International Journal of Shape Modeling
© World Scientific Publishing Company

# REPAIRING NON-MANIFOLD TRIANGLE MESHES USING SIMULATED ANNEALING

MARC WAGNER
*Institut für Informatik 9, Am Weichselgarten 9, 91058 Erlangen, Germany*
*mcwagner@immd9.informatik.uni-erlangen.de*
*http://www9.informatik.uni-erlangen.de*

ULF LABSIK
*Institut für Informatik 9, Am Weichselgarten 9, 91058 Erlangen, Germany*
*labsik@informatik.uni-erlangen.de*

GÜNTHER GREINER
*Institut für Informatik 9, Am Weichselgarten 9, 91058 Erlangen, Germany*
*greiner@informatik.uni-erlangen.de*

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.

*Keywords*: CAD-repair; repairing triangulations; simulated annealing.

1991 Mathematics Subject Classification: 52B70, 60J05

## 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.), others have serious problems with data sets not fulfilling certain sampling criteria (e. g., Cocone by Amenta et al. or Gopi's algorithm based on localized Delaunay triangulations).[1,2,3,4] 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.

## 1.1. *Previous work*

The problem of repairing CAD-models has been addressed by several authors in the last decade and a couple of completely different techniques has been developed.

Barequet et. al repair cracks in triangle meshes by matching border polygons. These borders are then stitched by inserting new triangles. Remaining holes are triangulated so that the area of the resulting triangulation is minimized.[5] Morvan et al. propose an interactive virtual environment where CAD-models can be repaired by manually inserting or deleting triangles.[6] Assuming that the triangle mesh was obtained from range images, Curless et al. deal with the problem of repairing CAD-models by applying a volumetric method capable of detecting and triangulating holes.[7] To generate topologically correct triangulations Barequet et al. propose an interactive system where cracks are fixed by merging two appropriate edges at a time by moving corresponding vertices. Remaining holes are filled with triangles so that a certain shape functional is minimized.[8,9] El-Sana et al. repair cracks by adding extra triangles, followed by a topology simplification step.[10] Adamy et al. propose a topological clean up and a technique based on linear programming to establish a topologically correct surface.[11] The approach of Carr et al. is based on radial basis functions, approximating the surface represented by the underlying point cloud.[12]

In this paper we present a new and completely different algorithm for generating manifold triangle meshes from incomplete or incorrect triangulations. The algorithm is based on a randomized optimization technique called simulated annealing. The simulated annealing changes the input triangulation step by step until a manifold triangulation has been obtained. The algorithm is able to handle even heavily damaged triangle meshes and converts them into manifold triangulations. In most cases the resulting shape is visually pleasing and similar to the object from which the underlying point cloud was sampled.

## 1.2. *Overview*

Our algorithm can be divided into the following three sequential phases which will be discussed in detail in Sections 2 to 4:

1. Preprocessing:
   (a) Removing bad triangles:
   First we delete all triangles, disturbing the manifold properties of the input triangulation, i. e., intersecting triangles and triangles causing topological errors.
   (b) Closing simple holes:
   Now we close as many holes as possible by simply inserting new triangles. Note that there are also holes which cannot be closed without removing existing triangles before.
2. Simulated annealing:

Holes which could not be triangulated in the preprocessing stage will now be closed by an optimization technique called simulated annealing. Randomly the triangulation is changed, temporarily even to worse configurations, until there are no more holes left. The resulting triangulation is a manifold triangulation then.

3. Postprocessing:
   (a) Inserting isolated points:
   During the simulated annealing it may occur, that points become separated from the triangulation. These isolated points will now be reinserted.
   (b) Smoothing the triangulation:
   To improve the resulting triangulation we finally apply a smoothing algorithm proposed by Dyn et al.[13]

After the presentation and discussion of our results in Section 5 we close with a summary and a conclusion in Section 6.

## 2. Preprocessing

The input of our mesh repair algorithm can be a damaged triangulation of any kind, i. e., a triangulation containing holes, topological errors and intersecting triangles.

### 2.1. *Removing bad triangles*

The first step on the way towards a manifold triangulation consists in generating a valid triangulation.

A *valid triangulation* of a point cloud $\mathcal{P}$ is characterized by the following two properties:

- The vertices of all triangles are elements of $\mathcal{P}$.
- There are no intersecting triangles and no local topological errors (a *local topological error* is either a triangle, sharing an edge with two or more other triangles or a triangle connected with one vertex to the center of a closed triangle fan (left part of Fig. 1)).

A *manifold triangulation* is a valid triangulation without holes, i. e., every edge is shared by exactly two triangles. Note that this definition rules out all types of global topological errors (e. g., Möbius-band-like structures or Klein-bottles, that is, triangulations where it is impossible to define an inner and an outer side in a consistent global way) since valid triangulations containing such errors cannot be closed without generating self intersections.

