Topological Noise Removal

Igor Guskov Zoë J. Wood

Caltech

Abstract

Meshes obtained from laser scanner data often contain topological noise due to inaccuracies in the scanning and merging process. This topological noise complicates subsequent operations such as remeshing, parameterization and smoothing. We introduce an approach that removes unnecessary nontrivial topology from meshes. Using a local wave front traversal, we discover the local topologies of the mesh and identify features such as small tunnels. We then identify non-separating cuts along which we cut and seal the mesh, reducing the genus and thus the topological complexity of the mesh.

Key words: Meshes, irregular connectivity, topology.

1 Introduction

Acquisition of computer models with highly detailed geometry is currently practical due to developments in laser range finder technology. Raw irregular meshes coming from model acquisition contain millions of triangles, and require efficient processing tools. Such data is typically converted (or *remeshed*) into a more efficient and "regular" representation such as NURBS or other spline/subdivision-based multiresolution surface representations [22][18] [21][25][24]. Several remeshing methods use simplification hierarchies of the initial irregular mesh in order to build efficient computational procedures. However, raw irregular meshes extracted from noisy volumetric data often have small tunnels and handles: artifacts of the acquisition process. We present an algorithm to eliminate such topological "noise", greatly improving the construction of the simplification hierarchy and thus in turn, improving the final remeshed model.

Due to the large size of scanned geometry, the manual removal of artifacts is a tedious and time-consuming task; we would like to automate this process as much as possible. However, models such as mechanical parts potentially have important non-trivial topology (holes, handles, tunnels, etc.). One therefore needs a clear criteria to discern which tunnels can be safely removed algorithmically. The contribution of this paper is to introduce a simple criteria for identifying topological noise, and a fast algorithm that finds small tunnels in the data, and removes them one by one. The user can control criteria to help determine which tunnels are noise and which are inherent to the model. In addition, we show that the performance of the naive implementation of our topology filtering algorithm can be significantly improved by a preprocessing step.

Figure 1: *Scanned meshes from Stanford 3D model repository [26]. All three meshes are valid 2-manifolds: the Buddha has genus 104, the dragon has genus 46, and David's head has genus 340. Most of these tunnels/handles are noise and can be safely removed.*

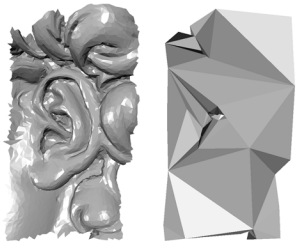
To demonstrate our approach we apply our technique to a variety of the Stanford laser range finder datasets. For example, we consider the dataset of the David's head from Stanford's Michelangelo project [26]. The original irregular mesh has genus 340. Obviously, none of these 340 tiny tunnels are actually present in the original sculpture, therefore all these tunnels (or handles) can be removed to facilitate further processing tasks. An irregular mesh of David's head containing more than a million tri-

angles is processed by our algorithm in one hour, removing 313 (92%) of the tunnels automatically. Complete filtering can also be made efficient using a combined filtering and simplification approach (see section 3 for more details).

1.1 Setting

The main application of our method is the processing of meshes coming from 3D model acquisition such as laser scanning. During acquisition, a complex model is often built from several scans. A number of popular mesh reconstruction methods [9][31] [19] combine several range maps using an auxiliary volumetric representation: a signed distance volume constructed from a collection of scans. An isosurface is then extracted using the Marching Cubes algorithm [27]. The result is an irregular mesh that is a proper manifold with boundary. The data coming from the scanner can be noisy and incomplete, hence the topological noise in the signed distance volume.

While the manifold property of the extracted surface can be guaranteed with small modifications to the original Marching Cubes algorithm[23], we observe that for noisy data it still produces topological artifacts such as tiny handles. It is often important to remove these handles so that they do not encumber later processing, such as simplification [20], smoothing, denoising [10], and parameterization for texture mapping and remeshing. Figure 6 shows the result of applying a smoothing procedure to a mesh with handles. While most of the surface gets smoother, the areas containing handles have visible artifacts. These handles will also complicate parameterization and for example, require unnecessary seams for texture mapping.



Topological artifacts will also cause problems for simplification algorithms that assume that the original mesh is a proper manifold with a boundary and preserve the genus of the surface for each simplification step. The inset on the left shows an example of such "poor" simplification. Such simplification is often used as an integral part of some larger multiresolution processing procedure that may rely on the topological equivalence of meshes on different levels of the hierarchy [25][8][17] [35][18].

