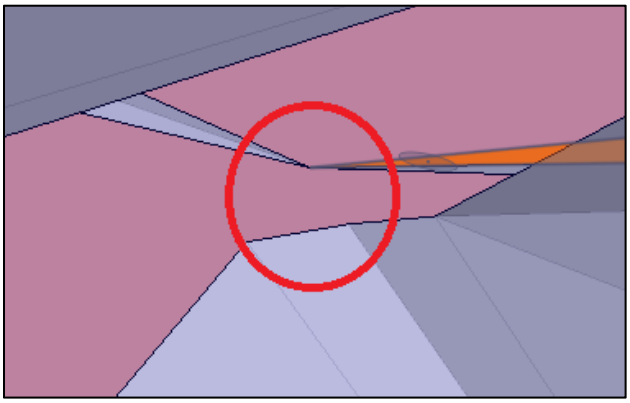

Figure 40: Non-Manifold Triangles Identified. Original Drawing in SpaceClaim

After removing the triangles, the non-manifold issue is resolved. However, the mesh becomes non-watertight since the triangles on top are also removed. This non-watertight issue is detected by 'Holes' button in Facets Tool, and fixed automatically using the green button as shown in Fig. 41.

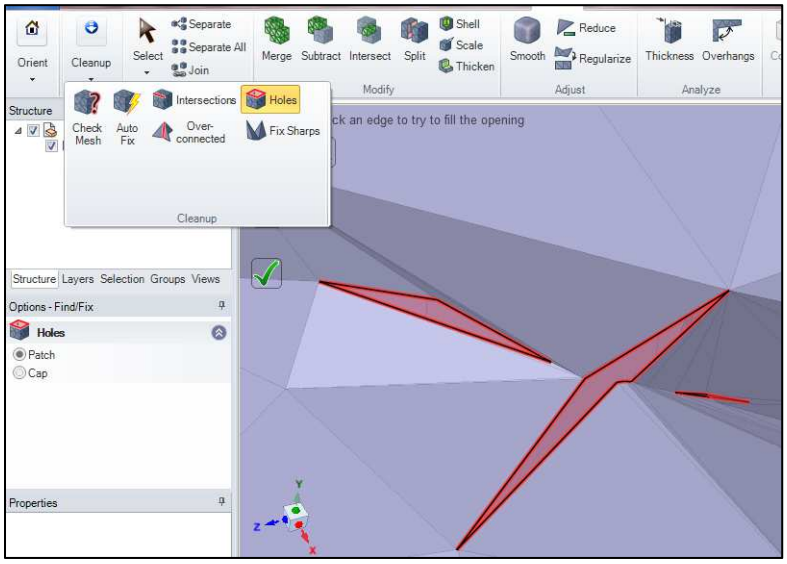

Figure 41: Non-Manifold Triangles Removed. Original Drawing in SpaceClaim

5.2.2 Resolving Self-Intersections in Individual Shells

Intersection issues are more evident in the areas of fingers, toes and groin where the mesh has significantly detailed definition and triangles are close to each other. These intersections are detected in SpaceClaim Facet tool.

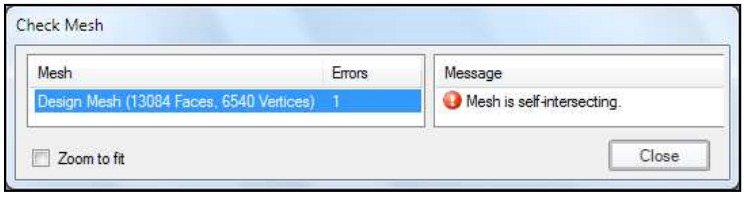

Figure 42: Intersections detected in SpaceClaim. Original Drawing in SpaceClaim

The intersection issues are visualized as red triangles in Fig. 43.

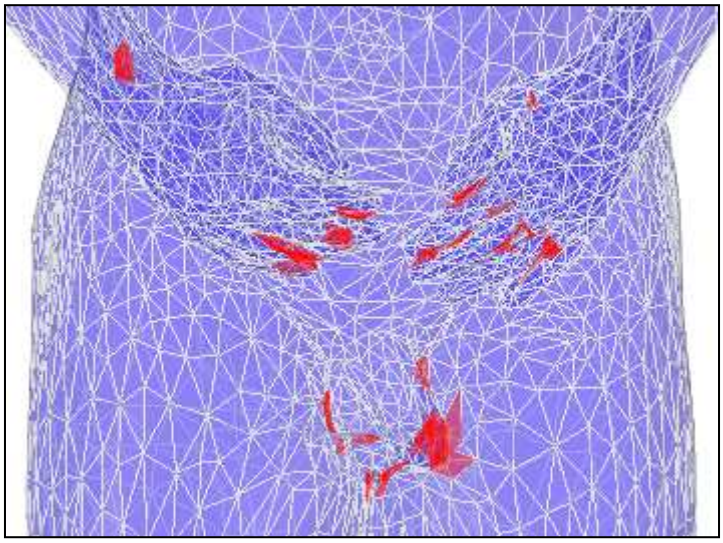

Figure 43: Intersections in Fingers and Groin Areas. Original Drawing in SpaceClaim

Figure 44: Intersections in Toes. Original Drawing in SpaceClaim

The intersection issues are resolved manually by moving vertices, edges or triangles as necessary. This is an iterative process in which moving and checking the intersections are repeated until all issues are resolved.

While moving the points, care is taken not to compromise the original structure and definition. In Fig. 45, the intersecting triangle (vertex, edge, or triangle) is selected and moved.

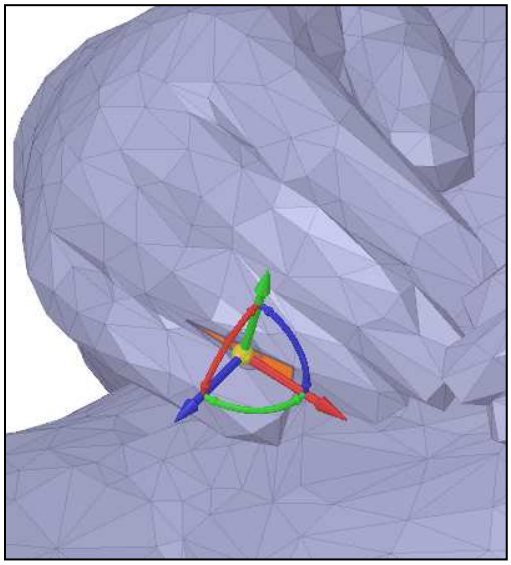

Figure 45: Resolving Intersections by Moving the Triangles. Original Drawing in SpaceClaim

When solving intersection issues, it is sometimes necessary and easier to delete the triangles and reconstruct them. To do so, triangles are selected and deleted. Then using SpaceClaim 'Holes' tool, holes are detected and healed. This results in the resolution of the intersections.

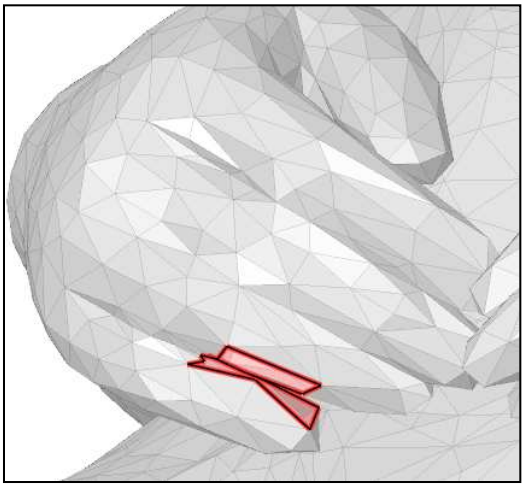

Figure 46: Resolving Intersections by Reconstruction. Original Drawing in SpaceClaim

5.2.3 Resolving Intersections among Shells

Once each individual shells have been healed so that no geometry problems are detected in SpaceClaim Facet tool, intersections among the shells need to be resolved. Since there are three shells, intersections among two adjacent layers are resolved at a time in the following order. Very few intersections should be expected at step 3.

1. Skin Shell and Fat Shell
2. Fat Shell and Average Body
3. Skin Shell and Average Body

To visualize the intersections between each pair of shells, each shell is assigned a custom color so that they are easily distinguishable.

For example, in the following figure, the inner shell is assigned bright green while the outer shell is assigned dull gray. The protruding structure of the inner green shell is easily observed.