In order to be able to extend our initial triangulation to a manifold triangulation, we first have to remove all triangles disturbing the properties of valid triangulations. In the following we will call those triangles *bad triangles*. There are three types of bad triangles (Fig. 1):

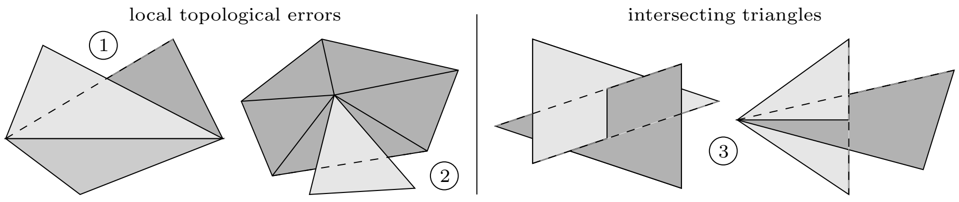

Fig. 1. The three types of bad triangles.

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

Detecting local topological errors (type 1 and type 2) can be performed very efficiently, because the corresponding tests are local tests (we only have to consider triangles adjacent to a certain edge or a certain vertex at a time).

If more than two triangles share a common edge (type 1), we remove all but those two triangles with the largest angles opposite the common edge. Experiments have shown that this strategy will yield the better triangulation in most cases (the triangulation, which is closer to the shape of the object, from which the underlying point cloud was sampled). Of course there are counter-examples but deleting the "wrong triangle" at this stage is usually not a serious problem. During the simulated annealing (Section 3) this triangle may be reinserted while another more disturbing triangle will be deleted.

If there is a triangle or an open triangle fan connected to the center of a closed triangle fan (type 2), we keep the latter. In the rare case, that two or more closed triangle fans are connected, we arbitrarily remove all but one of them.

For the intersection tests between triangles we use an algorithm presented by Möller.[14] To perform the intersection tests in an efficient way, we put all triangles into a list and sort that list according to their minimal coordinates along one of the three axes (that axis is chosen, along which the triangulation has its maximum dimension). By doing that, we only have to perform intersection tests between triangles, having nearby positions in the triangle list (there is no need to check two triangles, where the maximum coordinate of the first is less than the minimum coordinate of the second, and vice versa). If two triangles intersect each other, we remove that triangle with the smallest minimum angle. Usually that strategy leads to the better triangulation (the triangulation, which is closer to the shape of the original object). If not, this will still not be a problem, because the deleted triangle may be reinserted during the simulated annealing stage, as already mentioned above.

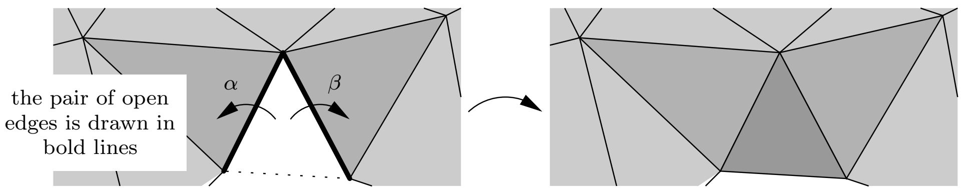

Fig. 2. Inserting a new triangle at a pair of open edges.

## 2.2. *Closing simple holes*

After the removal of bad triangles usually there are several holes in the now valid triangulation. In order to obtain a manifold triangulation these holes have to be closed. Some of these holes can simply be closed by inserting new triangles. In the following these holes will be denoted as *simple holes*. Note that there are also holes which can only be closed without violating the properties of valid triangulations if existing triangles are deleted (e. g., consider triangulations shaped like Möbius-bands or Mexican hats). These holes, referred to as *complex holes*, will be discussed in detail in Section 3.

The algorithm for triangulating *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, belonging to one triangle only; a *pair of open edges* are two open edges, connected by a common vertex, not belonging to the same triangle). A pair of open edges defines a triangle not present in the current triangulation, i. e., a pair of open edges is a possibility to insert a new triangle (Fig. 2).
2. Compute for each pair of open edges the sum of the two angles $\alpha$ and $\beta$ as shown in Fig. 2. $\alpha$ and $\beta$ are the angles between the virtual triangle, defined by the pair of open edges, and the two adjacent triangles, already present in the current triangulation.
3. Sort $\mathcal{E}$ according to $\alpha + \beta$ in descending order (triangles can now be inserted in a way that the discrete mean curvature of the triangulation stays small; doing that, usually leads to shapes of better quality).
4. Remove the first pair of open edges from the list $\mathcal{E}$. If the triangle defined by this pair of open edges is not a bad triangle, insert that triangle into the current triangulation and update $\mathcal{E}$.
5. Go back to 4 until $\mathcal{E}$ is empty.

## 3. Simulated annealing

### 3.1. *Complex holes*

Although all simple holes have been closed now, many triangulations still have open edges. The corresponding complex holes can only be closed if other triangles of the existing triangulation are removed.

