UNIVERSITY OF GRONINGEN

BACHELOR PROJECT THESIS

Mesh Simplification and Smoothing for Bone Fragment Models

Author:
Hugo BRANDÃO

Supervisors:
Jiří KOSINKA
Peter van OOIJEN
MaxWITJES

Scientific Visualization and Computer Graphics
Bernoulli Institute
Faculty of Science and Engineering

August 20, 2018

UNIVERSITY OF GRONINGEN

Abstract

Faculty of Science and Engineering
Scientific Visualization and Computer Graphics

Computer Science

Mesh Simplification and Smoothing for Bone Fragment Models

by Hugo BRANDÃO

The problem of simplifying a mesh while preserving as many features as possible is known and well studied. However, no general implementation that raises above the others exists, and therefore case-specific implementations are usually better. In this thesis we detail an algorithm for simplifying triangular meshes of bones, focusing on the preservation of feature areas depending on the bone, with user specified parameters. We present a tuned algorithm based in previous works and an implementation of that algorithm as part of a bigger pipeline of an UMCG project. The solution takes into account several factors such as Normal Field Deviation in order to succeed, and the results obtained show very good performance and similarity to the originals.

Acknowledgements

I would first like to thank my main advisor Jiří Kosinka of the Scientific Visualization and Computer Graphics group at the University of Groningen. My endless barrage of emails, questions and need of feedback did not make his advice less objective and helpful, being patient and nice every time he could. He consistently allowed this thesis to be my own, but advised me in the right the direction whenever he thought I needed it.

I would also like to thank the medical experts and supervisors who were involved in the validation of the results of the implementation: Dr. Peter van Ooijen, Dr. Max Witjes and Dr. Joep Kraeima.

Finally, I must express my very profound gratitude to my parents and to my family and friends, both in Portugal and in the Netherlands for providing me with unwavering support and continuous encouragement throughout my years of study and through the process of researching, developing and writing of this thesis. This would not have been possible without them. Thank you.

Contents

Abstract **iii**

Acknowledgements **v**

1 Introduction **1**
1.1 Introduction 1
1.1.1 Related Work 1

2 Concept **5**
2.1 Objectives of the Algorithm 5
2.2 Definitions 5
2.3 Calculation of the Cost of an Iteration 6
2.3.1 Edge Cost 6
2.3.2 Vertex Cost 7
2.3.3 Calculation of Position of New Point 7
2.4 Overview of the Algorithm 8
2.4.1 Calculation of Current Error 8

3 Realization **9**
3.1 Implementation 9
3.1.1 Dependencies and Frameworks 9
3.1.2 Requirements 9
3.1.3 Pipeline 10

4 Results **13**
4.1 Maxilla 13
4.2 Frontal Bone 15

5 Conclusion **17**
5.1 Evaluation 17
5.1.1 Compared against General Purpose Algorithm 17
5.2 Future Work 19

List of Figures

1.1 Edge Collapse Method 2

2.1 Normals of two different sets of triangles 6

3.1 Diagram of the pipeline by Laura Baakman 11

4.1 (A) original mesh; (B) 65000 faces; (C) 40000 faces (D) 10000 faces . . . 14
4.2 (A) original mesh; (B) 65000 faces; (C) 40000 faces (D) 10000 faces . . . 15

5.1 (A) Our implementation; (B) General Algorithm; 65000 faces 18
5.2 (A) Our implementation; (B) General Algorithm; 40000 faces 18
5.3 (A) Our implementation; (B) General Algorithm; 10000 faces 19

List of Tables

4.1 Results of the Maxilla Simplification with the Number of Faces as a Restriction 13
4.2 Results of the Maxilla Simplification with the Error Threshold as a Restriction 14
4.3 Results of the Frontal Bone Simplification with the Number of Faces as a Restriction 15
4.4 Results of the Frontal Bone Simplification with the Error Threshold as a Restriction 16

5.1 Comparison over 65000 faces of the Maxilla 17
5.2 Comparison over 40000 faces of the Maxilla 18
5.3 Comparison over 10000 faces of the Maxilla 19

Chapter 1

Introduction

1.1 Introduction

The advancements in the digital world have been progressively bigger, faster and better as time passed. One challenge is the integration of such advancements and technologies into other fields, for example, medicine.

One particular way of encoding data, or represent a real world object is a 3D mesh. A mesh is a network of vertices that form a surface with a particular shape and the connections between the vertices are usually coherently formed, or in other words, they connect in the same way throughout the mesh. In this particular case, we are focusing on triangular meshes.

Today we can analyze CT scans and encode its data in a 3D bone model, namely a triangular mesh, which can be used to better diagnose the patient. When reading seed points in CT scans and converting them to a mesh, the level of detail of it is usually very high, but although detail is important, the mesh is just too dense or heavy to be used in real time. Operation on the mesh take too long to be usable and cannot be used. For that reason we created an algorithm to simplify the original bone model while preserving the needed features from a clinical and geometric viewpoint.

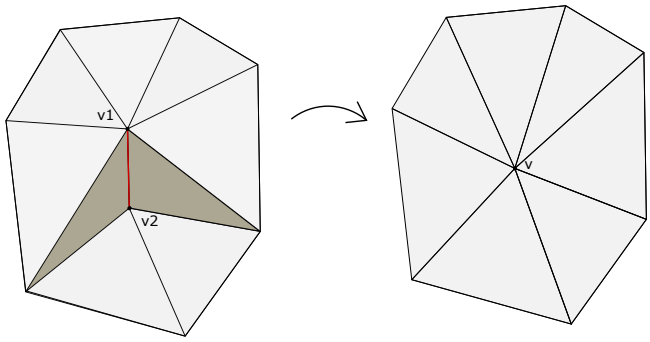
From a geometrical point of view, we want the model that is close to it defined by some metric that measures exactly what we want to turn into a measurement for quality and faithfulness. Because we want to have the simplified mesh as close to the original as possible, we use Hausdorff Distance as detailed in Cignoni, Rocchini, and Scopigno [2]. We use this measurement between the simplified mesh and the original to compute the Average Distance and the Root Mean Square (RMS). These two values are later explained in further detail in Chapter 4.