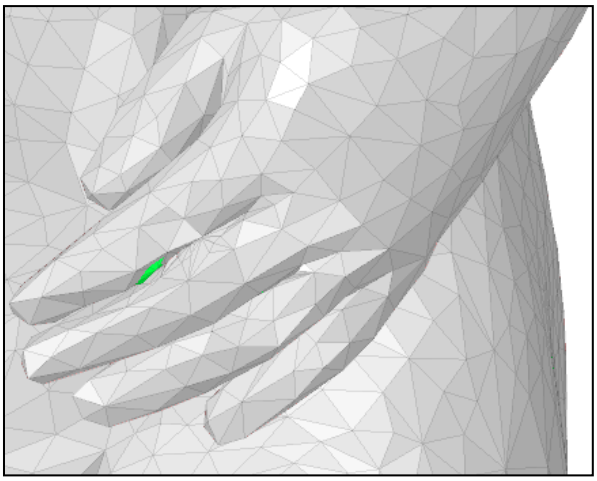

Figure 47: Visualization of Intersections between Two Shells. Original Drawing in SpaceClaim

The intersection is resolved by either moving the inner green shell inwards or moving the outer gray shell outwards without significant impact on the original topology. In Fig. 48, the vertex of the inner shell is moved inwards.

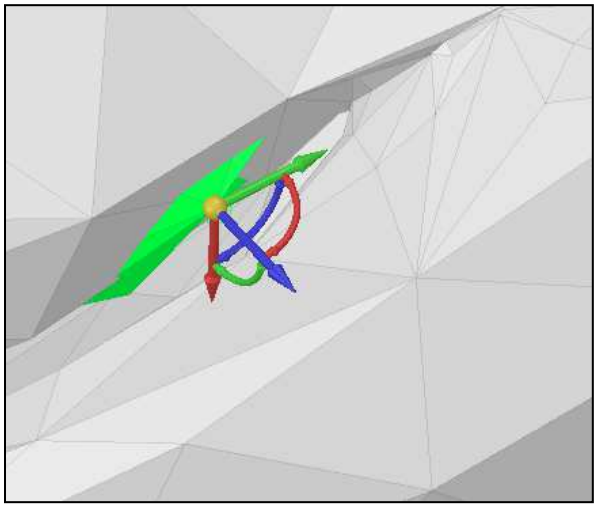

Figure 48: Resolving Intersections between Two Shells. Original Drawing in SpaceClaim

Assigning different colors for visualization is helpful in detecting intersection. However, occasionally, some intersections are too small to inspect visually. Therefore, it is necessary for a method to automatically detect all intersections.

To automate the detection of intersections between two shells, the two meshes are combined into a single mesh by 'joining' them together in SpaceClaim.

Figure 49: SpaceClaim Tools for Joining and Separating Two Shells. Original Drawing in SpaceClaim

In Fig. 50, a small protruding green triangle can be observed in the first picture. However, it is easy to miss this upon visual inspection. The second picture shows the detection after joining the two meshes together.

This automatic detection is utilized in addition to visual inspection to comprehensively validate that all intersection issues are fixed.

Once intersections are resolved, the two meshes are separated using 'Separate All' button in SpaceClaim.

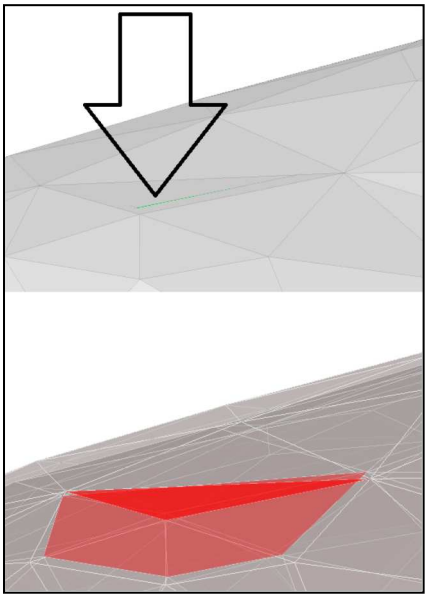

Figure 50: Intersection in Separate Meshes and Intersection Detected in Joined Mesh. Original Drawing in SpaceClaim

5.2.4 Resolving Issue of Multiple Pieces

A combination of automatic and manual resolutions of geometrical issues sometimes results in multiple pieces of mesh. However, this can be easily resolved in SpaceClaim Facets Tool.

As seen in Figure 51(a), the tool detects multiple pieces in the mesh. In Fig. 51(b), the pieces are separated into distinct objects using the 'separate' button. This results in separate 'design meshes' as in Fig. 51(c). Then the unwanted pieces are selected and removed from the design as in Fig. 51(d).

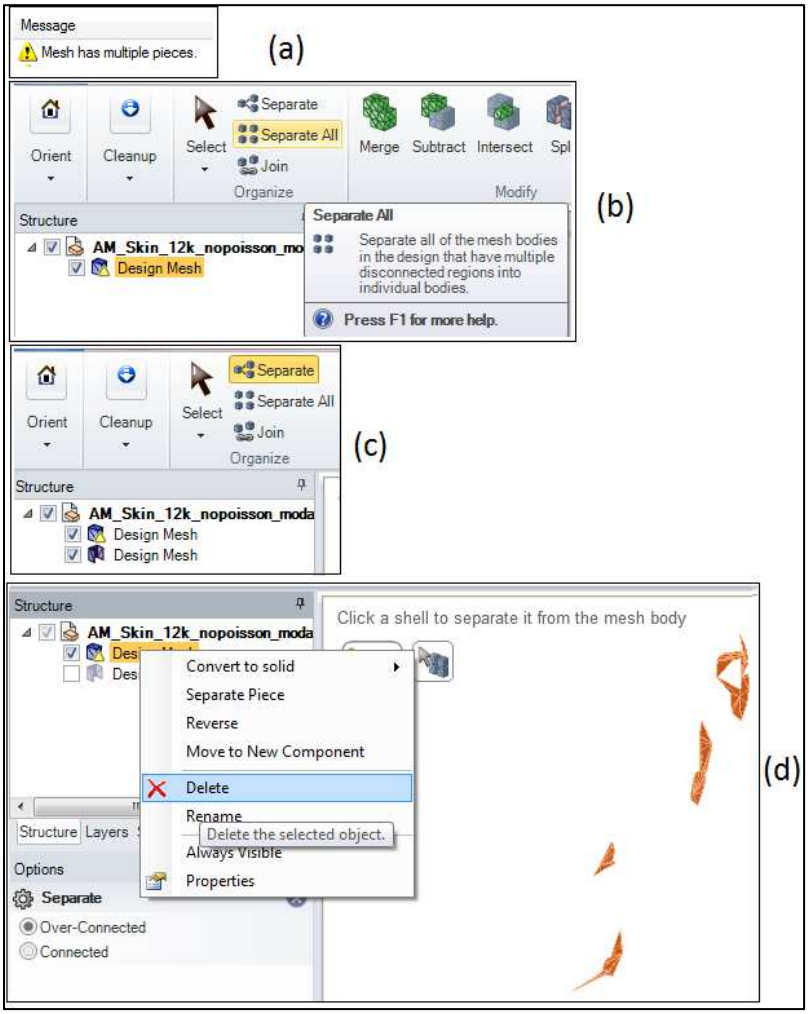

Figure 51: Resolving Multiple Pieces. Original Drawing in SpaceClaim

5.3 Separating Arms from Torso

In the original dataset, the arms are attached to the body. Realistically in MRI scans, the arms should be at a small distance from the body. To correct this, the triangles at the area where the arms are touching the body are removed. Afterwards, the edges of the arms and the body are moved further away from each other. The SpaceClaim 'Holes' tool is used to close the holes created. This results in the arms separating from the body.

Fig. 52(a) shows the right arm and the body touching each other. Fig. 52(b) shows the triangles being removed. Fig. 52(c) shows the resulting mesh, in which the arm is separated from the body.