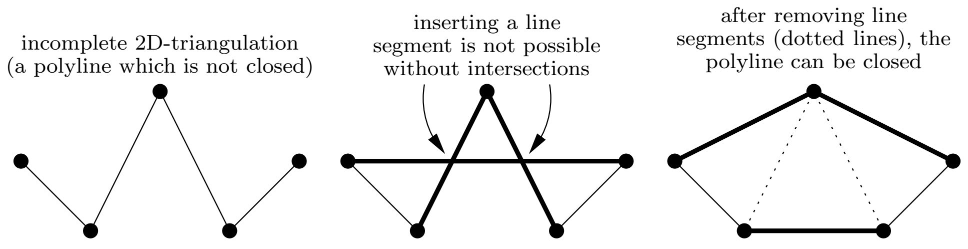
incomplete 2D-triangulation
(a polyline which is not closed)
inserting a line segment is not possible without intersections
after removing line segments (dotted lines), the polyline can be closed

Fig. 3. Closing a two dimensional complex hole.

To get a better understanding of what complex holes are, we will first discuss a simple two dimensional analogy. The two dimensional correspondence of a three dimensional triangle mesh is a polyline (Fig. 3, left part). In order to close that polyline, we can try to insert a line segment (this is like triangulating simple holes in three dimensions; a line segment in two dimensions is equivalent to a triangle in three dimensions). There is only one possibility to insert a line segment, which unfortunately leads to an intersecting polyline (Fig. 3, middle part; intersecting line segments are drawn in bold lines). Therefore we first have to remove two line segments, before we are able to close the polyline (Fig. 3, right part).

A good example for complex holes in three dimensions 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). Fig. 4 shows several complex holes in the foot model in detail. Open edges, forming the borders of complex holes, are colored red.

We deal with the problem of closing complex holes by applying a well known optimization method from probability calculus called simulated annealing.

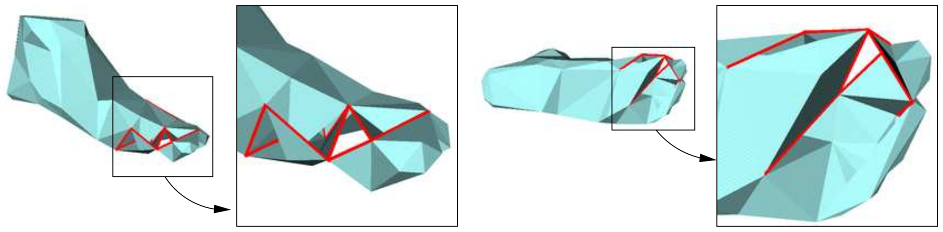

Fig. 4. Complex holes in the foot after the preprocessing stage (79 vertices, 156 triangles, 14 open edges (colored red)).

## 3.2. *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 help 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 Ref. 15 and 16).

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

$$p(z_j, z_k) = e^{-\frac{f(z_k) - f(z_j)}{T}} \tag{1}$$

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 $f$.

## 3.3. *Closing complex holes using simulated annealing*

The idea of using simulated annealing to solve optimization problems on triangulations was first proposed by Schumaker.[17] With the aid of simulated annealing a given planar triangulation is modified in a way, so 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 a manifold triangulation from a valid but incomplete triangulation (obtained by performing the preprocessing steps in Section 2).

As described in Section 3.2 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 $Z$.

### 3.3.1. *The set of states*

The set of states $Z$ is defined as the set of all valid triangulations $\mathbf{T}_j$ of the underlying point cloud, i. e., $Z = \{\mathbf{T}_1, \ldots, \mathbf{T}_n\}$.

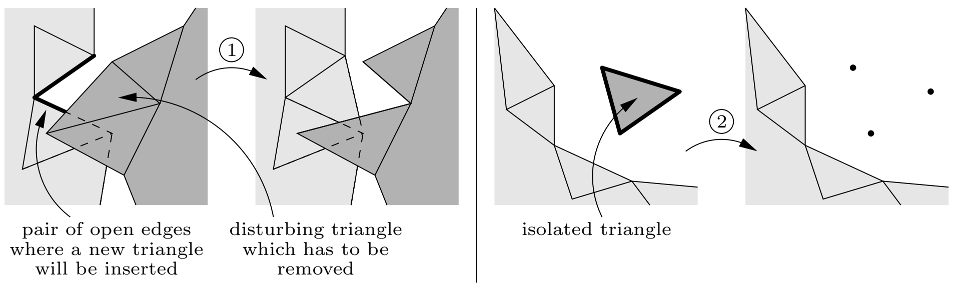
1
2
pair of open edges where a new triangle will be inserted
disturbing triangle which has to be removed
isolated triangle

Fig. 5. Transitions to neighboring triangulations.

### 3.3.2. *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 applying one of the following two transition rules:

1. Insert a new triangle at a pair of open edges and delete all disturbing triangles (triangles of the current triangulation, which violate the properties of manifold triangle meshes after the new triangle has been inserted; Fig. 5, left part).
2. Delete a triangle with three open edges (these triangles have to be deleted, because they cannot be reconnected to the triangulation with the transitions defined in 1; Fig. 5, right part). The three vertices of that triangle, now isolated from the current triangulation, will later be reinserted (Section 4.1).

