Repairing Non-Manifold Triangle Meshes using Simulated Annealing

Marc Wagner[1] Ulf Labsik[2] Günther Greiner[3]

University of Erlangen-Nuremberg[4]

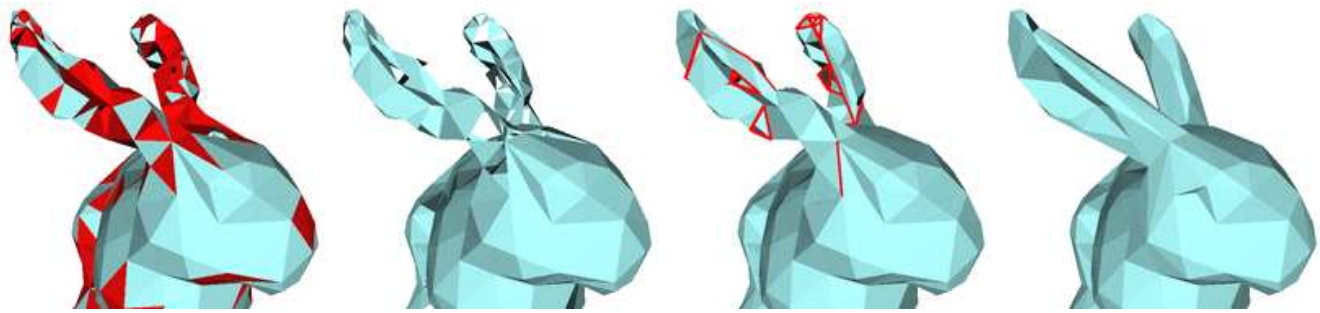

Figure 1: Repairing the bunny (487 vertices)

ABSTRACT

In the field of reverse engineering one often faces the problem of repairing triangulations with holes, intersecting triangles, Möbius-band-like structures or other artifacts. In this paper we present a novel approach for generating manifold triangle meshes from such incomplete or imperfect triangulations. Even for heavily damaged triangulations, representing closed surfaces with arbitrary genus, our algorithm results in correct manifold triangle meshes. The algorithm is based on a randomized optimization technique from probability calculus called simulated annealing.

CR Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling.

Keywords: Repairing triangulations, simulated annealing.

1 INTRODUCTION

Triangulating point clouds is an important issue in the field of reverse engineering. A lot of work dealing with this topic has been published and many surface reconstruction algorithms have been developed. While some of these algorithms guarantee a manifold triangulation of any given point cloud (e. g. Power Crust by Amenta et al. [1]) others have big problems with data sets not fulfilling certain sampling criteria (e. g. Cocone by Dey et al. [2], Gopi's algorithm based on localized Delaunay triangulations [3]). As algorithms of the latter type tend to be more efficient, it is of practical interest to know methods which extend incomplete or repair imperfect triangulations.

Unfortunately only few papers focus on this problem. To generate topologically correct triangulations Barequet et al. [4] use a technique for merging two appropriate edges at a time by moving corresponding vertices. El-Sana et al. [5] repair cracks by adding extra triangles followed by a topology simplification step. Adamy et al. [6] propose a topological clean up and a technique based on linear programming in order to establish a topologically correct surface. The approach of Carr et al. [7] is based on radial basis functions, approximating the surface represented by the underlying point cloud. In this paper we present a new and completely different algorithm for generating manifold triangle meshes from incomplete or incorrect triangulations. This algorithm is based on a randomized optimization technique called simulated annealing.

Our algorithm can be divided into three sequential phases (preprocessing, simulated annealing, postprocessing) which are presented in Sections 2 to 4. After the presentation and discussion of our results in Section 5 we close with a summary and a conclusion in Section 6.

[1]mcwagner@stud.informatik.uni-erlangen.de

[2]labsik@informatik.uni-erlangen.de

[3]greiner@informatik.uni-erlangen.de

[4]Institut für Informatik 9, Am Weichselgarten 9, 91058 Erlangen, Germany. Phone: +49 9131 85-29919. Fax: +49 9131 85-29931.

2 PREPROCESSING

The input of our algorithm can be any non-manifold triangulation, i. e. a triangulation containing holes, topological errors and intersecting triangles. To generate initial triangulations for the examples shown in this paper we used Gopi's algorithm based on lower dimensional localized Delaunay triangulations.