From a medical standpoint, there is no need to have a high level of detail in most zones, there is only a need to have it in the features area, such as fractures or highly important fitting zones for a prosthetic, and preserve the overall topology and geometry with a margin, usually less than a millimeter.

1.1.1 Related Work

The topic of mesh smoothing and simplification is a known and well researched area of Geometric Processing and Computer Graphics. There is not, however, a general implementation that can be considered "better than the rest", as different algorithms produce results with different advantages and disadvantages, preserving different features in different ways. We only discuss 3 specific instances of Quadric Edge Collapse methods, but more methods can be found in a more thorough research of Cignoni, Montani, and Scopigno [1] and Kaick and Pedrini [6]. We also use Edge Collapse, but methods such as Yuanfeng Zhou [12] use triangle decimation and have

FIGURE 1.1: Edge Collapse Method



found to have good results, but the computation seemed too slow for the purposes of our algorithm.

The way of Quadric Error Metrics (QEM) based simplification was introduced by Garland and Heckbert [3], proposing a fast and intuitive way to simplify polygon meshes. They proposed a vertex pair collapse, where two vertexes would join to form a new vertex in a new position that minimizes error. To choose the vertices to collapse, they associated the notion of *cost* to the vertices and their corresponding contractions with a 4x4 symmetric Matrix, called a Quadric. It aggregated the error of removing a vertex v from the mesh as the sum of the squared distances to the planes of v, that is, to the different planes of the triangles that contain that vertex.

Furthermore, they did not limit the vertex pair to a pair created by an edge of the model, making in some cases the simplified mesh lose some not filled spaces. This, as it implies making triangles appear where there were none before, is a big downside from the medical viewpoint, as we do not want to see bone where previously there was none, or even make fractures disappear completely from view. Despite this, they defined a symmetric matrix that encapsulates the error of each vertex as a distance from the plane of the area-average of the indent faces of that vertex.

Wei and Lou [9] went a step further on the above implementation, proposing a Feature Sensitive dimension, where a point:

$$p = (x, y, z)$$

with a normal N, becomes:

$$p' = (x, y, z, wNx, wNy, wNz)$$

here w is a non-negative number indicating feature sensitivity.

This approach produces some very faithful results and a preserving of sharp details that showcases very good results. However, due to its amount of attention for detail, the algorithm is slower than most, and due to our objective being the creation of an algorithm than can be used in a real time implementation, it cannot be used to the fullest extent. The idea of using a parameter to indicate feature sensitivity is used in this thesis, differencing itself by not being applied to the point, but rather to a computation better described in Chapter 2.

Hussain [4] described an implementation based on QEM, but instead of globally measuring the distance to the plane of the vertex and storing it in a 4x4 symmetric matrix, he computed it locally, reducing the computation time significantly. The downside to this is that the results are not as faithful to the original compared to Wei and Lou [9]. This particular method allowed us to tune the algorithm in such a way that we can simplify the bone mesh with speed and accuracy, maintaining its

relevance in this context of implementing it for a real time program to be used by medical experts.

For more methods not based on Quadric Error Metric Hussain, Grahn, and Persson [5] and Xing and Hui [10] are recommended.

Chapter 2

Concept

In this section we explain how the algorithm works and the geometry/parameters behind it, taking into account several aspects and requirements for this context.

2.1 Objectives of the Algorithm

We need our algorithm to preserve the features of the original mesh in such a way that it simplifies less relevant areas and preserves the geometry in the ones with more relevance. Also, this needs to be done in a relatively speedy way, as its intent is to be integrated in a program to be used in pre-operation planning. However, there is no way of choosing exactly what areas to simplify specifically without segmenting the mesh in some fashion, which would raise the execution time, and therefore we must create a parameter that differentiates areas with more relevance from the ones without it.

There is also a constraint of not being able to pass a certain given threshold in millimeters. As it is going to be used in practice, this threshold should be given by the medical expert using the algorithm, because the amount of error we can have in a given simplification changes from bone to bone and from patient to patient. User given parameters were already developed in Kho and Garland [7] and Wei and Lou [9], both showing good results in comparison with no user given input methods.

The novelty of this algorithm comes in the form of specifically being tuned for bone fragment models. Yao and Taylor [11] introduced a Laplacian-based smoothing to bone models generated from CT scans, but the features preserved were few.

2.2 Definitions

For the sake of simplicity and organisation we use the following definitions:

- A vector is a 3x1 matrix.
- An edge e of the mesh is a line segment bounded by two vertices, $v1$ and $v2$, and has two triangles incident on it, that we will call *twin triangles*.
- The area of a triangle t is denoted as Δ_t.
- n_v is the normal of vertex v and n_t is the normal of triangle t.
- All normals are considered to be normalized.

2.3 Calculation of the Cost of an Iteration

The algorithm is iterative, meaning that it performs the same operation a number of times over the mesh until certain conditions have been met. We detail in this section exactly what is done in one iteration, and what conditions need to be true for the computation to stop.

Our algorithm has a base in the area of a triangle, as well as its normal, as to the geometry and shape of a triangle. This makes us focus on Normal Field Deviation as a way to calculate the cost of a specific vertex.