1.2 Related work

A variety of researchers have relied on coding or matching the topology of a given mesh to a new configuration [29][3][4]. Most recently a lot of attention has been directed towards general simplicial complexes. Specifically, the problem of preserving the topology of simplicial complexes while applying edge contractions was considered by Dey et al. [34]. The recent paper by Edelsbrunner et al.[11] considers topological simplification in the context of alpha complexes. It is worth noting that topological simplification of simplicial complexes in $\mathbf{R}^3$ is a much harder and less intuitive problem than the one we consider.

El-Sana and Varshney [12][13] address a similar problem of controlled topology simplification for polygonal models. Their approach identifies removable tunnels by rolling a sphere of small radius over the object and filling the tunnels that are not accessible. The method performs well for mechanical CAD models. The interaction between mesh and topology simplification is also considered. Our approach is different in that it identifies tunnels by working within the surface, and thus can be applied to self-intersecting meshes as long as they are topologically 2-manifolds. Also, the focus of our work is to identify very small tunnels in noisy meshes.

Work has been done by Stander et al [33] on using critical points from Morse theory to guarantee the topology of the polygonization of an implicit surface. It is difficult to generalize this work to the irregular mesh setting without becoming computational intensive (see [6] for a potential solution). We focus on discrete methods that can rapidly discover the topology.

Recent work by Wood et al [37] presented an algorithm to quickly identify and reconstruct the topology of a surface implicitly represented in a volume. This work uses a wave front traversal in order to identify the global topology of the surface. The algorithm presented here has similarities but is generalized to the mesh setting with optimizations to discover small local topology and with optimizations to identify topological events. This work is closely related to work done by Axen [7] which relates a discrete wavefront traversal and critical points from Morse theory. However, neither of these approaches are used to filter the topology of the surface and thus do not present techniques to discover separating cuts nor change the genus of the surface.

It is worth noting that there is another way to potentially "filter" or smooth the noisy topology of scanned data by smoothing/down-sampling the initial volume data. Although this approach may remove many of the small tunnels present in the data, it will do so in an uncontrolled manner and will potentially wipe out other features of the model (thin tubes and connected components could be broken apart and the finer detailed geometry will disappear). Recent work by Gerstner and Pajarola [16]

on topology preserving volume simplification is one potential solution to try to control the effect of the downsampling, however, presently this work offers no method to distinguish important topology inherent to the model (such as a large handle) and small tunnels.

Finally, a great deal of work has been focused on simplifying meshes in general. Work by both Popovic and Hoppe [30] and Garland and Heckbert [15] could be applied to simplify "away" the small noisy tunnels present in the scanned meshes. However, in our work we seek more explicit topology changes that can be potentially adapted in a multiresolution processing algorithms such as MAPS [25].

1.3 Overview of the algorithm

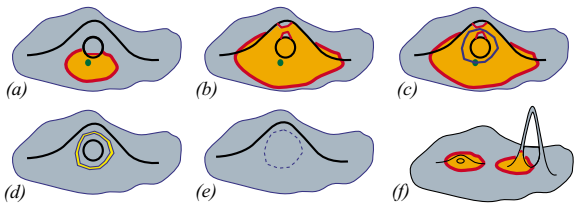


Figure 2: *(a-e) Overview of the algorithm: (a) a small region is grown around a seed face; (b) the genus of the grown region becomes non-zero; (c) a non-separating cut is found; (d) the mesh is cut; (e) both new holes are sealed. (f) The left handle is "fully inside" a ball of a small radius; the right handle is not. Note that both handles could be eliminated by short cuts. Our algorithm will only remove the left handle. (Formally, the left highlighted region is of genus one, while the right highlighted region is of genus zero.)*

We follow an approach similar to the ones presented in Wood et al. [37] and Axen et al.[7]. We grow an open region by adding faces one by one, while explicitly maintaining the active front edges. Every time a boundary component of the growing region touches itself along an edge, we split this boundary into two smaller boundary fronts and continue propagation. This results in a tree of active front components. Whenever boundaries of two different components touch along an edge, we claim to have found a handle. There are two stopping criteria for our region growing procedure – we either exhaust all the faces that are closer than some given radius from the seed face, or we actually find a handle in which case the growing stops and the mesh is cut along a non-separating curve. This operation does not change the connectedness of the surface but does reduce its genus introducing two new holes (boundaries), which are later triangulated using methods described in [32][25], or commercial packages [1][2]. (Figure 2 illustrates the entire process.) In this way, we remove the small handles one by one, filtering the local topology of the mesh.