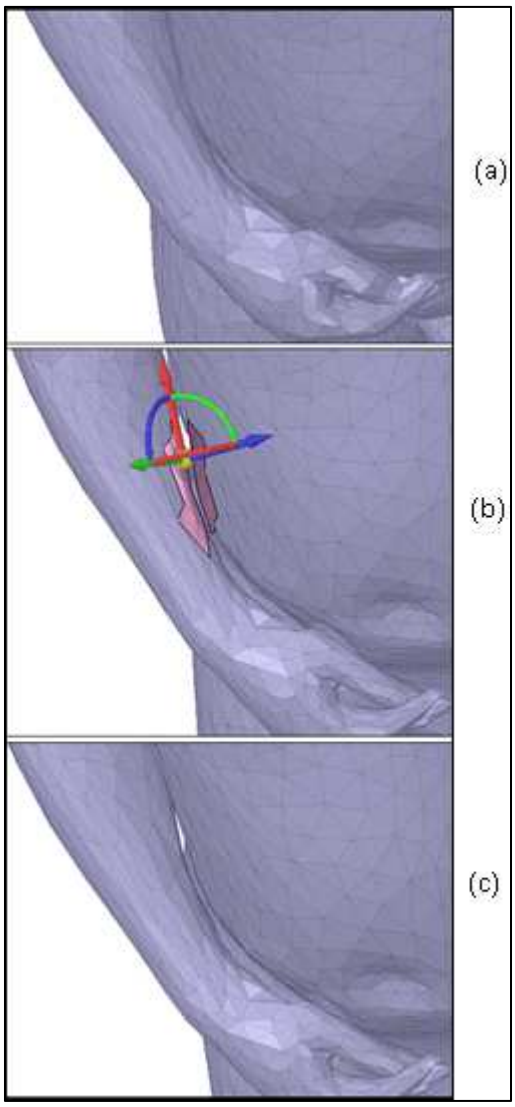

Figure 52: Separating Arms from Torso. Original Drawing in SpaceClaim

5.4 Final Shells

After all geometry problems (in individual shells and among the shells) are resolved, the final result of the three shells combined is achieved. A gap of 1 mm between skin shell and fat shell, and a gap of 2mm between fat shell and average body shell are illustrated in the following figure of the VHP-Male model hand.

The innermost shell (average body shell) is in yellow while the middle shell (fat shell) is in green and the outermost shell (skin shell) is in red.

Figure 53: Skin, Fat, and Average Body Shells of VHP-Male Hand. Original Drawing

Fig. 54 shows the full-body view of the VHP-Male body shell.

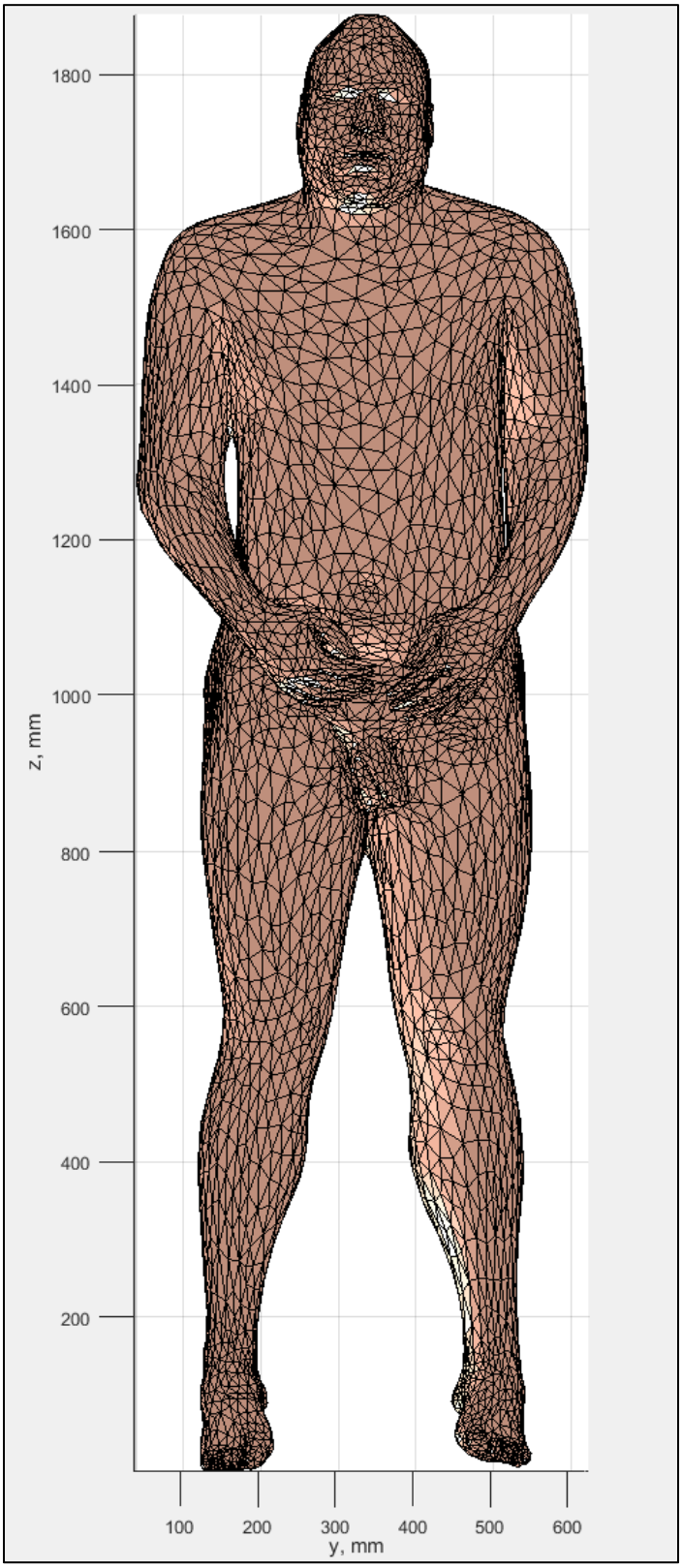

Figure 54: Full-Body Shell of VHP-Male

CHAPTER 6

FULL-BODY SHELLS FOR VHP-FEMALE CAD MODEL WITH EAR CANAL

6.1 Introduction

Tightly-spaced enclosed full-body shells are created for VHP-Female model using mesh expansion and shrinking methods described in Chapter 4. The geometrical problems are resolved using ANSYS SpaceClaim as described in Chapter 5. The thickness between any two enclosed shells are uniform throughout the full body, i.e. 1 mm between skin shell and fat shell, and 2 mm between fat shell and average body shell.

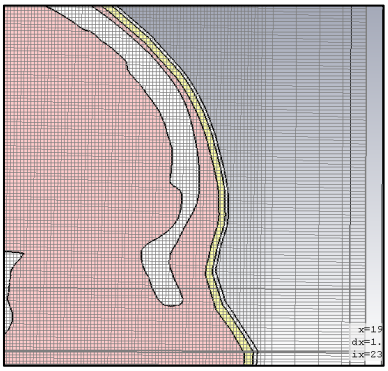

Figure 55: Skin, Fat, Average Body, and Skull. Original Drawing from VHP-Female Model

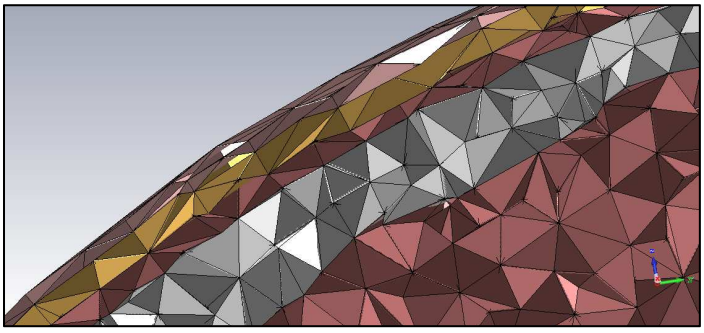

Figure 56: Close-up View of Skin, Fat, Average Body, and Skull. Original Drawing from VHP-Female Model

Uniform thickness, however, is not ideal in reality. For example, it would be unrealistic for ears/ear canals to have the same thickness of fat as the torso area since ears have much lower fat than the torso. To achieve more accurate simulations, the thickness in the ear canals are significantly reduced to a small number compared to the thickness of the rest of the body.

6.2 Segmentation of the Ear Canals

The manual segmentation result of the space defined by ear canals is shown in the Fig. 57. The segmentation result can be compared to the anatomy of the ear canal in Fig 58. The segmentation is found to accurately reflect the anatomy of the ear canal.