### 3.3.3. *The cost functions*

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

$$f_1(\mathbf{T}_j) = \text{number of open edges in } \mathbf{T}_j \ . \tag{2}$$

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 usually a major flaw of this technique. It is important to note that we know exactly when we have reached a global minimum of $f_1$ since 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 divided by the sum of all edge lengths (for detailed information about the curvature of triangle meshes see Ref. 13). $f_2$ is given by

$$f_2(\mathbf{T}_j) = \frac{\sum_{e \in E} \alpha_e \cdot \|e\|}{\sum_{e \in E} \|e\|} \ , \tag{3}$$

where $E$ is the set of all edges in $\mathbf{T}_j$ which are shared by two triangles, $\|e\|$ is the length of the edge $e$ and $\alpha_e$ is the dihedral angle of the two triangles adjacent to the edge $e$. Putting more weight on this cost function has two effects:

- Triangles are inserted in a way so that the mean curvature of the triangulation stays small. This leads to smoother meshes which are visually more pleasing and therefore of better quality.
- Because we are considering the mean curvature per unit edge length only the removal of triangles at sharp edges or corners will be favored. On the other hand experiments have shown, that the triangulation is usually torn into two or more pieces if triangles forming smooth regions are deleted. Hence by weighting $f_2$ appropriately we have a mechanism to prevent the triangulation from being torn into several parts.

### 3.3.4. *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

$$p_1(\mathbf{T}_j, \mathbf{T}_k) = e^{-\frac{c_1 \cdot (f_1(\mathbf{T}_k) - f_1(\mathbf{T}_j))}{T}} \tag{4}$$

and

$$p_2(\mathbf{T}_j, \mathbf{T}_k) = e^{-\frac{c_2 \cdot (f_2(\mathbf{T}_k) - f_2(\mathbf{T}_j))}{T}} \ , \tag{5}$$

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 large value for the ratio $c_1/c_2$ places more weight on a fast repairing process, 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 choosing $c_1/c_2 \approx 1/3$ yields good results for most input triangulations. The effects of different ratios $c_1/c_2$ will be discussed in more detail in Section 5.

### 3.3.5. *The complete simulated annealing process*

The complete simulated annealing process is made up of $s$ phases. In each phase $t$ simulated annealing steps are performed ($s$ and $t$ are user-defined parameters). The phases only differ from each other 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

$$T_k = (T_1)^k \ , \tag{6}$$

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 input triangulations. Note that besides Eq. (6) there are many other temperature functions imaginable. In principle any monotonically decreasing positive function could be used instead. We decided to use (6) because it is simple as well as leading to good results.

If the number of open edges in the current triangulation reaches 0, the simulated annealing can be stopped immediately. If there are still open edges after phase 0, the simulated annealing has to be restarted (maybe another set of parameters $(c_1, c_2, T_1, s, t)$ should be considered).

Specifying appropriate values for $s$ and $t$ requires a little experience. A strategy for automatic reconstruction, which has proven to be successful for most incomplete triangulations, 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 $t = (10 \times$ number of open edges) steps. If there are still open edges after phase 0, each time the number of phases is incremented by 1. This corresponds to starting with a higher temperature (Eq. (6)). On the other hand the probability to find a path to a closed triangulation via one or more worse intermediate states is much higher in a hot system than in cold system. Incrementing the number of phases therefore amounts to increasing the probability to convert the current triangulation into a manifold triangulation.

### 3.3.6. *Enlarging holes*

In rare cases it is possible that the current triangulation is not connected to one or more closed triangle meshes (a global minimum of $f_1$) by the transitions defined in Section 3.3.2. Then the triangulation cannot be closed by the simulated annealing process described above. To make sure that the simulated annealing always converges, 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 3.3.5 is used), all triangles with at least one open edge are deleted (Fig. 6). 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 manifold triangulation.

Note that this guarantees the convergence of the simulated annealing, since more and more triangles will be deleted as long as no manifold triangulation has been

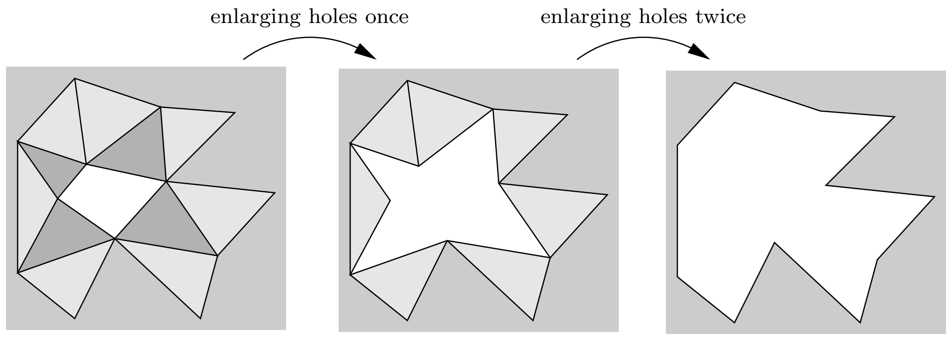

Fig. 6. Enlarging holes.

found. As an extreme example consider a point cloud obtained by randomly sampling points on a plane. The only possible manifold triangulation is the degenerate solution with no triangles, and that is exactly what our method will finally yield.