2 Algorithm

We consider a triangular mesh $\mathcal{M} = (\mathcal{K}, \mathbf{x})$ where $\mathcal{K} = \mathcal{V} \cup \mathcal{E} \cup \mathcal{F}$ is an abstract simplicial complex representing the connectivity of the mesh ($\mathcal{V}$, $\mathcal{E}$, and $\mathcal{F}$ are sets of vertices, edges, and faces, correspondingly), and $\mathbf{x} : \mathcal{V} \to \mathbf{R}^3$ is the coordinate function that gives the coordinates of every vertex of $\mathcal{V}$. $\mathbf{x}$ can be extended to the polytope $|\mathcal{K}|$ of $\mathcal{K}$ using barycentric coordinates [28]. In this paper the focus is on meshes extracted as isosurfaces of certain volumetric functions, and therefore, such meshes are guaranteed to be orientable manifolds. Thus, all meshes considered in this paper are presumed to be orientable manifolds with boundary. Topology of such surfaces is easily characterized by their genus.

2.1 m-Closures

Our interest lies in finding "small" tunnels in the mesh, where the "smallness" will be defined later. Thus, we need to characterize topological properties of local regions of the mesh. For example, given a collection of faces $T = \{t_1, t_2, \ldots, t_k\}$ we would like to explore topological properties of the surface region defined by this set of faces. One way to approach this characterization would be to find the closure $\bar{T}$ in $\mathcal{K}$, and look at its properties. Note that the closure $\bar{T}$ for arbitrary T may not have the manifold property anymore, see Figure 3 for example. It is in fact a subcomplex of $\mathcal{K}$ and can be characterized as a general 2-complex, see [36]. However, that characterization is far too general for our purposes here. We therefore introduce a different "closure" operation that for a mesh region builds a corresponding mesh that is a manifold with boundary, and as such can be easily described by its genus. We call this operation *manifold closure* or simply *m-closure*, defined as follows.

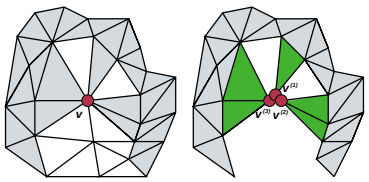


Figure 3: *Non-manifold closure ($N_v = 3$) is fixed by making three copies of the vertex v.*

First, note that Figure 3 represents the only way that $\bar{T}$ can be non-manifold. Moreover, it can be fixed with the following procedure (see Figure 3 for an illustration): for every non-manifold vertex $v \in \bar{T}$ its star neighborhood in $\bar{T}$ can be written as union of a number of semi-stars $H_v^{(i)}$: $St_{\bar{T}} v = \bigcup_{i=1}^{N_v} H_v^{(i)}$, where $\bigcap_{i=1}^{N_v} H_v^{(i)} = \{v\}$, and each semi-star is of the form: $H_v^{(i)} = \{\{v\}, \{v, u_0\}, \{v, u_0, u_1\}, \ldots, \{v, u_{k-1}, u_k\}, \{v, u_k\}\}$.

We define the *m-closure* of T in $\mathcal{K}$ as the mesh obtained from $\bar{T}$ by splitting every non-manifold vertex v with N_v

adjacent semi-stars into N_v vertices $v^{(1)}, \ldots, v^{(N_v)}$, and replacing each occurrence of v in the simplices of $St_{\bar{T}}v$ by the appropriate new vertex depending on which semi-star they belong to. We denote the resulting mesh as $\bar{m}T$. Note that the interiors of m-closure and usual closure coincide: $int\,\bar{T} = int\,\bar{m}T$.

2.2 Small tunnels

It is now necessary to define which tunnels need to be removed. For that purpose, we consider the dual graph $(\mathcal{F}, \mathcal{E}')$ of the mesh $\mathcal{M}$ where a dual edge (t_1, t_2) between two faces of the mesh is in $\mathcal{E}'$ if t_1 and t_2 share a (primal) edge in the triangulation. If some non-negative weight function w is defined on $\mathcal{E}'$, we can now define the distance $d(s,t)$ between any two faces s and t as the minimal sum of weights over all the paths in the dual graph. One easy example is given by setting $w(e') = 1$ for every $e' \in \mathcal{E}'$. It is also possible to make weights that would approximate geodesic distances on a manifold. In this paper we use $w \equiv 1$.

Now we can give the general principle that we use to remove small tunnels:

ε-simple meshes
Mesh $\mathcal{M}$ is ε-simple if for every face $t \in \mathcal{F}$ the m-closure of dual ε-ball $\bar{m}\{s : d(s,t) < \varepsilon\}$ is of genus zero.

Our goal therefore becomes to convert a given mesh into an ε-simple mesh. This can be done by finding closed cuts that leave the mesh connected. Each such cut will reduce the genus of the surface by one. In the following sections, we introduce an algorithm to find such non-separating closed cuts (a cut is non-separating if it leaves the surface connected [5].) These cuts will be found inside the corresponding ε-balls; note however that such short non-separating cuts can exist in meshes that are ε-simple for small ε, such as the ones containing long narrow handles, see Figure 2(f). However, it is not clear that such long handles should be automatically removed. In our approach, we will only find cuts corresponding to handles that are completely contained in small regions of the mesh.