2.3.1 Edge Cost

Before defining the cost of each vertex we begin by defining the cost of an edge, as the cost of the former has a dependency on the latter.

One common fact is that coplanar twin triangles are usually the best to decimate as they normally do not alter the geometry of the mesh as much as other points in the mesh would.

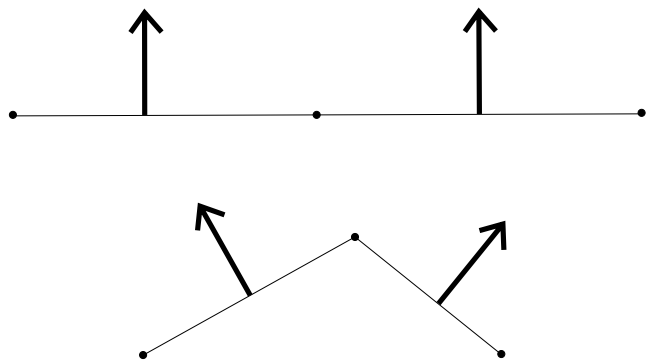


FIGURE 2.1: Normals of two different sets of triangles

As you can see in Figure 2.1, there is a very discerning property in order to determine the coplanarity of twin triangles, the dot product of their normals. When two triangles are coplanar the dot product of their normals is 1, 0 if the angle between them is 90 degrees and negative when its larger. Therefore we can arrive to the conclusion that using the normal field of such triangles can be used to our advantage. As per Hussain [4] we define the normal field deviation of an edge as:

$$NFD(e_i) = (\Delta_{t_{e1}} + \Delta_{t_{e2}})(1 - n_{t_i} \cdot n_{t_{i+1}}),$$

here t_{e1} and t_{e2} denote the twin triangles incident on the edge e_i. This, however, removes a factor in bone smoothing, which is that the fact introduced above is not necessarily the best when applied to bones. Making the cost of coplanar triangles zero no matter the length of the edge or the area that they have is a massive setback in terms of medical faithfulness. This is denoted in this thesis as the *null phenomenon*. We then introduce the formula for our Normal Field Deviation as:

$$NFD(e_i) = (\Delta_{t_{e1}} + \Delta_{t_{e2}})(1 - w(n_{t_i} \cdot n_{t_{i+1}})) \tag{2.1}$$

here w denotes a tolerance given by the user. We have found that applying a tolerance to the dot product makes the coplanar triangles have a low cost, but not 0. Because the dot product of coplanar triangles is 1, then its cost will be the summed area of the triangles times $1 - w$, making sure that in specific situations the simplification algorithm chooses a better zone of the mesh to decimate that could have a

lower cost than that. In this thesis the tolerance used was $w = 0.95$, as it produced the best results measured as detailed in Chapter 4.

While the new formula produces better results, it can be further enhanced by taking account of the fact that we do not want to choose vertices that are too far from each other, as bones have a specific general shape, decimating a large edge could cause that geometry to be changed too much. We therefore introduce the formula for calculating the cost of an edge e_i where we multiply the NFD with the length of e_i:

$$Cost(e_i) = NFD(e_i)length(e_i), \tag{2.2}$$

here $length(e_i)$ is, as the name implies, the length of that edge.

2.3.2 Vertex Cost

This is the most fundamental part of the algorithm, as this value decides which vertex to decimate on that iteration. Once again, we base our approach on Hussain [4], defining vertex cost as:

$$Cost(v) = \sum_{e_i} Cost(e_i) + \sum_{t_i} \Delta_{t_i} - ||\sum_{t_i} \Delta_{t_i} n_{t_i}||, \tag{2.3}$$

here we sum the cost of all edges and add that to the measure of local geometry, over all triangles of the one ring neighborhood of the vertex v.

The local geometry is parametrized in order to capture situations where the Normal Field Deviation does not work, such as:

Assume that the total area of the neighborhood of v is the same and the dihedral angle between triangles the same as well. Then if we only have the NFD, both vertices will be mark with the same cost, although we can see that their shape is very different and will not have the same impact when removed from the mesh.

This takes into account all of the parameters that we need for the bone simplification, meaning the preservation of the overall shape, parameterizing the area of the triangles so the collapse does not alter the shape of the bone, and also taking into account the local geometry and minimizing or completely solving the *null phenomenon*.

2.3.3 Calculation of Position of New Point

After choosing the vertex with the least cost, v, we need to choose which edge to decimate. This is simply done by choosing the edge with the least cost of the set of edges incident on v.

One particular approach to edge collapse is to simply choose one of the two vertices and simply collapse the edge to that point. It can, however, be enhanced to calculate the point on the edge that minimizes the sum of the squared distances of the new point to the planes of the triangular faces of the one ring neighborhood of v.

The optimal point, as based on Hussain [4] can be calculated by solving this linear system:

$$Au - b = 0, \tag{2.4}$$

here

$$A = \sum_{t_j} A_j,\ b = \sum_{t_j} A_j v_j \ and\ A_j = n_{t_j} n_{t_j}^T \tag{2.5}$$

This is the measure proposed by Garland and Heckbert [3] but as a local measure, which makes this algorithm extremely attractive to bone mesh simplification.

2.4 Overview of the Algorithm

The algorithm will, with each iteration, be closer to the number of faces defined by the user, or if none was defined, it will iterate until its Hausdorff Distance is bigger than the given error threshold. The general skeleton of the algorithm is as follows:

Input: triangular mesh M, with set V of vertices and set F of faces, the target number of faces *numbf*, the tolerance on coplanar triangles *tol* and the error threshold *err*.