2.1 Removing bad triangles

A manifold triangle mesh is a closed triangle mesh without topological errors and intersecting triangles. In order to be able to extend our initial triangulation to one or more manifold triangle meshes, we first have to remove all triangles disturbing the properties of manifold triangle meshes. In the following we will call these triangles *bad triangles*. There are three classes of bad triangles:

- Triangles sharing an edge with at least two other triangles.
- Triangles connected with one vertex to the center of a closed triangle fan.
- Intersecting triangles.

Topological errors (the first two items in the list) are easy to detect. For the intersection tests between two triangles we use an algorithm presented in [8].

2.2 Closing simple holes

After the removal of bad triangles usually there are several holes in the triangulation which have to be closed. Some of these holes can simply be closed by inserting new triangles. The algorithm for such *simple holes* consists of the following steps:

1 Find all pairs of open edges and put them into a list $\mathcal{E}$ (an *open edge* is an edge which only belongs to one triangle; a *pair of open edges* are two open edges connected by a common vertex).

2 Compute for each pair of open edges the sum of the two dihedral angles α and β as shown in Figure 2.

3 Sort $\mathcal{E}$ according to $\alpha+\beta$ in descending order (triangles can now be inserted in a way that the discrete curvature of the triangulation stays small; this leads to much better shapes).

4 Remove the first pair of open edges from $\mathcal{E}$. If the triangle defined by this pair of open edges is not a bad triangle, insert this triangle into the triangulation and update $\mathcal{E}$.

5 Go back to 4 until $\mathcal{E}$ is empty.

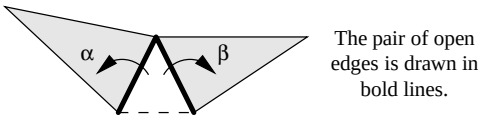

Figure 2: Computing the two dihedral angles

3 SIMULATED ANNEALING

Although all simple holes have been closed now, many triangulations still have open edges. The corresponding holes are holes which can only be closed (without violating the properties of manifold triangle meshes) if other triangles from the existing triangulation are removed. In the following these holes are denoted as *complex holes* (Figure 3). A good example for complex holes are small Möbius-band-like structures which occur quite frequently near sharp edges or corners (a Möbius band has a single boundary curve, and only one side; it is non-orientable, which means that it contains a path for which it is impossible to define a left- and right-hand side in a consistent global way). We deal with the problem of closing complex holes by applying a well known method from probability calculus called simulated annealing.

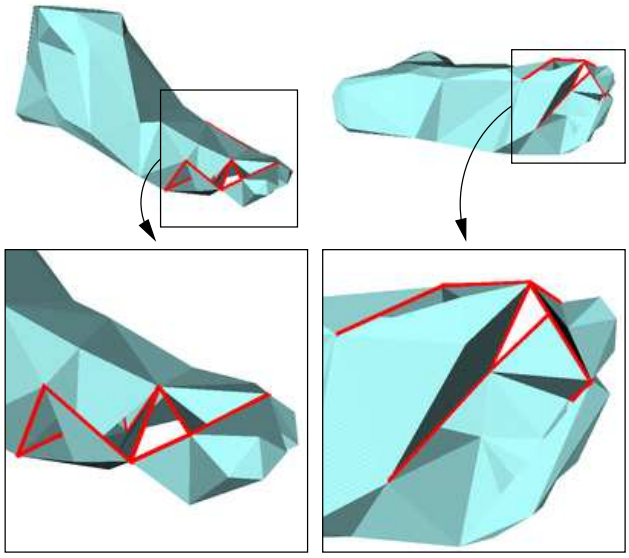

Figure 3: Complex holes in the foot (79 vertices) after the preprocessing stage (open edges are colored red)

3.1 Simulated annealing in general

Simulated annealing is a stochastic computational technique, derived from the physical process of heating and cooling

a substance to obtain a strong crystalline structure. With the aid of Simulated Annealing it is possible to find the global minimum (or at least a very good local minimum) of a given function $f : Z \to \mathbb{R}$, mapping a finite set of states $Z = \{z_1, \ldots, z_n\}$ to the set of real numbers, in a short time.

Before the Simulated Annealing can be started, it is necessary to define for every state $z_j \in Z$ a small set of neighboring states $Z_j = \{z_{j_1}, \ldots, z_{j_{n_j}}\}$ with certain properties (for details see [9, 10]).