The size for ε varies depending on the input data and the relative size of tunnels to be filtered. For example, in practice we found that values ranging from four to twelve were appropriate for input models ranging from 184K to 4,000K faces.

2.3 Region growing

In this section we describe an algorithm that looks for tunnels in the neighborhood of a seed face. Later, in Section 2.6 we explain a global search for tunnels that will use this local procedure as an elementary operation.

The local procedure starts with a seed face $t_{seed} \in \mathcal{F}$. The faces from the ε-ball around t_{seed} are considered one by one in the order produced by using Dijkstra's algorithm on the dual graph. Thus, a sequence $t_1, t_2, \ldots, t_k$ is constructed. We define the i-th active region as $A_i(t_{seed}) := \bar{m}\{t_{seed}, t_1, \ldots, t_i\}$ for $i = 1, \ldots, k$. Algorithmically, the active region is grown one face at a time, while the explicit representation of active boundaries is maintained. Every time a new face is added, we check the genus of the resulting active region. The process starts with one triangle which is obviously of genus zero. We then proceed either until all the faces of the ε-ball are exhausted, or until we find that after the current triangle is added, the genus of the active region has grown. If the latter happens, the region growing stops and a non-separating closed cut is found inside the active region. We then cut the mesh (possibly locally subdividing it), seal the two resulting holes, and start with the current seed face again. Thus, the small tunnels in the mesh are extinguished one by one.

We now describe the particulars of maintaining the active region and tracking its genus.

2.4 Evolution of the active region

Suppose the active region A is given and another face t needs to be added to it. By construction, $A \cap t$ contains an edge. The change of active region is performed using the following three operations: *add-triangle*, *close-crack*, and *merge-edge*[1] . We describe these operations below in more detail.

Add-triangle
We assume that the active region and the new incoming triangle share at least one common edge. Then the *add-triangle* operation adds the triangle to the active region by merging across a common edge. The resulting mesh has one more face, two more edges, and one more vertex than the original one (see Figure 4(a)). The number of boundary components does not change. Thus, the genus of the corresponding mesh region does not change. Indeed,

$$\begin{aligned} \chi_{new} &= V_{new} - E_{new} + F_{new} + H_{new} \\ &= (V_{old} + 1) - (E_{old} + 2) + (F_{old} + 1) + H_{old} \\ &= \chi_{old}. \end{aligned}$$

Since the genus of the region is $g = 1 - \chi/2$, and χ is unchanged, the genus of the current mesh region is preserved during the *add-triangle* operation.

In order to find the non-intersecting cut later, each face stores a pointer to the face to which it was added. To set

[1]Note that there is no need for a *merge-vertex* operation (when a single vertex is adjacent to more than two boundary edges) due to *m-closure*.

up the notation, let t be the new face and $t' \in A$ be a face from the active region that shared a common edge with t. We call t' the *parent of* t, or $t' = parent(t)$.

Close-crack

Once the new triangle is added to the mesh we need to resolve possible self-adjacencies along the boundary. One local inconsistency is depicted in Figure 4(b). We fix the boundary locally by eliminating two boundary edges. The resulting mesh has one less edge, and one less vertex than the original one. The number of faces and boundary components does not change. Thus, the genus of the corresponding mesh does not change. Again,

$$\begin{aligned} \chi_{new} &= (V_{old} - 1) - (E_{old} - 1) + F_{old} + H_{old} \\ &= \chi_{old}. \end{aligned}$$

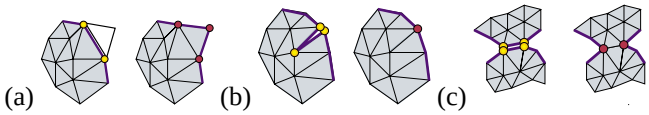


Figure 4: *(a)* add-triangle *operation; (b)* close-crack *operation; (c)* merge-edge *operation. Current mesh region shown in gray, with its boundary in blue.*

Merge-edge

The last operation required to maintain a consistent active region is not local, in that it requires adjacency tests between different parts of the boundaries, or even between different boundary components. Indeed, the *close-crack* operation cannot resolve situations such as the one shown in Figure 4(c). Here two edges lying on two separate pieces of the boundary of the current region correspond to the same edge of the original mesh. We fix this inconsistency by merging the current region(s) across this edge. As a result the number of boundary components will either increase by one (when the merged edges belong to the same boundary component), or decrease by one (when two different boundary components become one). Note that these two cases closely correspond to the topological events described in [37], when a region of the active edge front either splits into two when a handle in the surface is encountered, or when it merges back into a single front at the other side of the handle. The *merge-edge* operation results in one less edge and two less vertices for the active region, and the number of faces does not change. Depending on the value of the change in the number of boundary components we will encounter two cases:

A boundary splits.

$$\begin{aligned} \chi_{new} &= (V_{old} - 2) - (E_{old} - 1) + F_{old} + (H_{old} + 1) \\ &= \chi_{old}. \end{aligned}$$

Boundaries merge.

$$\begin{aligned} \chi_{new} &= (V_{old} - 2) - (E_{old} - 1) + F_{old} + (H_{old} - 1) \\ &= \chi_{old} - 2. \end{aligned}$$

In this last case the genus of the active region increases by one. When this final case is detected, we proceed by performing a non-separating cut, thus reducing the genus by one.

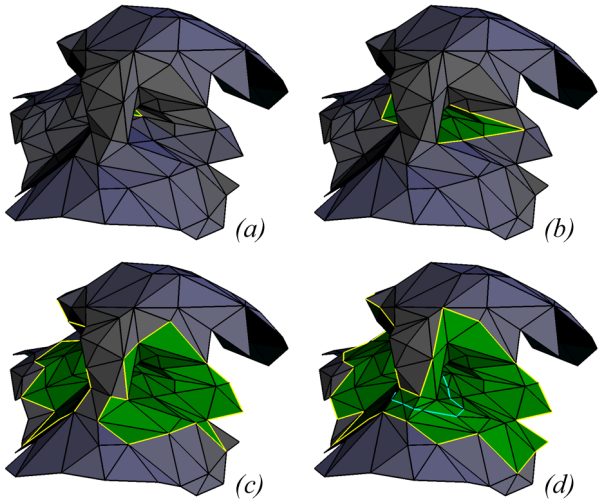
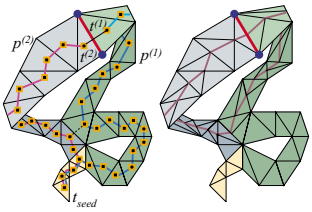


Figure 5: *Running the algorithm locally: (a) the active region is seeded with a single face; (b) propagation has started; (c) the active region has two boundary components; (d) two boundary components have merged and a non-separating cut is found in a locally subdivided mesh.*

2.5 Cutting the mesh



In this section, we describe how a non-separating cut is found inside the active region after a merge-edge operation has merged two boundary components. Suppose that the two boundaries merged along the edge $e_M = \{v^{(1)}, v^{(2)}\} = t^{(1)} \cap t^{(2)}$. We build two sequences of faces, $p^{(1)}$ and $p^{(2)}$, defined as $p^{(j)} = (t_1^{(j)}, \ldots, t_{K^j}^{(j)})$, where $t_{k+1}^{(j)} = parent(t_k^{(j)}), j = 1, 2$. Note that both of these face paths end at the original seed face which has no parent. After excluding a common tail of these two paths we have a closed path in the dual graph of the active region. It is then possible to subdivide the faces on this closed path so that there is a closed cut along the edges of this locally subdivided mesh which does not intersect itself, see figure below. Note that this path is *completely inside the interior of the current active mesh region.*

We can also prove that this cut is non-separating, that is, it leaves the active mesh region (and hence the mesh itself) connected. In order to prove that we simply notice that the two vertices $v^{(1)}$ and $v^{(2)}$ lie on the different sides of the cut locally but we can reach $v^{(2)}$ from $v^{(1)}$ by following the boundary of the current active region (we can do that because the cut is fully inside the active region and thus does not touch the boundary). We also further reduce the length of the cut, by using reductions similar to the one shown in above figure. During these reductions we do not allow faces $t^{(1)}$ and $t^{(2)}$ to disappear, therefore the argument above still holds. We then seal these two new gaps in the mesh, and thus remove the handle. Figure 5 illustrates the process on a fragment of a real mesh.

The subdivision performed during the cut computation changes distances in the dual graph. We fix this problem by assigning zero weights to the new edges introduced during subdivision (of course, the dual edges corresponding to the edges in the cut itself simply disappear from the dual graph of the modified mesh.)