Output: simplified triangular mesh $\overline{M}$ with less or equal number of faces than *numbf* and, if given by the user, only one vertex with an error above err.

For each Vertex $v \in V$

Calculate the cost of v and insert v in the sorted collection *Coll*;

EndFor

While number of faces in $F > numbf$ or $maxError < err$

Get lowest cost Vertex v out of *Coll*;

Get lowest cost edge e of the ones incident on v, going from v to v_e;

Compute optimal point u on the edge e;

Collapse edge e to u;

Calculate the Hausdorff Distance d of u to M

If $d > maxError$ Recompute the cost of every Vertex v_i in one ring neighborhood of v and v_e and update *Coll*; Remove v from V and all of the degenerate faces from F;

EndWhile

Here *Coll* denotes a sorted collection that has the vertices ordered in ascending order given by their cost and *maxError* is the max distance over all the iterations.

We have found this to be the best approach to the problem among all our other approaches. This is a result driven development, tuning the algorithm by reacting to how the algorithm behaved with the different datasets. The results and evaluation are discussed in Chapter 4.

2.4.1 Calculation of Current Error

From early on, the need to simplify the bone model as much as possible without removing the necessary features was present. Based in the Metro tool, Cignoni, Rocchini, and Scopigno [2], we defined the method to measure how different from the original, at a given point, was from the simplified. The Hausdorff Distance encapsulates what we want in terms of error parametrization for our two meshes. The farther away a point is, the more different from the original it is and more changed the geometry is as well.

For optimization purposes, calculating the Hausdorff Distance of the simplified mesh to the original was just too costly and slow. By realizing that every iteration 2 vertices were converted into 1 with new coordinates to minimize error, we saw that calculating the distance of that resulting vertex and comparing it to an already stored maximum, we could with certainty have the value of the Hausdorff Distance without going through all of the points, and therefore saving execution time. This, due to not disturbing more than 2 vertices every iteration, is sufficient to have an exact value for our Hausdorff Distance.

Chapter 3

Realization

In this Chapter we detail our implementation of the algorithm and the different difficulties and obstacles that we overcame.

3.1 Implementation

As the nature of the project was to have a program that would work and could be integrated in a pipeline of a medical application, not only was there a need to develop a tuned algorithm for bone simplification, but the implementation was key in what was a result driven development and evolution of the algorithm. It was all implemented in C++, with some third party frameworks that are all listed below.

To optimize the running time of the application, due to the total time being the time of parsing of the input file, the processing of the data and the writing of a new file, we adopted a version of the half-edge structure, firstly introduced in Muller and Preparata [8], as our method of storing and navigating the mesh. A first implementation of a structure suggested in Hussain [4] proved to be significantly slower than the half-edge structure.

3.1.1 Dependencies and Frameworks

- Qt, for the collections and the geometry operations;
- CGAL, for the calculation of Hausdorff Distance to measure the results;
- Eigen, for the linear system solving and matrix operations;
- Cmake, for the compilation and ease of integration with other frameworks.

3.1.2 Requirements

As this program would be integrated in a larger application, some requirements were established in order to make sure that the program would fit nicely into the pipeline.

It was established that the *de facto* file extension would the Wavefront .obj file, as it has all of the information needed for the previous and next step in the pipeline. For this reason, we implemented a parser and a file creator of .obj files as connectors to the pipeline.

Also, the medical experts expressed an important need in the program as well; interaction with the user. Namely, because no bone is exactly the same as any other, the medical expert (user), knowing the case and the patient, knows exactly what can be simplified and how much it can be simplified. This originated the following format of input parameters:

```
inputFile.obj numbFaces tolerance errorThreshold
```

This allows the user to have control over the different parameters of Chapter 2 algorithm.

The `inputFile.obj` is the receiving file from the previous stage of the pipeline. It is a highly detailed mesh that needs to be simplified for it to be usable later in the program.

The `numbFaces` is the number of faces that the simplified mesh will have once the iterations have stopped. If `errorThreshold` is not 0, the number of faces may be lower than the number, depending on the max Error of the mesh once it has `numbFaces`number of Faces.

The `tolerance` is the parameter w, as detailed in Section 2.3.1.

The `errorThreshold` is the maximum distance that a point in the simplified mesh has from the original, after the number of Faces of the simplified mesh has reached `numbFaces` .

Because the program needs to have a reasonable execution time, optimizations were made along all phases of development and complexity calculated and taken into account, eventually resulting in the results showcased in Chapter 4.

3.1.3 Pipeline

The algorithm explained in Chapter 2, as mentioned before, was created to be integrated in a larger program, as you can see in figure 3.1. The objective of it is to create a tool that from patient's CT scans creates an environment to help in pre-operation decisions, in order to make better decisions and improve the procedure.