The process of Simulated Annealing starts in an arbitrary state $z_j \in Z$. Randomly one of the neighboring states $z_k \in Z_j$ of the current state z_j is chosen. If $f(z_k) < f(z_j)$ a better state z_k has been found and the current state is replaced by that z_k. However if the new state is worse, a stochastic experiment against the probability

$$p(z_j, z_k) \quad = \quad e^{-\frac{f(z_k)-f(z_j)}{T}}$$

has to be performed. Only if the stochastic experiment is successful, the worse state z_k replaces the current state z_j, otherwise the current state z_j is kept. The parameter T (T has to be greater than 0) is called the temperature. T controls the probability to accept worse states. While the temperature T is lowered slowly, the last step is iterated again and again. When finally after many iterations a low temperature has been reached, there is a very good chance that the current state is the global minimum of the function f.

3.2 Closing complex holes using simulated annealing

The idea of using simulated annealing to solve optimization problems on triangulations was first proposed by Schumaker [11]. With the help of simulated annealing a given planar triangulation is modified in a way that it optimizes a certain optimization criterion.

In contrast to this algorithm where the input is already a complete and valid triangulation, we use the technique of simulated annealing to create one or more manifold triangle meshes from a correct but incomplete triangulation.

As described in Section 3.1 our first task is to define a finite set of states Z, small sets Z_j of neighboring states for all $z_j \in Z$ and a cost function $f : Z \to \mathbb{R}$ assigning a value to every state in Z.

3.2.1 The set of states

We define the set of states Z as the set of all valid triangulations $\mathbf{T}_j$ of the underlying point cloud $\mathcal{P}$. A valid triangulation $\mathbf{T}_j$ is characterized by the following properties:

- The vertices of all triangles of $\mathbf{T}_j$ are elements of $\mathcal{P}$.
- The triangulation $\mathbf{T}_j$ has no topological errors and no intersecting triangles (but may contain holes).

3.2.2 The sets of neighboring states

The set of neighboring states Z_j of a valid triangulation $\mathbf{T}_j \in Z$ are all triangulations which can be derived from $\mathbf{T}_j$ by:

A Inserting a new triangle at a pair of open edges and deleting all disturbing triangles (triangles of the existing triangulation which violate the properties of manifold triangle meshes after the new triangle has been inserted).

B Deleting a triangle with three open edges (these triangles have to be deleted, because they cannot be reconnected to the main part of the triangulation with the transitions defined in **A**).

3.2.3 The cost functions

Our primary objective is to generate closed manifold triangle meshes. Hence the first cost function is defined by

$$f_1(\mathbf{T}_j) = \text{number of open edges in } \mathbf{T}_j \quad .$$

In general in simulated annealing one does not know if the current state is a global minimum of the cost function or not. This uncertainty is a big flaw of this technique. It is important to note that we know exactly when we have reached a global minimum of f_1 as the function value at this minimum is 0. Therefore we will continue with the simulated annealing until we have reached a triangulation $\mathbf{T}_j$ with $f_1(\mathbf{T}_j) = 0$. This is the case, if our current triangulation consists of one or more closed manifold triangle meshes.

In order to have some influence on the shape of the resulting triangulation, we have defined a second cost function f_2, which is the discrete mean curvature of the whole triangulation (for details see [12]) divided by the sum of all edge lengths. f_2 is given by

$$f_2(\mathbf{T}_j) \quad = \quad \frac{\sum_{e \in E} \alpha_e \cdot \|e\|}{\sum_{e \in E} \|e\|} \quad ,$$

where E is the set of all edges in $\mathbf{T}_j$ which have two triangles, $\|e\|$ is the length of the edge e and α_e is the dihedral angle of the two adjacent triangles of the edge e. Putting more weight on this cost function allows the user to produce meshes of higher quality and to prevent that the triangulation is torn into several parts.

3.2.4 A single simulated annealing step

A single simulated annealing step is performed as follows. Let $\mathbf{T}_j$ be the current triangulation. First a neighboring triangulation $\mathbf{T}_k \in Z_j$ is chosen randomly. Then two stochastic experiments are performed against the probabilities

$$p_1(\mathbf{T}_j, \mathbf{T}_k) \quad = \quad e^{-\frac{c_1 \cdot (f_1(\mathbf{T}_k)-f_1(\mathbf{T}_j))}{T}}$$

and

$$p_2(\mathbf{T}_j, \mathbf{T}_k) \quad = \quad e^{-\frac{c_2 \cdot (f_2(\mathbf{T}_k) - f_2(\mathbf{T}_j))}{T}}$$

determined by the two cost functions f_1 and f_2. If both experiments are successful, the transition from $\mathbf{T}_j$ to $\mathbf{T}_k$ is accepted and $\mathbf{T}_k$ replaces the current triangulation, otherwise the current triangulation $\mathbf{T}_j$ remains unchanged.