2.6 Global procedure and preprocessing

In the previous section we described a procedure that grows a mesh region of some radius $\varepsilon > 0$ centered at a seed face and removes all the tunnels that are discovered inside this mesh region one by one. We can run this procedure starting from all the faces in the original mesh. This will produce a mesh that is ε-simple. However, as ε grows the running times of this naive algorithm become unacceptable. We propose a preprocessing step that excludes large portions of seed faces from the consideration. We rely on the following fact which is true in a metric space. Let $B_R(t_0)$ be the closed ball of radius R centered at t_0 (note that we measure the distances on the surface, so in our case, a ball is a surface region.) Then for any $t' \in B_{R-\varepsilon}(t_0)$ the ball centered at t' of radius ε is contained in $B_R(t_0)$, in fact, $B_\varepsilon(t') \subset B_R(t_0)$. Therefore, in the preprocessing step we will be growing balls until their genus changes, without any restriction on their radius. Suppose that we have grown a mesh region A that includes the ball $B_R(t_0)$ for some $R > \varepsilon$, and the genus of A is zero. Then we can be assured that any subset of A will also be of genus zero, and since the balls of radius ε centered inside the smaller region $B_{R-\varepsilon}(t_0)$ are subsets of A, we can exclude them from the potential seed set. These large regions are seeded in the preprocessing step at randomly chosen faces of the original mesh (in practice, taking one percent of the original number of faces produces good results). This procedure greatly reduces the potential seed set for a given ε. For example, without preprocessing, the algorithm takes 1147 seconds to perform filtering with radius 3 on the David's head model; while the improved procedure takes only 136 seconds. More performance numbers can be found in Table 1.

Dataset	Radius	Removed handles	Time
David's head I	8	241	35m 34s
4000K faces	10	264	1h 24m 43s
genus 340	12	283	3h 13m 30s
David's head II	8	291	12m 53s
1173K faces	10	301	27m 37s
genus 340	12	313	56m 52s
David's head III	8	323	4m 27s
184K faces	10	326	9m 36s
genus 340	12	330	19m 6s
David (complete statue)	8	12	34m 4s
8254K faces	10	13	45m 11s
genus 20	12	14	57m 43s
Buddha	8	71	10m 23s
1087K faces	10	82	34m 24s
genus 104	12	85	2h 43m 9s
Dragon	8	21	6m 4s
870K faces	10	32	16m 59s
genus 46	12	35	53m 3s
St.Matthew	6	3	21m 19s
3382K faces, genus 5	12	4	29m 37s

Table 1: *Timings given for Pentium III Xeon 550 MHz.*

3 Results

We have implemented the algorithm described in the preceding sections and performed various experiments reducing the topological noise for a number of meshes from the Stanford Archive and the Digital Michelangelo project [26]. We have found that most of the models reconstructed using Curless and Levoy's VRIP method [9] have topological artifacts. We noticed that higher resolution models and meshes that were more convoluted in shape typically have more topological noise. We have run our algorithm on models of different resolution with different threshold radius settings and recorded the number of tunnels removed and the algorithm's running time. These results are illustrated in Table 1. Note that this automatic technique fails to generate a genus zero surface when there are tunnels larger than the given ε. However, it is unclear whether tunnels of a certain size should indeed be filtered. In the future it would be useful to include a user interface to allow the user to inspect tunnels of a larger radius and determine if they should be filtered or retained.

We have applied various mesh processing techniques to meshes that have been topologically filtered using our algorithm with encouraging results. In particular, we were able to apply the multiresolution remesher of Guskov et al. [18] to the simplified genus zero mesh of David's head. The base mesh for this remesh contained

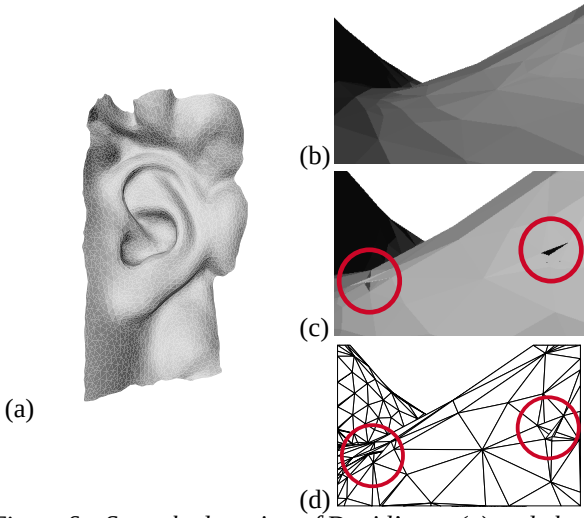
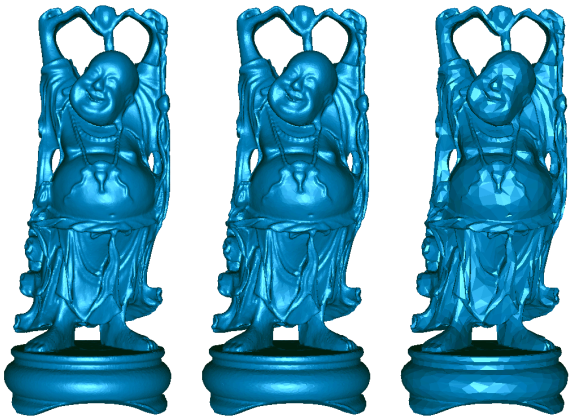


Figure 6: *Smoothed version of David's ear (a) and close up view of the smoothed ear after topology filtering (b) and a close up of the artifacts that occur without filtering (flipped triangles) (c) and a detailed view of the tunnels causing the artifacts (d).*

262 triangles. It would be impossible to achieve such a small number of patches without first applying a topology filtering operation to the original data (remember that the original mesh had 340 tunnels).