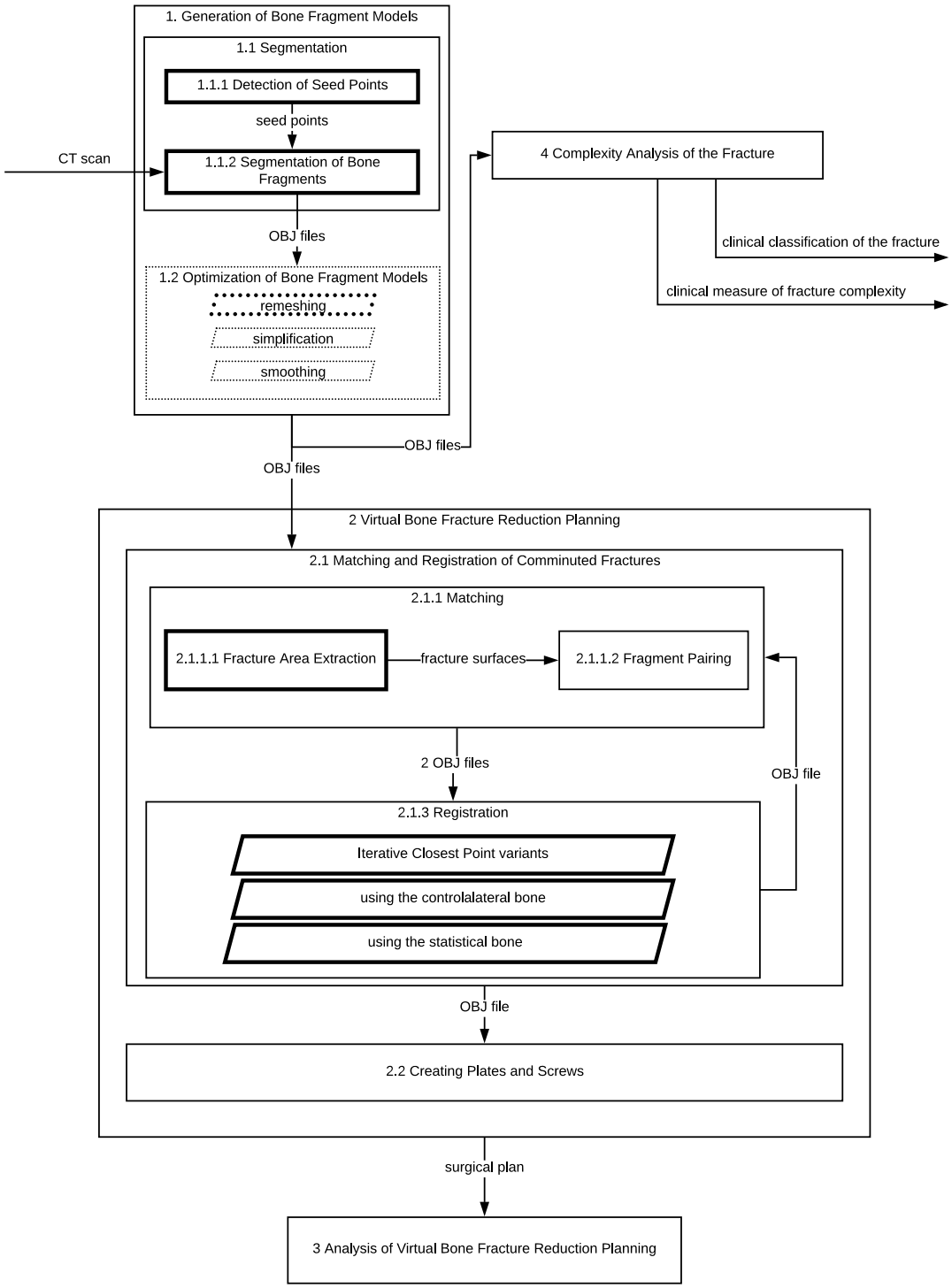


FIGURE 3.1: Diagram of the pipeline by Laura Baakman

Chapter 4

Results

In this section, the performance of our algorithm is evaluated. We take into consideration the number of faces of the resulting mesh, the max distance of a point in the simplified mesh to the original, the average of that distance over all points in the mesh, the Root Mean Square of that same measurement and execution time.

The execution times were measured on a system equipped with an Intel Core i7-4510U 2.60GHz CPU and no restriction of main memory.

Due to having a 65000 faces requirement for the mesh to go to the next stage of the pipeline, we start our tests by that same threshold in the number of faces, and then lowering the number further and further.

4.1 Maxilla

This is one of the larger meshes in our dataset, with originally 105002 faces. It has a great amount of detail in the center of the bone that needs to not be filled or disappear.

You can see in Table 4.1 and Figure 4.1 that our results have a very good performance. The average distance stays under 0.1 for most simplifications until we reach the point of oversimplification. There are, however, points, such as the max Distance, that are in the regions where when using the bone model it won't be noticed. We have seen that normally the maximum is in a secluded area, far away from the areas with more relevance.

N. Faces	Max Distance	Avg. Distance	RMS	Time (sec.)
65000	0.234100	0.0044500	0.013230	4.72423
40000	0.764376	0.0182381	0.034618	7.09167
10000	0.610886	0.0795993	0.108830	9.44289
1000	1.701230	0.3427230	0.457791	10.1368

TABLE 4.1: Results of the Maxilla Simplification with the Number of Faces as a Restriction

Because at a later stage of the development process we may not have the 65000 faces threshold, simplifying the mesh by an error driven iteration process is also a need. Our results as per Table 4.2 show that the maximum is over the threshold, but the average and RMS stay low relative to the threshold, and the execution time remains consistent with the previous approach.