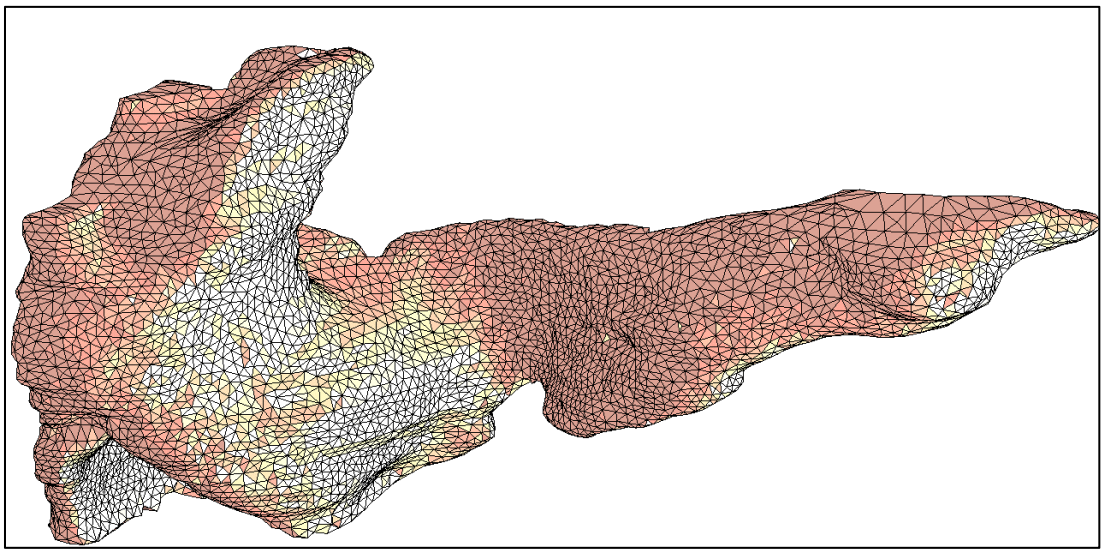

Figure 57: Direct Manual Segmentation of Space Defined by Left Ear Canal. Original Drawing from VHP-Female Model

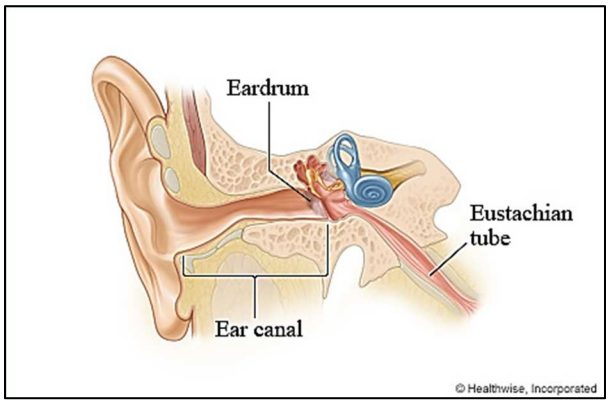

Figure 58: Anatomy of Ear Canal. Drawing from http://www.webmd.com/cold-and-flu/ear-infection/ear-canal

The following figures show the segmentation results of the entire ears – both left and right. Accurate representations of the ear canals can be observed in the dense triangle definitions inside and outside the ears.

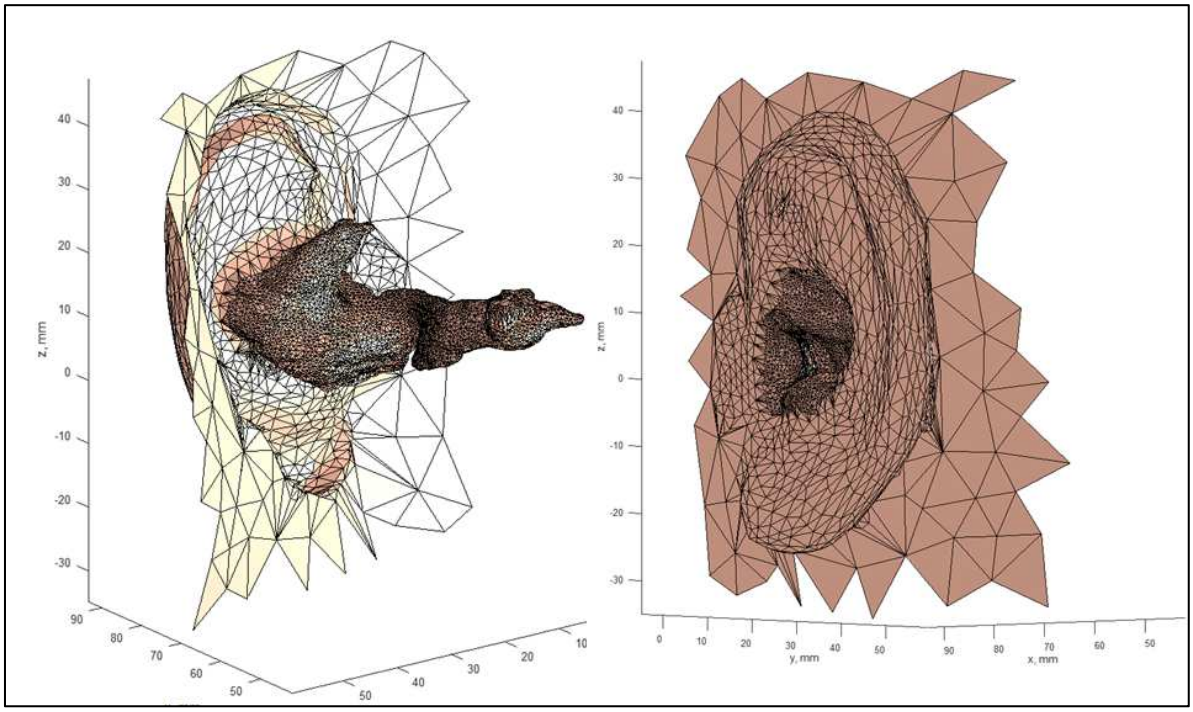

Figure 59: Segmentation Result of Left Ear. Original Drawing from VHP-Female Model

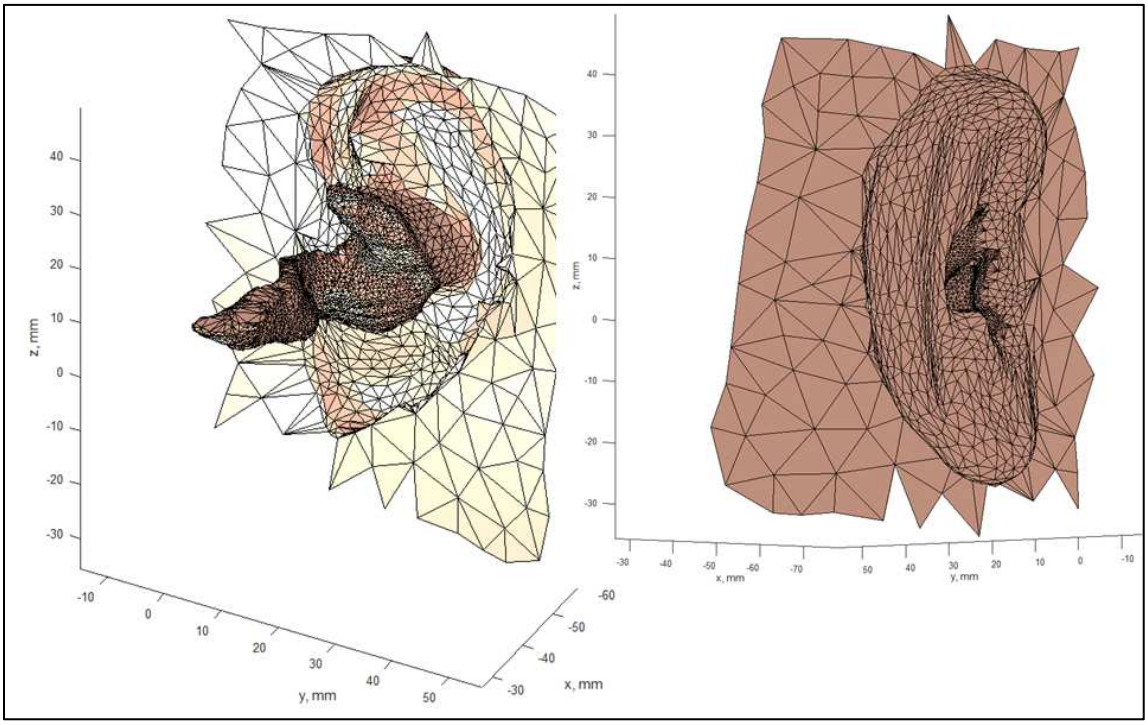

Figure 60: Segmentation Result of Right Ear. Original Drawing from VHP-Female Model

The segmented ear canals are integrated into the VHP-Female Head. Fig. 61 shows the VHP-Female Head (skin shell) with ear canals.