Similarly, parameterization of mesh regions is a fundamental part of many remeshing, texturing, and other mesh processing algorithms. The inset on the left shows the parameterized mesh region of the David's ear. The texture coordinates are assigned with the (u, v)-coordinates computed with the shape-preserving parameterization of Floater [14]. The original unfiltered region of this mesh contained twelve tunnels. Our algorithm removes all of these tunnels in fifteen seconds, and produces a mesh that is homeomorphic to a square, allowing it to be properly parameterized.

Additionally, acquired meshes often contain geometric noise, and have to be filtered with various mesh smoothing/noise removal techniques. In particular, we used the method described in Desbrun et al. [10]. If the original mesh contains unnecessary non-trivial topological artifact, the smoothing procedure typically results in a mesh with artifacts that foil its appearance (such as flipped triangles), as shown in Figure 6. This is due to the fact that smoothing operators cannot modify the topology of the mesh, and the presence of these small handles impairs the smoothing process by limiting its effects. Attempts to smooth the region around small tunnels can potentially result in collapsing the tunnel, creating undesirable degeneracies. Thus first removing the topological noise greatly improves the performance of geometric noise removal procedures, as illustrated by Figure 6.



1,088K faces genus 104 | 178K faces genus 15 | 16K faces genus 7

Figure 7: *Using both topology and geometry simplification on the Buddha mesh (see Table 7.) Note that all the meshes shown here are valid manifolds. The geometry was simplified with Raindrop Geomagic Studio [1].*

Finally, we have explored iterating between removing topological noise and applying topology preserving mesh simplification. Running these two processes alternatively decreases the amount of time to discover all the small tunnels on a given mesh. The results of such an iteration sequence are presented in Table 2 and Figure 7. It is clear that if the topology simplification is used as a part of a multiresolution technique such as remeshing, this gradual approach would be preferable for efficiency reasons. However, this process of simplification followed by filtering is not appropriate for all applications as some geometric details are lost during simplification. Thus, in order to retain the finest level geometric details in the filtered output, the more lengthy filtering algorithm with a larger ε, should be used. We leave the complete exploration of these ideas as future work.

4 Conclusions and future work

Topological noise is a serious problem for many scanned models. This noise results in visible artifacts when these meshes are smoothed, encumbers parameterization and hinders the performance of many multiresolution techniques. We have presented a simple criteria for identifying such topological noise and a computational procedure that removes these topological artifacts. The algorithm is very robust and is able to process extremely large meshes.

In this paper we have focused on removing the topological noise from the original resolution of the model and did not concern ourselves with larger scale genus changing operations. In fact most of the tunnels removed with our algorithm are in the very crooked parts of small regions of the mesh, and their removal does not affect the visual appearance of the model. It would be very interesting to explore genus changing operations in the multiresolution setting, perhaps directly within a mesh simplification or remeshing framework. Such a system would ideally include a user interface that allowed the user to select whether to filter handles of an ambiguous size. This would include work on visualizing the tunnels and the potential non-separating cut placement. Another exciting prospect for future work is the direct removal of topological noise from the original volume data.

Size(faces)	Genus before	Genus after	Time
1087K	104	33	623s
178K	33	15	94s
54K	15	11	64s
16K	11	7	32s

Table 2: *Multiple resolutions processing of the Buddha mesh. Mesh simplification was used to reduce the face count of the models between the topology filtering steps (see Figure 7.) Threshold radius was set to 8 for all runs.*

5 Acknowledgments

We would like to thank Mathieu Desbrun, Peter Schröder, and Andrei Khodakovsky for many useful discussions about this work and we'd like to thank the Stanford Computer Graphics Group for sharing their wonderful models with us.

6 References

[1] Geomagic studio 3.0, 2000. Raindrop Geomagic.

[2] Paraform 2.0, 2000. Paraform Inc.

[3] Ergun Akleman and Jianer Chen. Guaranteeing 2-manifold property for meshes. In *Proceedings of the International Conference on Shape Modeling and Applications*, 1998.