The two constants c_1 and c_2 are parameters, which have to be specified by the user. A big value for the ratio $\frac{c_1}{c_2}$ places more weight on a fast reconstruction, while a small value leads to better results (smoother meshes, lower probability of tearing the triangulation into two or more parts). Experiments have shown, that the values $c_1 = 1.0$ and $c_2 = 3.0$ produce good results for most point clouds.

3.2.5 The complete simulated annealing process

The complete simulated annealing process is made up of s phases. In each phase a number of t simulated annealing steps are performed (s and t are user-defined parameters). The phases differ from each other only by the slowly descending temperature T, which controls the probability to choose worse states. Let phase $s-1$ be the first phase and phase 0 be the last phase. Then the temperature T_k of phase k is given by

$$T_k \quad = \quad (T_1)^k \quad ,$$

where $T_1 > 1$ is the temperature of phase 1. With the parameter T_1 the user can control how fast the temperature will change ($T_1 \approx 1$ stands for a very slow change while $T_1 \gg 1$ results in a fast descent). $T_1 = 1.25$ has proven to be appropriate for most point clouds.

If the number of open edges in the triangulation reaches 0 the algorithm terminates. If there are still open edges after phase 0, the simulated annealing has to be started again (maybe another combination of parameters (c_1, c_2, T_1, s, t) should be considered).

Finding appropriate values for s and t requires a bit of experience. A strategy for automatic reconstruction which has proven to be successful is the following. $c_1 = 1.0$, $c_2 = 3.0$ and $T_1 = 1.25$ are always kept constant. At the beginning there is only one phase ($s = 1$) consisting of $t = (10 \times \text{number of open edges})$ steps. If there are still open edges left after phase 0, the number of phases is each time increased by 1 until the simulated annealing finally provides a closed triangulation.

3.2.6 Enlarging holes

In rare cases it is possible that the current triangulation $\mathbf{T}_j$ is not connected to one or more closed triangle meshes (a global minimum of f_1) by the transitions defined in Section 3.2.2. In such cases the triangulation cannot be closed by the simulated annealing process described in the previous sections. To make sure that the simulated annealing will always converge, we use a simple trick called *enlarging holes*.

If the simulated annealing does not converge for a long time (after starting the whole process five times if the strategy for automatic reconstruction described in Section 3.2.5 is used), all triangles with at least one open edge are deleted. By doing this the existing holes are enlarged and new transitions become possible. In most cases the simulated annealing will now find a global minimum of f_1. If this is not the case the existing holes are enlarged twice, then thrice and so on, until the simulated annealing finally reaches a triangulation consisting of one or more closed manifold triangle meshes.

4 POSTPROCESSING

4.1 Inserting isolated points

In some cases there is a small number of *isolated points* (points of the underlying point cloud which are not anymore connected to the resulting triangulation) after the simulated annealing which have to be reinserted into the triangulation (Table 1 lists the number of isolated points after the simulated annealing stage for the examples in Figures 1 and 5).

Let $\mathbf{p}_j$ be an isolated point. In order to insert this point into the current triangulation we first delete a triangle near the point $\mathbf{p}_j$. The isolated point $\mathbf{p}_j$ is then inserted into this newly generated hole as shown in the top line of Figure 4. The triangle is chosen in a way that it minimizes the mean curvature of the resulting triangulation under the condition that this triangulation still has the properties of manifold triangle meshes. If $\mathbf{p}_j$ is close to one of the edges of the triangulation deleting two triangles usually leads to better shapes (bottom line Figure 4).

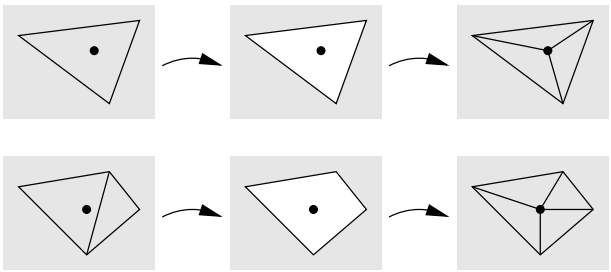

Figure 4: Inserting an isolated point over one and two triangles respectively

Note that there are triangulations where isolated points cannot be inserted with the method presented in this section. However none of these rather theoretical constructs (triangulations where no triangle is completely visible from the point to be inserted) has ever been observed in our tests.