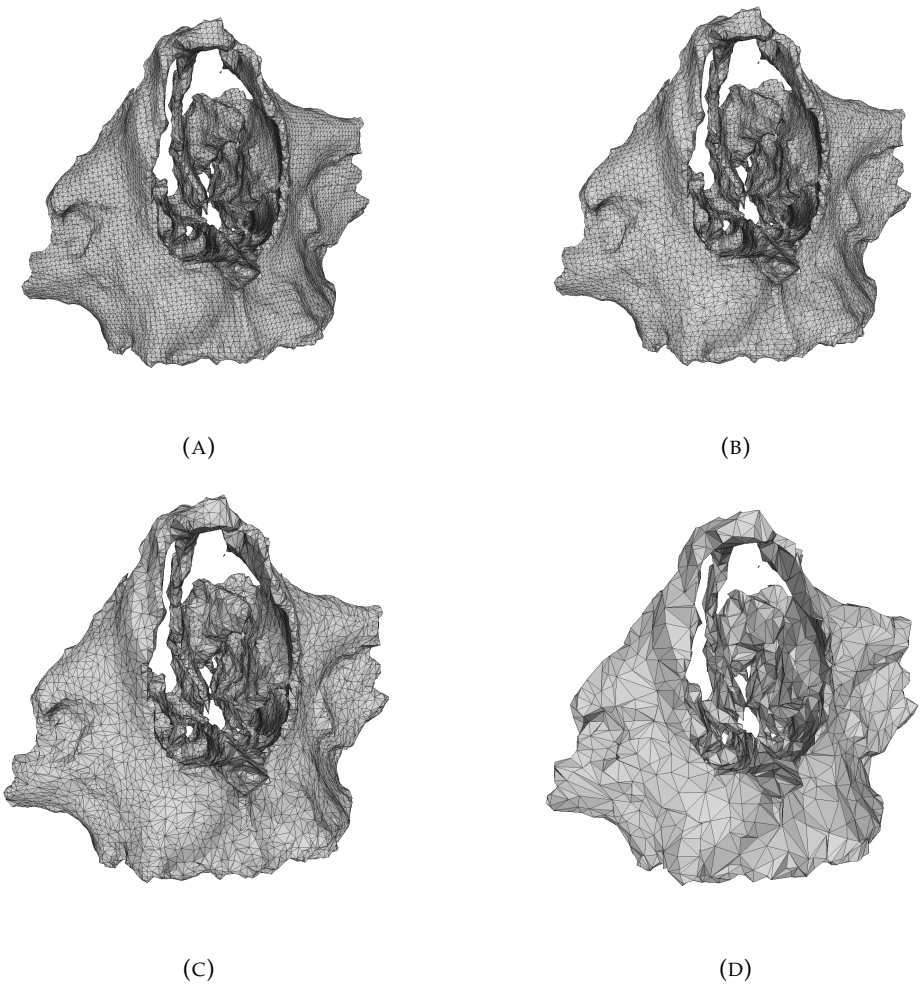


FIGURE 4.1: (A) original mesh; (B) 65000 faces; (C) 40000 faces (D) 10000 faces

Error	Max Distance	Second	Avg. Distance	RMS	N. Faces	Time (sec.)
0.1	0.11880	0.0784087	0.0003710	0.0023458	**94698**	1.36195
0.5	0.59351	0.2342100	0.0087516	0.0208670	**53864**	5.83796
1	1.70658	0.7480950	0.1083870	0.1500150	**6578**	9.71421

TABLE 4.2: Results of the Maxilla Simplification with the Error Threshold as a Restriction

4.2 Frontal Bone

This mesh is representative of a bone above the maxilla, and the relevant part of it is the big smooth area that needs to be preserved as much as possible. The original mesh has 78640 faces, and results in Figure 4.2 show that there is almost no change in the geometry of the mesh. From a geometrical standpoint, the Average distance and the RMS also stay very low even when reducing the number of faces to less than 15% of the original 78640.

N. Faces	Max Distance	Avg. Distance	RMS	Time (sec.)
65000	0.155529	0.0007788	0.0040539	1.54032
40000	0.249408	0.0072742	0.0185502	3.89901
10000	0.558263	0.0574132	0.0821606	6.28863
1000	1.487280	0.2648230	0.3481450	6.85886

TABLE 4.3: Results of the Frontal Bone Simplification with the Number of Faces as a Restriction

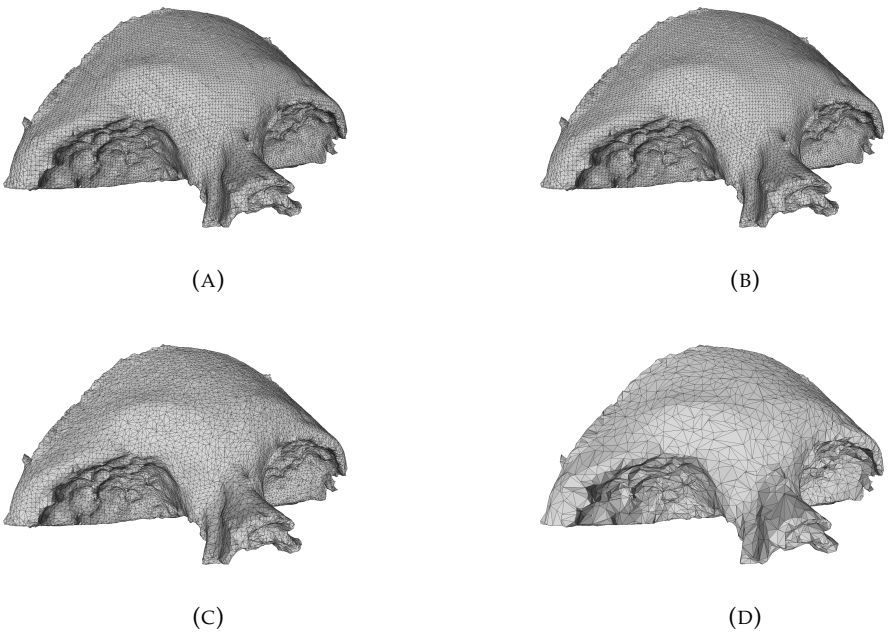


FIGURE 4.2: (A) original mesh; (B) 65000 faces; (C) 40000 faces (D) 10000 faces

The results are similar to those discussed in the previous section, giving us a sense of consistency over how the bone models are being simplified. In Table 4.4 we can also see that the number of faces gets to nearly 15% of the original with just a 0.5 error threshold.