[4] Ergun Akleman, Jianer Chen, and Vinod Srinivasan. A new paradigm for changing topology of 2-manifold polygonal meshes. In *Pacific Graphics'2000*, 2000.

[5] P. Aleksandrov. *Combinatorial Topology*, volume 1. Graylock Press, 1956.

[6] U. Axen. Computing morse functions on triangulated manifolds. In *Proceedings of the SIAM Symposium on Discrete Algorithms (SODA)*, 1999.

[7] U. Axen and H. Edelsbrunner. Auditory morse analysis of triangulated manifolds. In H.-C. Hege and K. Polthier, editors, *Mathematical Visualization*, pages 223–236. Springer-Verlag, Berlin, Germany, 1998.

[8] Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, pages 115–122, 1998.

[9] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. *Proceedings of SIGGRAPH 96*, pages 303–312.

[10] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of SIGGRAPH 99*, pages 317–324, 1999.

[11] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *The 41st Annual Symposium on Foundations of Computer Science*, 2000.

[12] J. El-Sana and A. Varshney. Controlled simplification of genus for polygonal models. *Proceedings of the IEEE Visualization '97*, pages 403–412.

[13] J. El-Sana and A. Varshney. Topology simplification for polygonal virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):133–144, June 1998.

[14] Michael S. Floater. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14:231–250, 1997.

[15] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 96*, pages 209–216, 1996.

[16] Thomas Gerstner and Renato Pajarola. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. *Visualization* '00 Proceedings , 2000.

[17] Igor Guskov, Wim Sweldens, and Peter Schröder. Multiresolution signal processing for meshes. *Proceedings of SIGGRAPH 99*, pages 325–334, 1999.

[18] Igor Guskov, Kiril Vidimče, Wim Sweldens, and Peter Schröder. Normal meshes. *Proceedings of SIGGRAPH 2000*, pages 95–102, 2000.

[19] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Computer Graphics (SIGGRAPH 1992 Proceedings)*, pages 71–78, 1992.

[20] Hugues Hoppe. Progressive meshes. *Proceedings of SIGGRAPH 96*, pages 99–108, 1996.

[21] L. Kobbelt, J. Vorsatz, U. Labsik, and H-P. Seidel. A shrink wrapping approach to remeshing polygonal surfaces. *Computer Graphics Forum*, 18:119 – 130, 1999.

[22] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. *Proceedings of SIGGRAPH 96*, pages 313–324, 1996.

[23] J.-O. Lachaud. Topologically Defined Iso-surfaces. Research Report 96-20, Laboratoire de l'Informatique du Parallélisme, ENS Lyon, France, 1996.

[24] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Computer Graphics (SIGGRAPH 2000 Proceedings)*, pages 85–94, 2000.

[25] Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. *Proceedings of SIGGRAPH 98*, pages 95–104, 1998.

[26] Marc Levoy. The Digital Michelangelo project. In *Proceedings of the 2nd International Conference on 3D Digital Imaging and Modeling*, Ottawa, October 1999.

[27] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface reconstruction algorithm. In *Computer Graphics (SIGGRAPH 1987 Proceedings)*, pages 163–169, 1987.

[28] J. R. Munkres. *Elements of Algebraic Topology*. Addison-Wesley, Redwood City, 1984.

[29] Tamal Dey Nina Amenta, Sunghee Choi and Naveen Leekha. A simple algorithm for homeomorphic surface reconstruction. *ACM Symposium on Computational Geometry*, pages 213–222, 2000.

[30] Jovan Popovic and Hugues Hoppe. Progressive simplicial complexes. *Proceedings of SIGGRAPH 97*, pages 217–224, 1997.

[31] K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro, and W. Stuetzle. Robust meshes from multiple range maps. In *Proceedings of International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 205–211, May 1997.

[32] J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Workshop on Applied Computational Geometry (Philadelphia, Pennsylvania)*, pages 124–133. Association for Computing Machinery, May 1996.

[33] Barton T. Stander and John C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 279–286, August 1997.

[34] S. Guha T. K. Dey, H. Edelsbrunner and D. V. Nekhayev. Topology preserving edge contraction. *Publ. Inst. Math. (Beograd) (N.S.)*, 66:23–45, 1999.

[35] Gabriel Taubin, André Gueziec, William Horn, and Francis Lazarus. Progressive forest split compression. *Proceedings of SIGGRAPH 98*, pages 123–132, 1998.

[36] E. F. Whittlesey. Finite surfaces: a study of finite 2-complexes. *Math. Mag.*, pages 11–22 and 67–80, 1960.

[37] Zoë Wood, Mathieu Desbrun, Peter Schröder, and David Breen. Semi-regular mesh extraction from volumes. In *Proceedings of Visualization 2000*, 2000.