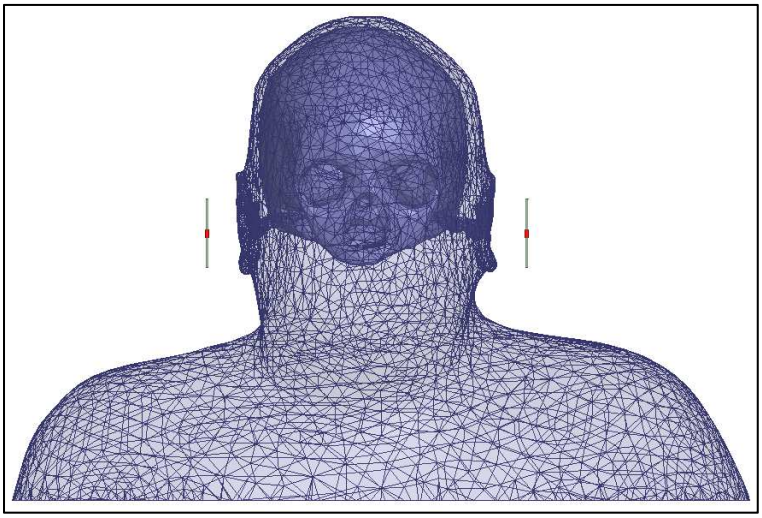

Figure 61: Female Head Model with Ear Canals. Original Drawing from VHP-Female Model

6.4 Variable Thicknesses in Female Head with Ear Canals

In VHP-Female head, in which detailed ear canals are integrated, smaller thickness than the rest of the body is realized in the ear canals.

To achieve this, four separate shells are created at thickness 0.3mm, 0.6mm, 1mm, and 3mm from the original skin shell.

Ears from 0.3mm shell is extracted and integrated in 1mm shell.

Likewise, ears from 0.6mm shell is extracted and integrated in 3mm shell.

This procedure is illustrated in the Fig. 62.

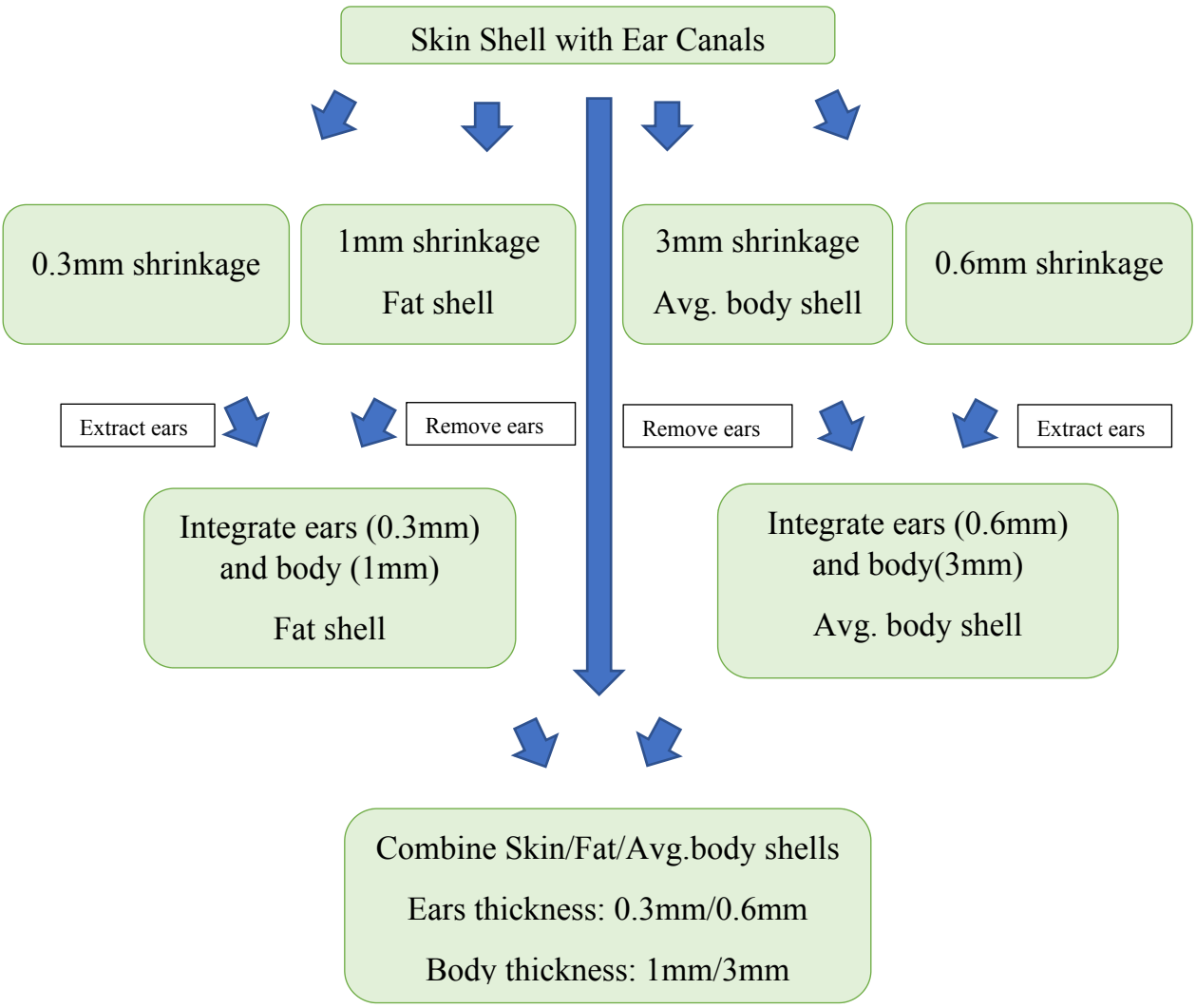

Figure 62: Stages of Creating and Integrating VHP-Female Shells with Ear Canals

6.4.1 Extraction of Ears

Ears are extracted from the (0.3mm/0.6mm) shell in MATLAB.

First, the x coordinate range, y coordinate range and z coordinate range of the ears are determined. The vertices not in the determined x/y/z ranges are deleted from the triangulation matrix afterwards.

```
% Get ears from .3mm shell
load('Skin_Cut_p3mm.mat');
P1 = P;
for i=1:length(P1)
    if (~((((P1(i,1) >= -80) && (P1(i,1) <= -20))  || ... % x right ear
           ((P1(i,1) >=  50) && (P1(i,1) <= 100))) && ... % x left ear
           ((P1(i,2) >= -30) && (P1(i,2) <=  50))  && ... % y
           ((P1(i,3) >= -40) && (P1(i,3) <=  50))))       % z
        P1(i,:) = [NaN NaN NaN];
    end
end
% Delete all rows marked 3 NaNs
P1_del_idx = sum(isnan(P1),2);
P1(P1_del_idx==3,:) = [];
```

The resulting points are reconstructed as in the following figure.

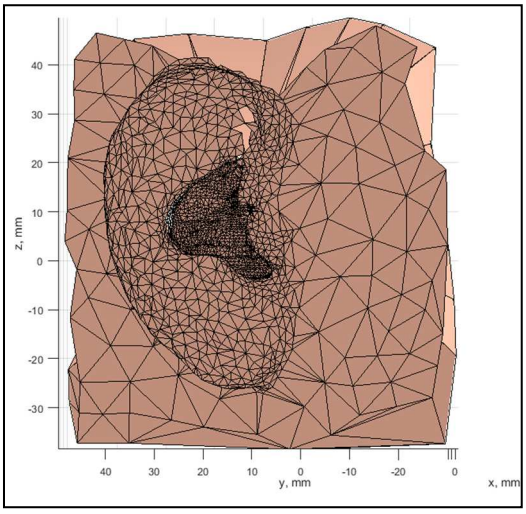

Figure 63: Extracted Ears from 0.3mm/0.6mm Shrunken Shells. Original Drawing

6.4.2 Removal of Ears

In MATLAB, the ears from 1mm/3mm expanded shells are removed by deleting the vertices in the determined x/y/z coordinate ranges.

```
% Remove ears from 1mm shell
load('Skin_Cut_1mm.mat');
P2 = P;
for i=1:length(P2)
    if ((((P2(i,1) >= -80) && (P2(i,1) <= -20))  || ... % x right ear
         ((P2(i,1) >=  50) && (P2(i,1) <= 100))) && ... % x left ear
         ((P2(i,2) >= -30) && (P2(i,2) <=  50))  && ... % y
         ((P2(i,3) >= -40) && (P2(i,3) <=  50)))        % z
        P2(i,:) = [NaN NaN NaN];
    end
end
% Delete all rows marked 3 NaNs
P2_del_idx = sum(isnan(P2),2);
P2(P2_del_idx==3,:) = [];
```

When reconstructed, the resulting points create the following mesh without ears.