Error	Max Distance	Second	Avg. Distance	RMS	N. Faces	Time (sec.)
0.1	0.153762	0.0845059	0.0005039	0.0029393	**68502**	1.17031
0.5	0.558263	0.4643510	0.0470106	0.0695714	**12558**	6.07148
1	1.547400	0.6952780	0.097963	0.13718400	**4814**	6.57706

TABLE 4.4: Results of the Frontal Bone Simplification with the Error Threshold as a Restriction

Chapter 5

Conclusion

5.1 Evaluation

In this section we evaluate our algorithm and compare it to a general purpose one, and give some final notes on the process.

We tackled a specific instance of the mesh simplification problem, by focusing in the area altered when simplified and the features in the area that we want to simplify. This produced results that we consider to be better for the particular case of bones.

Although the algorithm was relatively complex, implementing it proved to be simpler than expected, and therefore can be easily integrated with the pipeline and run in an optimized time, reaching our objectives set at the beginning of the development phase.

Therefore, we believe, based on the results discussed in Chapter 4, that the algorithm (and its implementation), were successful in achieving the objectives set up at the beginning of the development. The medical team also expressed their approval over the simplification, maintaining the relevant features where needed.

5.1.1 Compared against General Purpose Algorithm

As this algorithm was developed to perform better on bones, we purposely tuned it given the results that we saw when applying that simplification to the bone fragment models. The algorithm we compared it to was the Quadric Edge Collapse provided in the Meshlab ©program.

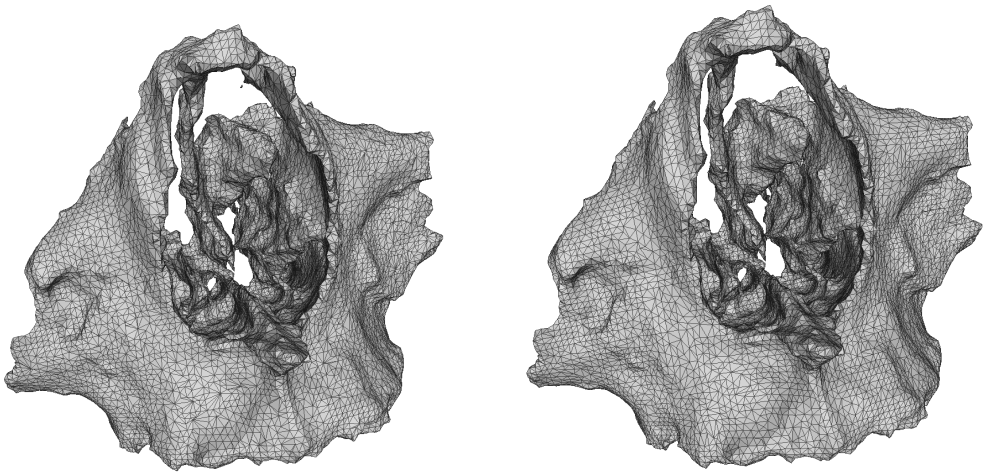
When comparing them both at 65000 and 40000 faces, they visually seem very similar, in figures 5.1 and 5.2, but the difference comes when we analyze the geometric faithfulness to the original.

In general, the algorithms are very similar, but the stark difference is in the max distance of the mesh. While our algorithm has a much lower threshold, when applying the generic algorithm the simplification would stop much earlier than ours, as the maximum distance is much larger. This can be seen in table 5.1 and 5.2.

However, when we get to 10000 faces, in figure 5.3, we can visually see where the algorithm decides to simplify. The shape is preserved on both, but different parts are preserved and others more warped. In our algorithm every feature, in this case, is slightly warped in a homogeneous way, preserving the geometry the same

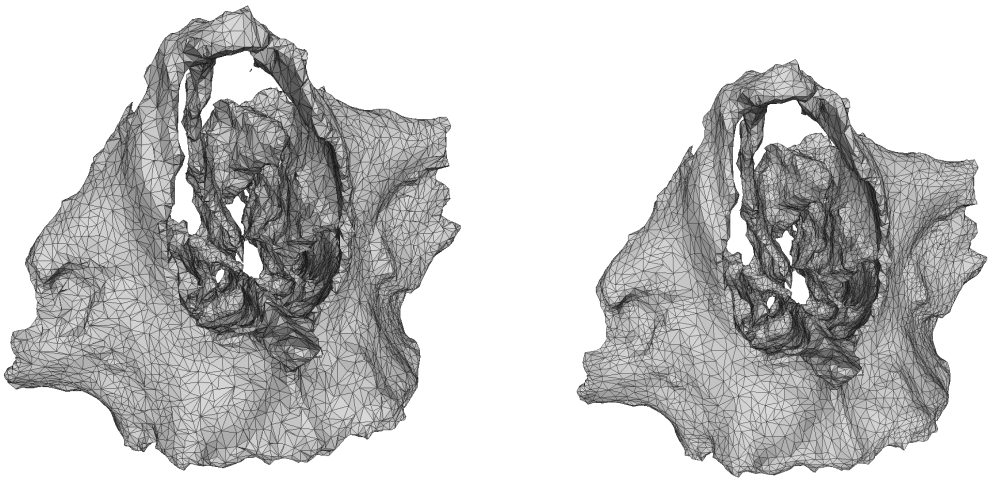
Measurement	Our Algorithm	General Purpose Algorithm
Max. Distance	0.234100	1.664236
Avg. Distance	0.004450	0.003316
RMS	0.013230	0.014676

TABLE 5.1: Comparison over 65000 faces of the Maxilla



(A) (B)