However our experiments have shown that most incomplete input triangulations do not require the use of enlarging holes more than once. It follows that although the convergence is assured by enlarging holes, there is no need to worry that too many triangles of the current triangulation are deleted and with that too much information about the shape is discarded.

## 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.

Let $\mathbf{p}$ be an isolated point. In order to insert this point into the current triangulation, we first delete a single triangle or a pair of adjacent triangles near $\mathbf{p}$. The isolated point $\mathbf{p}$ is then inserted into this newly generated hole as shown in Fig. 7. The hole is chosen in a way so that the mean curvature of the resulting triangulation is minimized on the condition that this triangulation still has manifold properties. The mean curvature $\overline{C}$ of a triangulation $\mathbf{T}$ is given by

$$\overline{C}(\mathbf{T}) = \sum_{e \in E} \alpha_e \cdot \|e\| \ , \tag{7}$$

where $E$ is the set of all edges in $\mathbf{T}$, $\|e\|$ is the length of the edge $e$ and $\alpha_e$ is the dihedral angle of the two triangles adjacent to the edge $e$.[13]

If there is more than one isolated point, we insert them sequentially in arbitrary order.

Note that there are triangulations, where isolated points cannot be inserted with the method presented in this section. However none of these rather theoretical

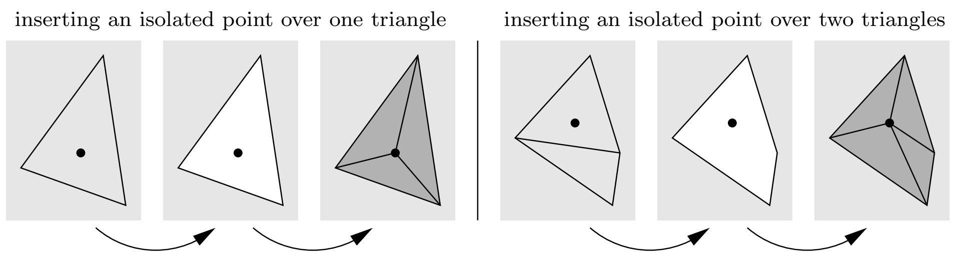
inserting an isolated point over one triangle
inserting an isolated point over two triangles

Fig. 7. Inserting an isolated point over one and two triangles respectively.

constructs (triangulations where no triangle and no pair of adjacent triangles is completely visible from the point to be inserted) has ever been observed in our tests.

### 4.2. *Smoothing the mesh*

The qualities of the resulting triangle meshes can further be improved by using a smoothing algorithm proposed by Dyn et al.[13]. 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 mesh repair algorithm to many different, incomplete or imperfect triangulations. To generate the imperfect input triangulations, shown in Fig. 9 and 10, we used Gopi's algorithm.[4] Gopi's algorithm is an efficient triangulation algorithm, based on lower dimensional localized Delaunay triangulation. It guarantees a manifold triangle mesh if the given point cloud fulfills a certain sampling criterion (the sampling density has to be inversely proportional to the curvature of the surface of the corresponding object) but it usually results in an incomplete or imperfect triangulation otherwise. The latter is the case for most small point clouds and for all point clouds, obtained by sampling surfaces with sharp edges and corners.

The number of triangles of our imperfect input triangulations is ranging from less than 200 to more than 130,000. All our tests resulted in correct manifold triangle meshes. Since we assured the convergence of our method in Section 3.3.6, this fact is not very surprising. Even heavily damaged triangulations could be repaired and extended to manifold triangle meshes without loosing their shape (Fig. 9 and 10 and Table 1).

Since simulated annealing is a probability based optimization technique, usually our algorithm generates different manifold triangulations, if applied to the same imperfect input mesh more than once. This is an advantage compared to other deterministic methods, because the user can simply restart the repairing process, if the resulting triangulation is not satisfactory. The user can influence the results by

Table 1. Statistics of the examples in Fig. 9 and 10.

| model | #ve | #tr | #bt | #oe | #ip | $t_{pre}$ | $t_{sim}$ | $t_{post}$ |
|---|---|---|---|---|---|---|---|---|
| bunny (small) | 487 | 1,167 | 435 | 241 | 3 | 0 | 2 | 0 |
| bunny (medium) | 7,958 | 16,986 | 2,209 | 923 | 0 | 9 | 3 | 2 |
| bunny (large) | 66,562 | 133,653 | 871 | 153 | 0 | 109 | 8 | 19 |
| tetrahedron[a] | 1,000 | 2,239 | 585 | 380 | 6 | 0 | 10 | 1 |
| unitcube[b] | 1,000 | 3,567 | 2,241 | 1,079 | 187 | 1 | 101 | 9 |
| foot | 79 | 173 | 63 | 65 | 1 | 0 | 0 | 0 |
| beethoven | 2,515 | 6,065 | 1,943 | 751 | 15 | 2 | 15 | 3 |
| horse | 9,264 | 19,629 | 2,941 | 1,585 | 7 | 14 | 25 | 6 |