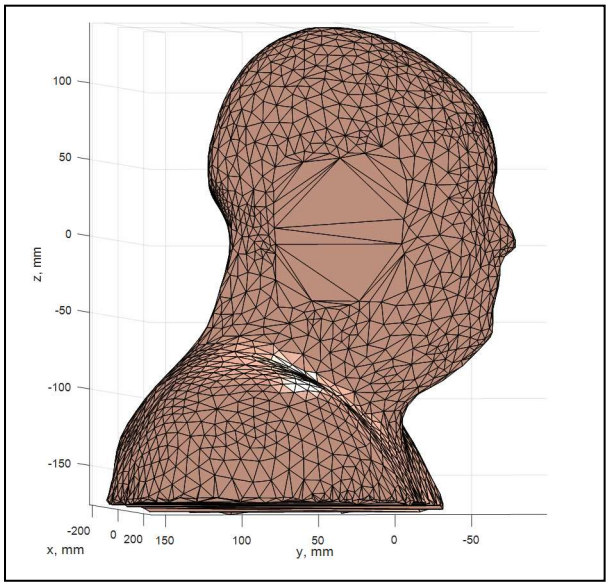

Figure 64: 1mm/3mm Shell with Ears Removed. Original Drawing

6.4.3 Integrating Extracted Ears into Shell with Removed Ears

The vertices of the shell with removed ears are appended to the vertices of the extracted ears. The new set of vertices are re-triangulated to obtain the mesh in Fig. 65 where triangles are in non-uniform orientation. The triangles are re-oriented uniformly to obtain the mesh in Fig. 66.

```
% Combine extracted .3mm ears with 1mm shell
Pnew = [P1; P2];
[tnew,tnew_norm]=MyRobustCrust(Pnew);
```

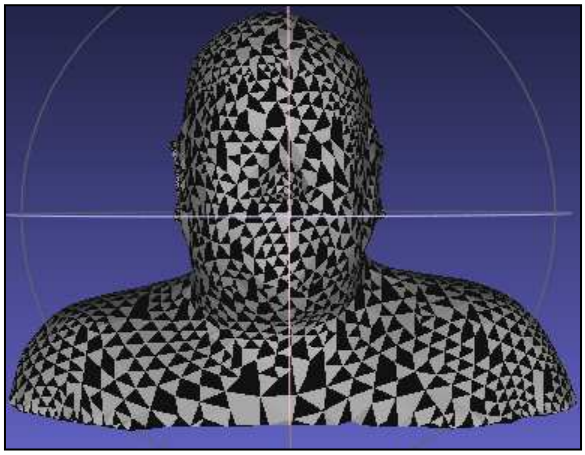

Figure 65: Re-triangulated Combined Shell. Original Drawing

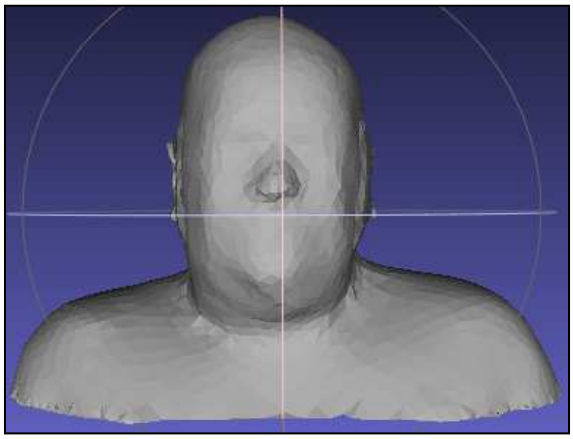

Figure 66: Combined Shell with Re-oriented Triangles. Original Drawing

6.4.4 Healing Meshes

Before proceeding to resolving intersections in SpaceClaim, the meshes are imported into a mesh processing software MeshMixer and healed using Inspector Tool (AutoRepair All).

Afterwards, the meshes are imported into SpaceClaim and re-checked for geometry problems. Intersection issues are resolved as described in Chapter 5.

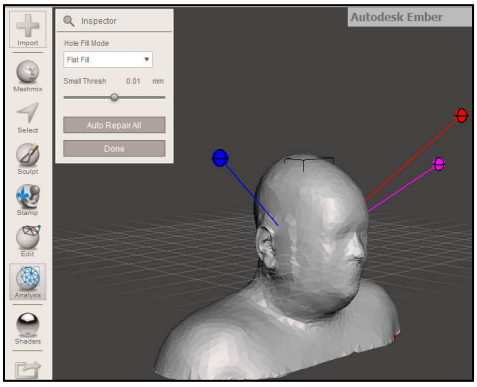

Figure 67: Mesh Healing in MeshMixer. Original Drawing in MeshMixer

6.4.5 Final Shells with Variable Thicknesses

After all geometry issues are resolved, including intersections in individual shells and among shells, the following final three shells are obtained.

As seen in the figure, the ears have thickness of 0.3mm and 0.6mm from the outermost shell while the rest of the body has thickness of 1mm and 3mm from the outermost shell.

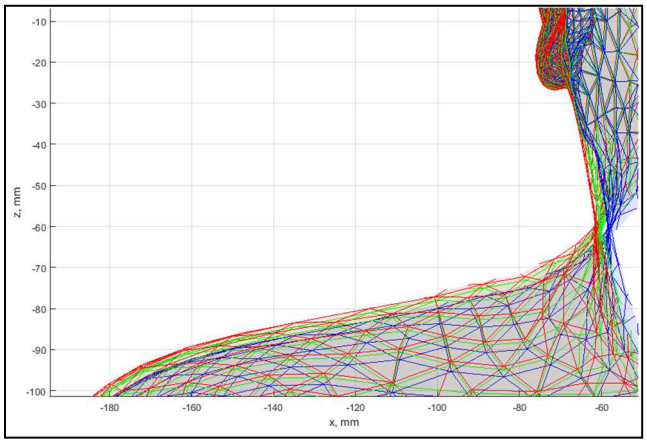

Figure 68: Shells with Variable Thicknesses. Original Drawing

Fig. 69 shows the cross-sectional view of the VHP-Female Head Model with ear canals integrated with other organs such as skull, gray matter, white matter, etc.

The thickness among shells is much less in the ear canals than the thickness in other parts of the head.

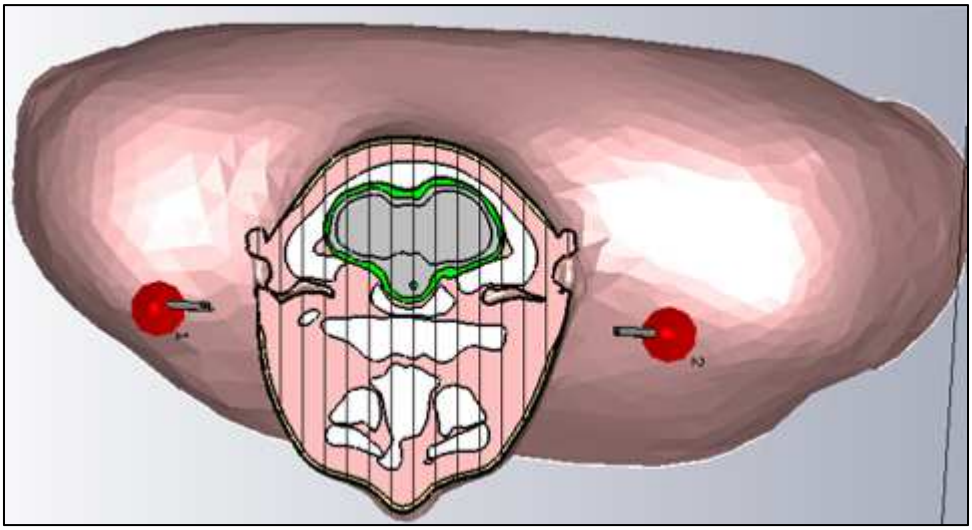

Figure 69: Cross Sectional View of Female Head with Ear Canals. Original Drawing from VHP-Female Model

Fig. 70 shows the full-body view of the VHP-Female shell.