FIGURE 5.1: (A) Our implementation; (B) General Algorithm; 65000 faces

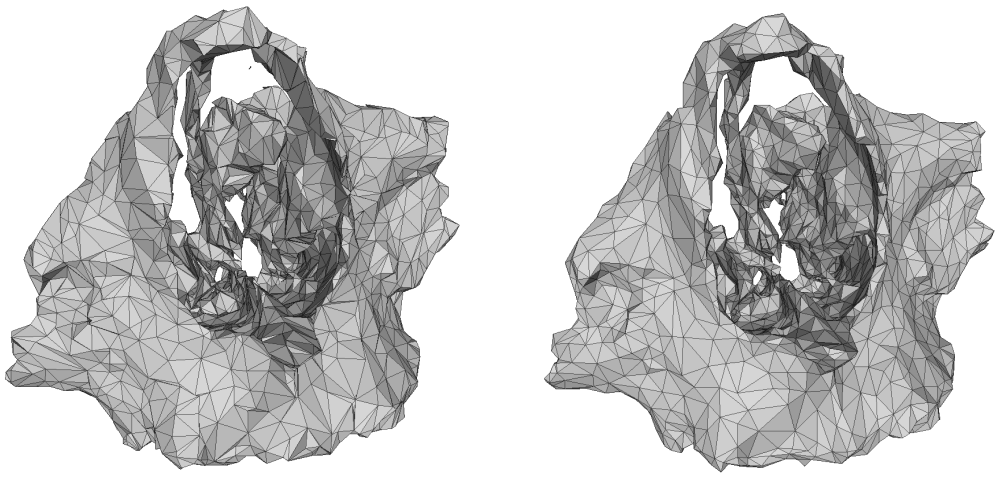


(A) (B)

FIGURE 5.2: (A) Our implementation; (B) General Algorithm; 40000 faces

Measurement	Our Algorithm	General Purpose Algorithm
Max. Distance	0.764376	3.749424
Avg. Distance	0.018238	0.013936
RMS	0.034618	0.042912

TABLE 5.2: Comparison over 40000 faces of the Maxilla



(A) (B)

FIGURE 5.3: (A) Our implementation; (B) General Algorithm; 10000 faces

Measurement	Our Algorithm	General Purpose Algorithm
Max. Distance	0.610886	4.410943
Avg. Distance	0.079599	0.068592
RMS	0.108830	0.104818

TABLE 5.3: Comparison over 10000 faces of the Maxilla

way throughout the mesh, while the general purpose algorithm has some regions preserved better, but others extremely warped, and the geometric analysis shows just that, where the max distance is incredibly different.

Looking closely at the images in this section, there is a tiny component just under the top arch that is removed by the general purpose algorithm even when the simplified mesh has 65000 faces. Our algorithm, however, does not remove any component as even the tiniest detail on a bone can be important for a procedure on it. We may sacrifice a tiny portion of overall smoothness, but we get the preservation of the components of the bone. This may also be the reason why the maximum distance is so large in the general purpose algorithm.

5.2 Future Work

Although the results are promising, we believe that there is room for improvement, as the user defined tolerance is a very subjective parameter and finding the optimal one is an arduous task. Finding that value before simplifying could be a new feature or stage in the program.

Also the medical team expressed an idea of "brush simplification" where instead of simplifying the mesh, one would select the specific zones to be simplified, like a pass of a brush, and the non selected zones would be left untouched. This could prove to be very valuable method to maintain the features only in the needed zones, optimizing speed and faithfulness to the original.

Bibliography

[1] Paolo Cignoni, Claudio Montani, and Roberto Scopigno. "A comparison of mesh simplification algorithm". In: *Computers & Graphics* 22 (Feb. 1998), pp. 37–54.

[2] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. In: (June 1998), pp. 167 –174.

[3] M. Garland and P. S. Heckbert. "Surface Simplification Using Quadric Error Metrics". In: *Ph. D. Dissertation of Carnegie Mellon University* (May 1997).

[4] Muhammad Hussain. "Efficient Simplification Methods for Generating High Quality LODs of 3D Meshes". In: *Journal of Computer Science and Technology* 3.24 (Mar. 2009), 604–inside back cover.

[5] Sjaid Hussain, Håkan Grahn, and Jan Persson. "Feature-Preserving Mesh Simplification: A Greedy Vertex Cover". In: *Computer Graphics Forum* (June 2008).

[6] Oliver van Kaick and Helio Pedrini. "A Comparative Evaluation of Metrics for Fast Mesh Simplification". In: *Computer Graphics Forum* 25 (June 2006), pp. 197–210.

[7] Youngihn Kho and Michael Garland. "User-Guided Simplification". In: *ACM Symposium on Interactive 3 Graphics, Monterey, USA* (Apr. 2003), pp. 123–126.

[8] D.E. Muller and F.P. Preparata. "Finding the Intersection of two Convex Polyhedra". In: *Theoretical Computer Science* 7 (Dec. 1978), 217–236.

[9] Jin Wei and Yu Lou. "Feature Preserving Mesh Simplification Using Feature Sensitive Metric". In: *Journal of Computer Science and Technology* 3.25 (Mar. 2010), pp. 595–605.

[10] Lianping Xing and Kin-Chuen Hui. "Entropy-based Mesh Simplification". In: *Computer-Aided Design and Applications* 7 (Aug. 2013), pp. 911–918.

[11] Jianhua Yao and Russell Taylor. "Construction and simplification of bone density models". In: *Proceedings of SPIE - The International Society for Optical Engineering* 4322 (July 2001).

[12] Fuhua Cheng Yuanfeng Zhou Caiming Zhang. "Mesh Simplification by Volume Variation with Feature Preserving". In: *Computer Graphics Forum* (2012).