*Note*:
#ve = number of vertices
#tr = number of triangles in the initial triangulation
#bt = number of bad triangles in the initial triangulation
#oe = number of open edges in the initial triangulation
#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
[a]This point cloud was obtained by randomly sampling 1000 points on the surface of a tetrahedron.
[b]This point cloud was obtained by randomly sampling 1000 points inside the unitcube.

specifying appropriate parameters $(c_1, c_2, T_1, s, t)$ for the simulated annealing stage. Especially the parameters $c_1$ and $c_2$ are suited to control the ratio between a fast repairing process and a high quality result. Fig. 8 shows typical output meshes of our algorithm where every mesh is the result of a different run (meshes on the left: $c_1 = 1.0$, $c_2 = 0.2$, i. e., more weight on a fast repairing process; meshes on the right: $c_1 = 1.0$, $c_2 = 5.0$, i. e., more weight on high quality results).

To further confirm that our mesh repair algorithm is able to create a manifold triangulation from any input triangle mesh, no matter how heavily damaged it is, we randomly sampled 1,000 points inside the unitcube. Of course the initial triangulation with Gopi's algorithm contains many errors (2,241 bad triangles, 1,079 open edges) and has no structure at all (there is no closed manifold triangulation of reasonable shape for a point cloud like that). Nevertheless our algorithm was able to generate a manifold triangulation (Fig. 10, top line).

Our algorithm proved to be quite efficient as well. For most big triangle meshes ($\geq 100,000$ triangles) the simulated annealing takes significantly less time than the algorithm providing the initial triangulation (the corresponding point clouds were densely sampled and only small regions are not fulfilling the sampling criterion and have to be repaired). For small triangle meshes ($\leq 1,000$ triangles) the simulated annealing takes proportionally much more time but the whole process of triangulating and repairing is only a matter of seconds (Table 1; all times were measured on an AMD Athlon 1200 MHz with 1 GB RAM; the simulated annealing parameters

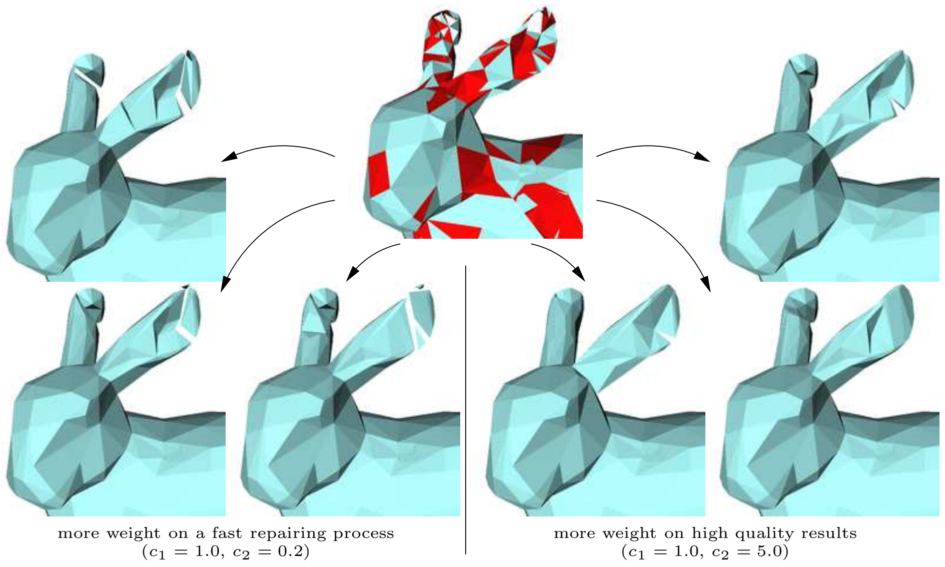
more weight on a fast repairing process
($c_1 = 1.0$, $c_2 = 0.2$)
more weight on high quality results
($c_1 = 1.0$, $c_2 = 5.0$)

Fig. 8. Typical results for different parameters $c_1$ and $c_2$ and different runs of our algorithm (bunny, 487 vertices, 1,167 triangles, 435 bad triangles, 241 open edges).

were chosen according to Section 3.3.5).

## 6. Summary and 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 a closed manifold triangulation. In contrast to other deterministic mesh repair 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 and 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 and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Symposium on Computational Geometry*, pp. 213-222, 2000.
3. T. K. Dey, S. Funke and E. A. Ramos. Surface Reconstruction in almost Linear Time

under Locally Uniform Sampling. *European Workshop on Computational Geometry, Berlin*, 2001.