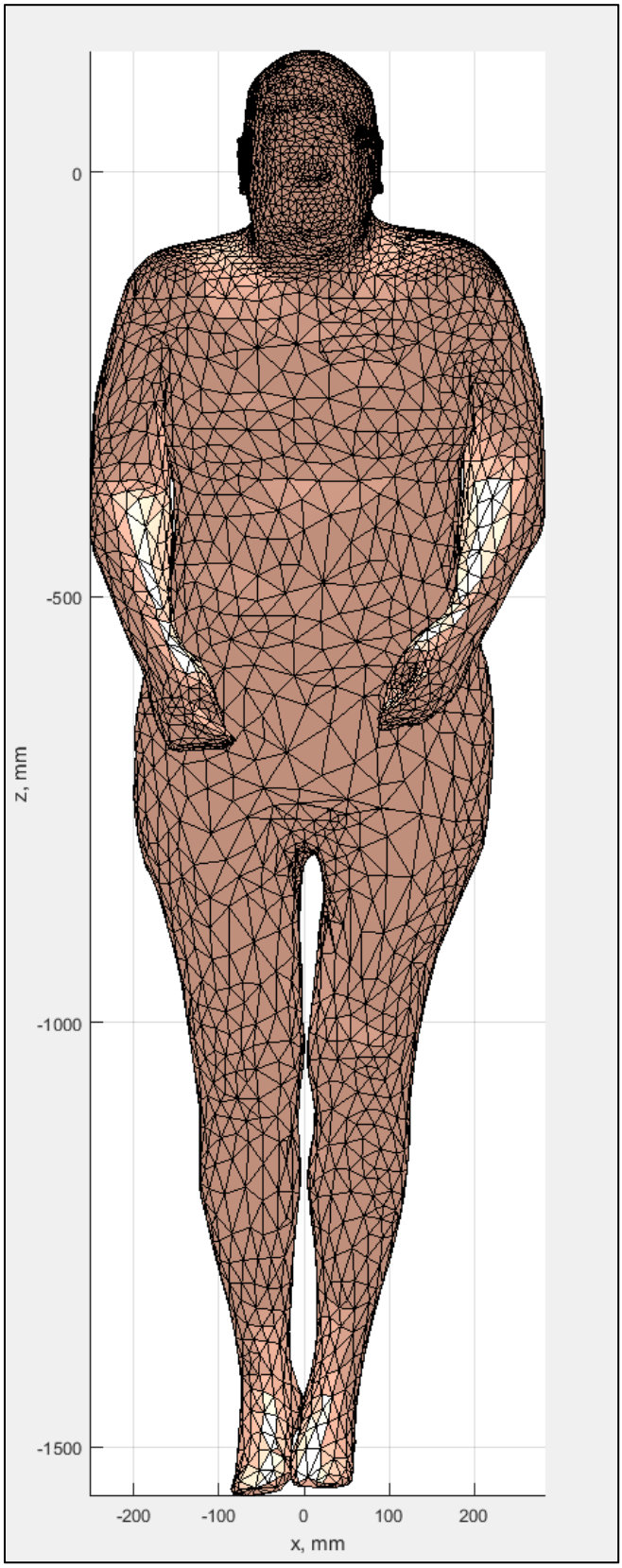

Figure 70: Full-body Shell of VHP-Female

CHAPTER 7

DISCUSSION AND CONCLUSIONS

This study presents the tools and methodologies for segmentation, processing and healing of meshes from various geometrical problems. Methods of expansion/shrinking full-body shells to create tightly-spaced shells are also described. The shrinkage methods are applied to create layers of enclosed full-body shells – skin shell, fat shell, and average body shell - for VHP-Male and VHP-Female models. Furthermore, procedures to create and integrate highly accurate ear canals to the VHP-Female model are presented. Variable thickness in a single shell has been achieved in VHP-Female model, where ears have significantly smaller thickness than the head.

In addition to full-body shell creations, an algorithm to decimate meshes based on surface curvature, along with local Laplacian smoothing, is developed. The algorithm is tested on VHP-Male bladder and is proven to attain the goal of decimating meshes while retaining definitions on surfaces of higher curvature.

Appendix A: Decimation Based on Surface Curvature

The following is the script to perform decimation based on surface curvature.

```
% Curvature decimation
%
% Description:
%     Mesh decimation based on curvature
%     and do local laplacian smoothing at decimated area
%
% Aung Htet
%
%

% Clear
clear all; clc;

% Load mesh
[FileName, PathName] = uigetfile('*.mat','Select the mesh file');
load(FileName);

% Save original P, t
P0=P; t0=t;
%---------------------------------------
% user parameters
num_iter  = 25;  % one edge is collapsed per iteration
lp_smooth = 1;   % do laplacian smoothing or not
lp_alpha  = 0.8; % laplacian weigh value
%---------------------------------------

deleted = [];
for itr=1:num_iter
    % Print progress
    fprintf('\n');
    disp (['Iteration ', num2str(itr), ' out of ' , num2str(num_iter)]);
    disp (['Current number of faces ', num2str(length(t))]);

    % compute normals
    normals = meshnormals(P, t, 0);

    % find neighbors of each triangle
    N_T = cell(length(t),1);
    for i=1:length(t)
        % neighbor triangles of each triangle
        n_t = [];

        % triangles attached to first edge
        [a b]=find(t==t(i,1));
        [c d]=find(t==t(i,2));
        n_t = [n_t;intersect(a,c)];

        % triangles attached to second edge
        [a b]=find(t==t(i,2));
        [c d]=find(t==t(i,3));
        n_t = [n_t;intersect(a,c)];

        % triangles attached to third edge
        [a b]=find(t==t(i,1));
        [c d]=find(t==t(i,3));
        n_t = [n_t;intersect(a,c)];

        % keep neighboring triangles of each triangle
        N_T{i}=(unique(n_t))';
    end
```

```
% find average angle of each triangle w.r.t neighboring triangles
avg_angs = zeros(length(N_T),1);
for i=1:length(N_T)
    avg_ang = 0;
    % add angles between all neighboring triangles
    for j=1:length(N_T{i})
        avg_ang = avg_ang + subspace(normals(i,:)',normals(N_T{i}(j),:)');
    end
    % compute average
    if (avg_ang ~= 0)
        avg_ang = avg_ang/j;
    end
    avg_angs(i) = avg_ang;
end

% save original angles before decimation
if (itr==1)
    avg_angs0 = avg_angs;
end

% triangle with smallest average angle w.r.t neighboring triangles
[minA, minA_i] = min(avg_angs);

% compute lengths of edges of smaller angle triangle
s_t = t(minA_i,:);
d1 = sqrt(sum((P(s_t(1),:)-P(s_t(2),:)).^2));
d2 = sqrt(sum((P(s_t(2),:)-P(s_t(3),:)).^2));
d3 = sqrt(sum((P(s_t(3),:)-P(s_t(1),:)).^2));
% get the smallest edge
[minD, minD_i] = min([d1;d2;d3]);
if (minD_i==1)
    s_e = [s_t(1), s_t(2)];
elseif (minD_i==2)
    s_e = [s_t(2), s_t(3)];
else
    s_e = [s_t(3), s_t(1)];
end

% merge 2 vertices of the smallest edge together
t(t==s_e(1)) = s_e(2);
del_P = s_e(1);

% find attached nodes to the merged node
lp_nodes = find_att_nodes(t, s_e(2));
tmp_lp_nodes = lp_nodes;
for i=1:length(tmp_lp_nodes)
    lp_nodes = [lp_nodes, find_att_nodes(t, tmp_lp_nodes(i))];
end

% search faces with the 2 same vertices and delete
tmp = (diff(sort(t,2),1,2)==0);
del_tris = sum(tmp,2);
t(del_tris>0,:)= [];
avg_angs(del_tris>0,:)= [];

% Local laplacian smoothing at the nodes attached to deleted node
if (lp_smooth)
    lp_nodes = [del_P, unique(lp_nodes(:)')];
    P = meshlaplace3Dlumped(P, t, lp_nodes, lp_alpha);
end

% plot before and after
if (itr==1) || (itr==num_iter)
    if (itr==1)
        figure;
        subplot(1,2,1); title('BEFORE');
        P_pl=P0; t_pl=t0; avg_angs_pl = avg_angs0;
```

```
        else
            subplot(1,2,2); title('AFTER');
            P_pl=P; t_pl=t; avg_angs_pl = avg_angs;
        end
        FV = struct('vertices', P_pl, 'faces', t_pl);
        patch(FV,'FaceColor','interp',...
            'FaceVertexCData',avg_angs_pl/(max(avg_angs_pl)-min(avg_angs_pl)),...
            'FaceColor', 'flat',...
            'edgecolor',[0 0 0]);
        axis equal; view(3)
    end

end
```