4.2 Smoothing the mesh

The quality of the resulting triangle meshes can further be improved by using a smoothing algorithm proposed in [12]. The basic idea of this algorithm is to apply a series of edge flipping operations to minimize the discrete mean curvature of the triangulation.

5 RESULTS

We applied our algorithm to many different, incomplete or imperfect triangulations. These triangulations were generated by using Gopi's algorithm [3] on point clouds not fulfilling the corresponding sampling criterion. The sizes of those point clouds were ranging from less than 100 points to more than 50,000 points. All our tests resulted in correct manifold triangle meshes.

Even heavily damaged triangulations could be repaired and extended to manifold triangle meshes without loosing their shape (Figures 1 and 5; from left to right: the initial triangulation with Gopi's algorithm (bad triangles are colored red), the triangulation after the removal of bad triangles, the input mesh for the simulated annealing (open edges are colored red), the output of our algorithm).

Our algorithm proved to be quite efficient as well. For most densely sampled point clouds ($\geq 50,000$ points) the simulated annealing takes significantly less time than the algorithm providing the initial triangulation (only small regions in the point clouds are not fulfilling the sampling criterion and have to be repaired). For small triangle meshes ($\leq 1,000$ vertices) the simulated annealing takes proportionally much more time but the whole process of triangulating and repairing is only a matter of seconds (Table 1; the times were measured on an AMD Athlon 1200 MHz with 1 GB RAM).

model	#v	#ip	t_{pre}	t_{sim}	t_{post}
foot	79	1	0	0	0
bunny	487	3	0	2	0
tetrahedron	1000	6	0	10	1
beethoven	2515	15	2	15	3
horse	9264	7	14	25	6

#v = number of vertices
#ip = number of isolated points after the simulated annealing stage
t_{pre} = time in seconds needed for preprocessing
t_{sim} = time in seconds needed for simulated annealing
t_{post} = time in seconds needed for postprocessing

Table 1: Number of isolated points after the simulated annealing stage and performance of our algorithm for the examples in Figures 1 and 5

6 CONCLUSION

By using the technique of simulated annealing we are able to generate manifold triangle meshes even from heavily damaged triangulations. By randomly changing the triangulation and allowing temporarily even worse states, the algorithm will finally generate one or more manifold triangle meshes. In contrast to other deterministic mesh repairing algorithms, our approach may lead to different results when applying it several times to the same point cloud.

Combining our method with existing algorithms for point cloud triangulation leads to efficient triangulation algorithms, which are able to generate manifold triangle meshes even from sparsely sampled point clouds not fulfilling any sampling criteria.

REFERENCES

[1] N. Amenta, S. Choi, R. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, vol. 19, no. 2-3, pp. 127-153, 2001.

[2] N. Amenta, S. Choi, T. K. Dey, N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Proc. 16th Sympos. Comput. Geom.*, pp. 213-222, 2000.

[3] M. Gopi, S. Krishnan, C. T. Silva. Surface Reconstruction based on Lower Dimensional Localized Delaunay Triangulation. *Computer Graphics Forum*, 2000.

[4] G. Barequet, S. Kumar. Repairing CAD Models. *IEEE Visualization 97*, pp. 363-370, 1997.

[5] J. El-Sana, A. Varshney. Topology Simplification for Polygonal Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 2, 1998.

[6] U. Adamy, J. Giesen, M. John. New Techniques for Topologically Correct Surface Reconstruction. *IEEE Visualization*, pp. 373-380, 2000.

[7] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, T. R. Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. *ACM SIGGRAPH 2001*, pp. 67-76, 2001.

[8] T. Möller. A Fast Triangle-Triangle Intersection Test. *Journal of Graphics Tools*, vol. 2, no. 2, 1997.

[9] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, vol. 21, no. 6, pp. 1087-1092, 1953.

[10] S. Kirkpatrick, C. Gelatt Jr., M. Vecchi. Optimization by Simulated Annealing. *Science*, vol. 220, pp. 671-680, 1983.