4. M. Gopi, S. Krishnan and C. T. Silva. Surface Reconstruction based on Lower Dimensional Localized Delaunay Triangulation. *Computer Graphics Forum*, vol. 19, no. 3, pp. C467-C478, 2000.
5. G. Barequet and M. Sharir. Filling gaps in the boundary of a polyhedron. *Computer-Aided Geometric Design*, vol. 12, no. 2, pp. 207-229, 1995.
6. S. M. Morvan and G. M. Fadel. IVECS: An interactive virtual environment for the correction of .STL files. *Conference on Virtual Design, University of California, Irvine, CA*, 1996.
7. B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. *Computer Graphics (SIGGRAPH '96 Proceedings)*, vol. 30, pp. 303-312, 1996.
8. G. Barequet and S. Kumar. Repairing CAD Models. *IEEE Visualization*, pp. 363-370, 1997.
9. G. Barequet, C. Duncan and S. Kumar. RSVP: A Geometric Toolkit for Controlled Repair of Solid Models. *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 2, pp 162-177, 1998.
10. J. El-Sana and A. Varshney. Topology Simplification for Polygonal Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 2, pp. 133-144, 1998.
11. U. Adamy, J. Giesen and M. John. New Techniques for Topologically Correct Surface Reconstruction. *IEEE Visualization*, pp. 373-380, 2000.
12. J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum and T. R. Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. *Computer Graphics (SIGGRAPH '01 Proceedings)*, pp. 67-76, 2001.
13. N. Dyn, K. Hormann, S. J. Kim and D. Levin. Optimizing 3D triangulations using discrete curvature analysis. *Mathematical methods for curves and surfaces*, pp. 135-146, 2000.
14. T. Möller. A Fast Triangle-Triangle Intersection Test. *Journal of Graphics Tools*, vol. 2, no. 2, pp. 25-30, 1997.
15. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087-1092, 1953.
16. S. Kirkpatrick, C. Gelatt Jr. and M. Vecchi. Optimization by Simulated Annealing. *Science*, vol. 220, pp. 671-680, 1983.
17. L. L. Schumaker. Computing Optimal Triangulations Using Simulated Annealing. *Computer Aided Geometric Design*, vol. 10, no. 3-4, pp. 329-346, 1993.

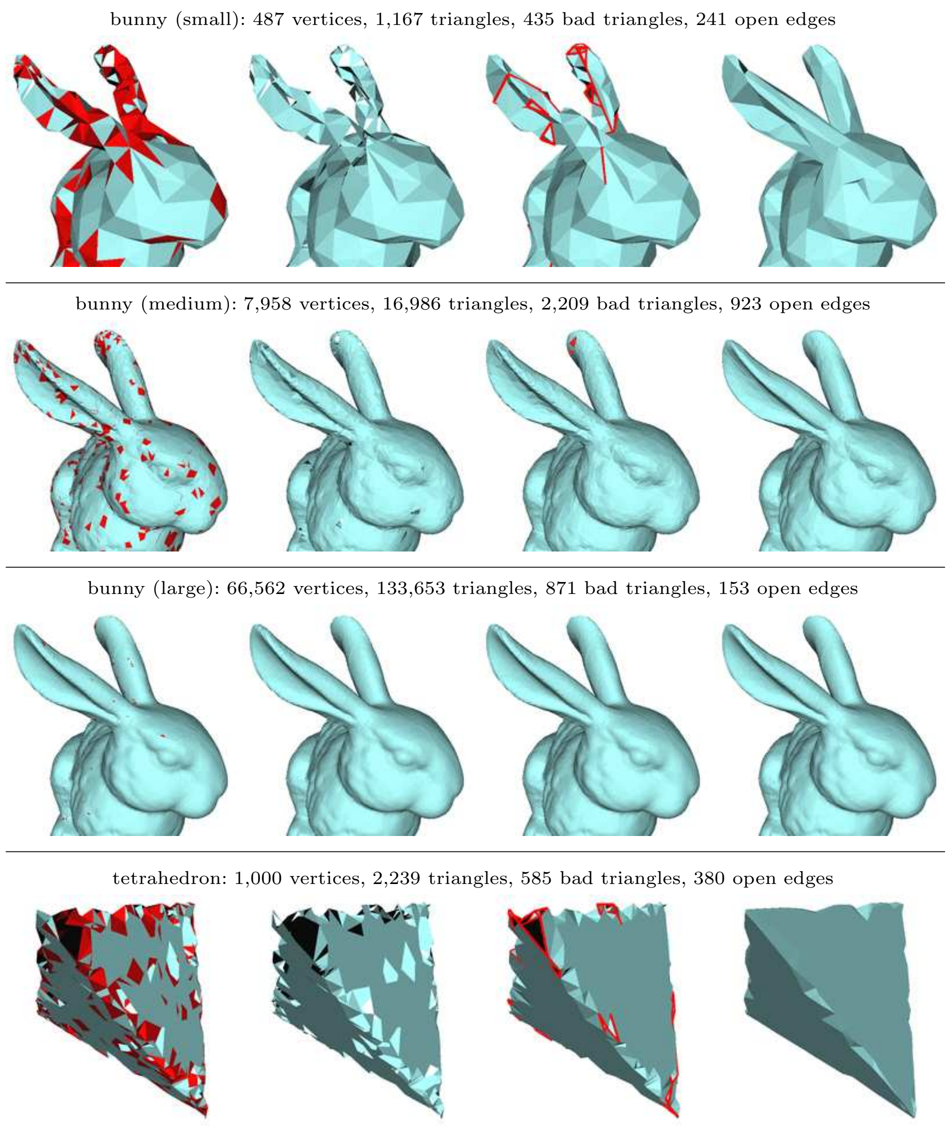
bunny (small): 487 vertices, 1,167 triangles, 435 bad triangles, 241 open edges
bunny (medium): 7,958 vertices, 16,986 triangles, 2,209 bad triangles, 923 open edges
bunny (large): 66,562 vertices, 133,653 triangles, 871 bad triangles, 153 open edges
tetrahedron: 1,000 vertices, 2,239 triangles, 585 bad triangles, 380 open edges