Appendix B: Segmentation Result of VHP-Female Intestine

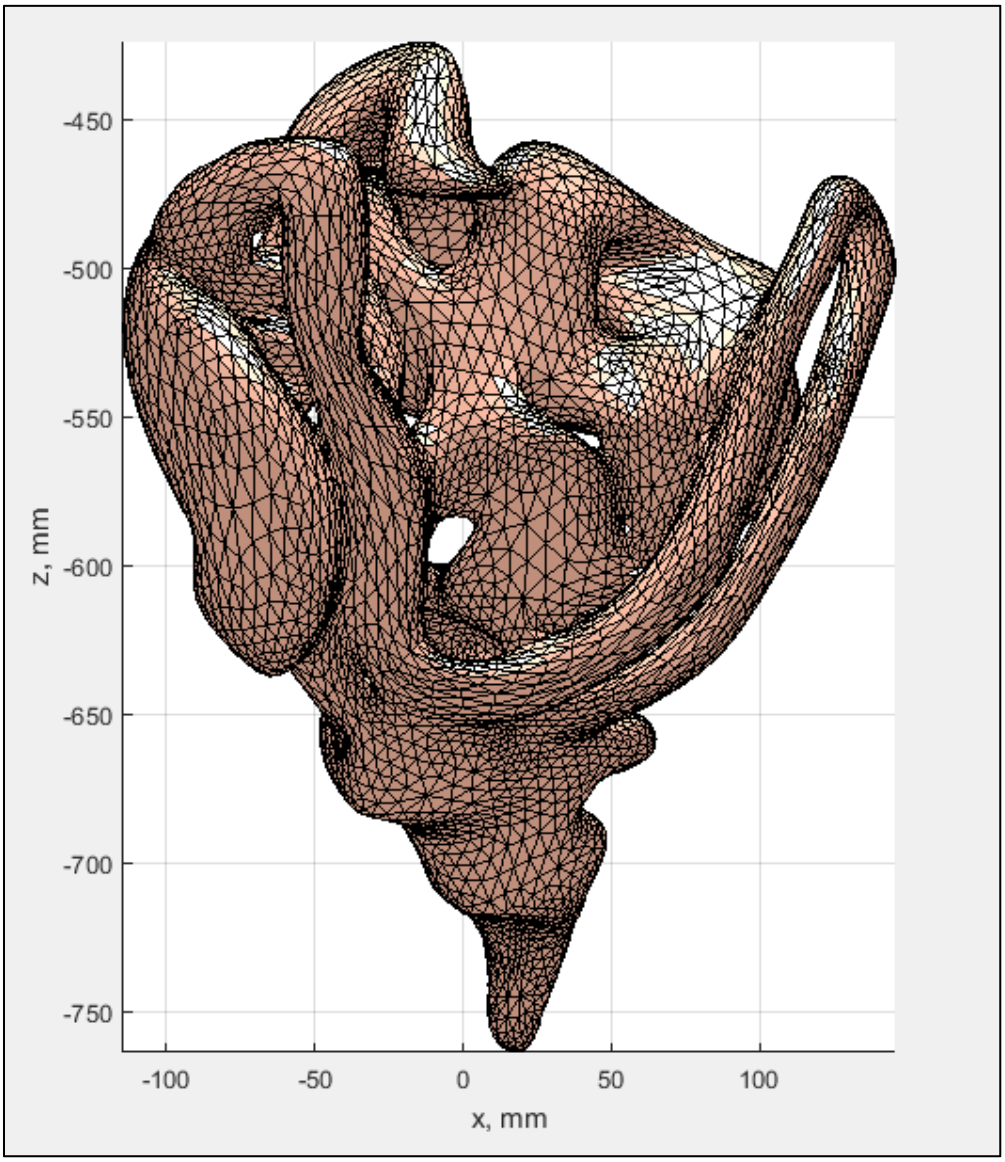

Figure 71: Segmentation Result of VHP-Female Intestine (View1)

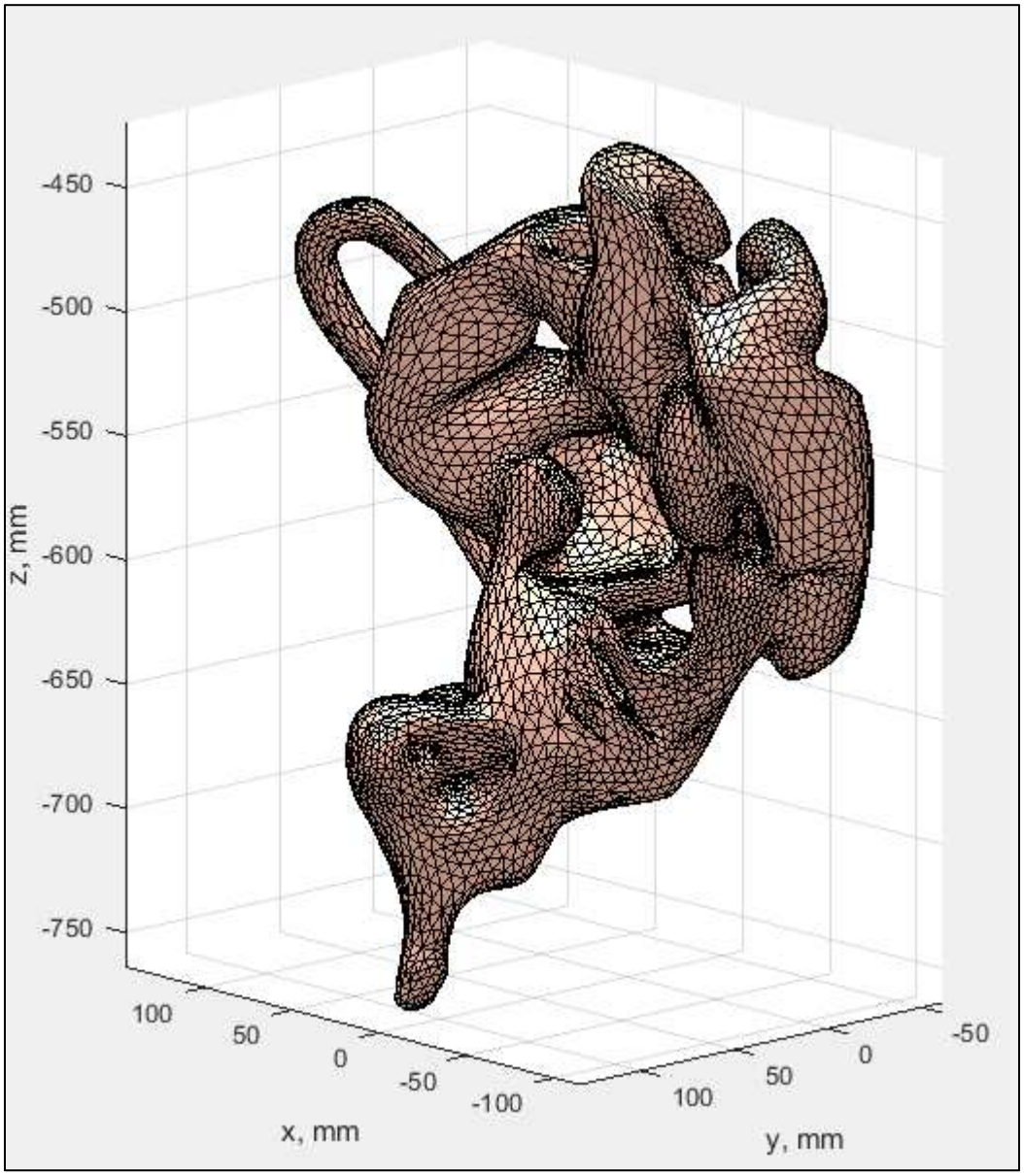

Figure 72: Segmentation Result of VHP-Female Intestine (View2)