[11] L. L. Schumaker. Computing Optimal Triangulations Using Simulated Annealing. *Computer Aided Geometric Design*, vol. 10, no. 3-4, pp. 329-346, 1993.

[12] N. Dyn, K. Hormann, S. J. Kim, D. Levin. Optimizing 3D triangulations using discrete curvature analysis. *Mathematical methods for curves and surfaces*, pp. 135-146, 2000.

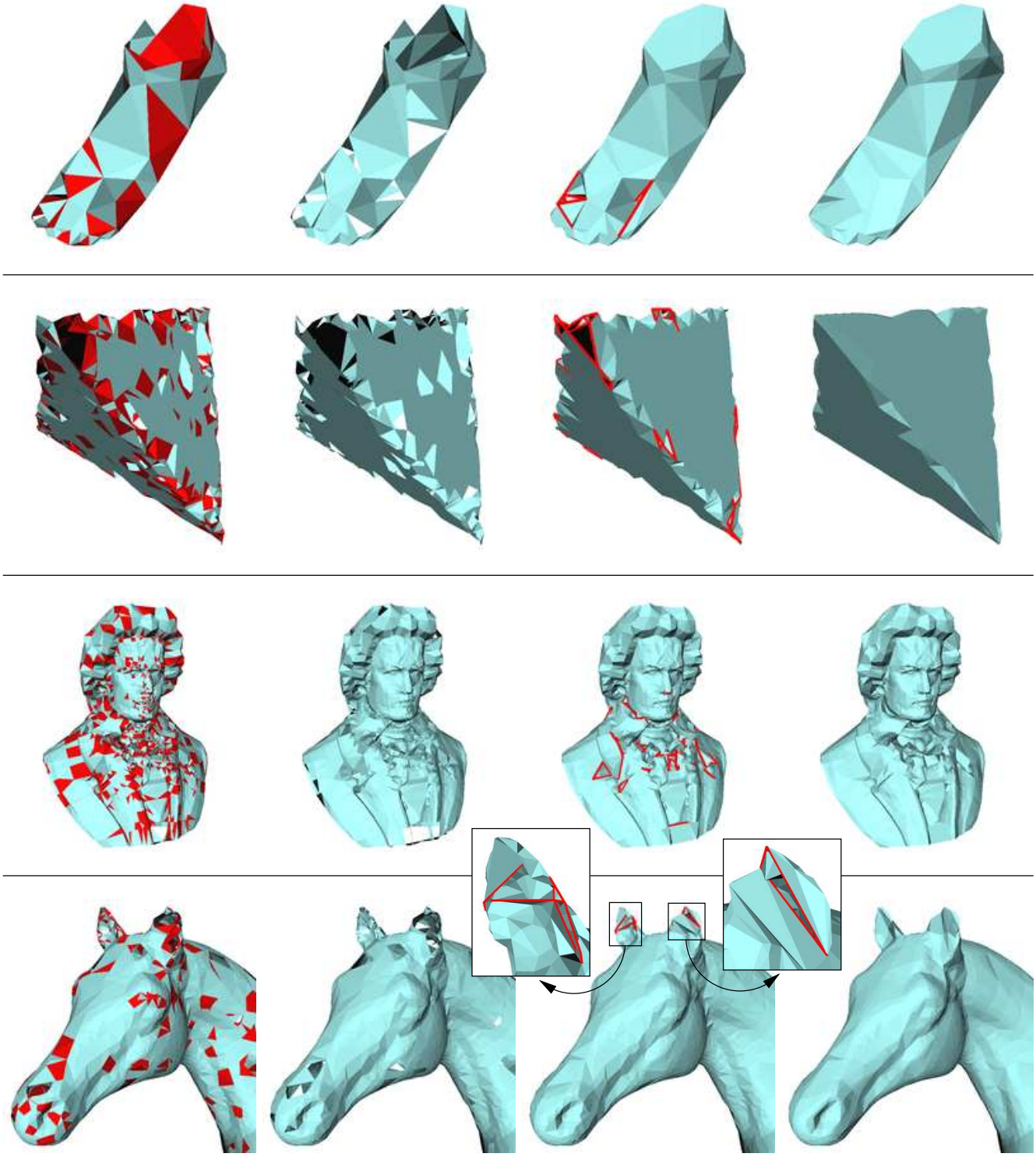

Figure 5: Repairing the foot (79 vertices), the tetrahedron (1000 vertices), the beethoven (2515 vertices) and the horse (9264 vertices) (from top to bottom)