Fig. 9. 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. The simulated annealing parameters were chosen according to Section 3.3.5.

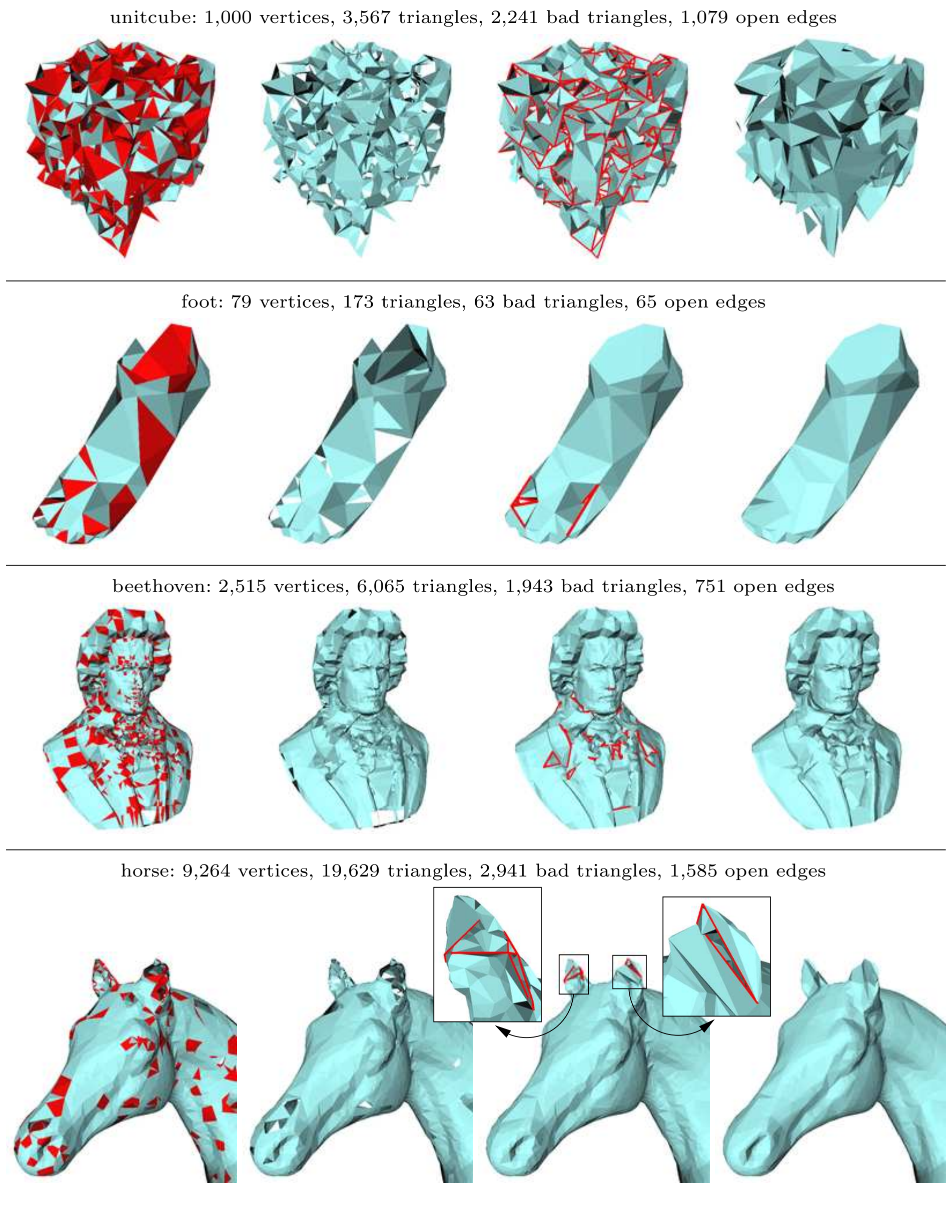
unitcube: 1,000 vertices, 3,567 triangles, 2,241 bad triangles, 1,079 open edges
foot: 79 vertices, 173 triangles, 63 bad triangles, 65 open edges
beethoven: 2,515 vertices, 6,065 triangles, 1,943 bad triangles, 751 open edges
horse: 9,264 vertices, 19,629 triangles, 2,941 bad triangles, 1,585 open edges

Fig. 10. 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. The simulated annealing parameters were chosen according to Section 3.3.5.