Mesh Simplification by Stochastic Sampling and Topological Clustering

Tamy Boubekeur* and Marc Alexa+

*Telecom ParisTech - CNRS *LTCI*
+TU Berlin

Abstract

We introduce TOPSTOC, a fast mesh simplification algorithm. The two main components are stochastic vertex selection and re-indexing of triangles. The probability for vertex selection depends on a local feature estimator, which prefers areas of high curvatures but still ensures sufficient sampling in flat parts. Re-indexing the triangles is done by breadth-first traversal starting from the selected vertices and then identifying triangles incident upon three regions. Both steps are linear in the number of triangles, require minimal data, and are very fast, while still preserving geometrical and topological features. Additional optional processing steps improve sampling properties and/or guarantee homotopy equivalence with the input. These properties provide an alternative to vertex clustering especially for CAD/CAM models in in the areas of previewing or network graphics.

Key words: Surface Simplification, Mesh Subsampling, Clustering, Stochastic Geometry Processing

1. Introduction

Mesh simplification is a fundamental tool in geometry processing. One may distinguish two different application scenarios:

- Constructing the best possible approximation of a given discrete surface, where the resulting mesh will be stored and reused, and the time spent on the simplification is of minor concern [1, 2, 3].
- Down-sampling a mesh instantly for preview on or transmission over limited devices, where the approximation is usually discarded after the task is performed, and it is critical that the mesh is provided instantly [4, 5].

We are tackling the latter problem and provide TOPSTOC, a solution that focuses on preserving topological and geometrical features better than similarly fast techniques (see Fig. 1). A more detailed view on related techniques and their relation to our approach is given in Section 2.

TOPSTOC is designed for cases in which simplification is not performed in a preprocess, but as part of a loop combined with other processing steps on data that is large and contains geometrical and topological features that would ideally be retained. Our approach is composed of two steps, which are each interesting in their own right as a tool in digital geometry processing:

1. The set of vertices for the simplified model is computed by sweeping over the input and selecting vertices probabilistically (see Section 3.1). The number of vertices can be prescribed and the selected vertices concentrate around features.

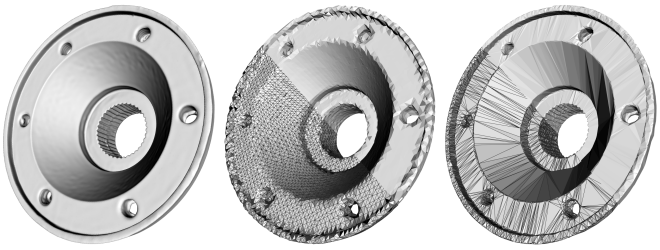


Figure 1: The carter model (1M triangles) is simplified in less than a second with grid vertex clustering (middle image) and our new simplification algorithm (on the right).

2. The triangles connecting the selected vertices are computed by first assigning all vertices in the original mesh to the topologically closest selected vertex and then identifying triangles in the original mesh that are incident upon vertices assigned to three different selected vertices (see Section 3.2).

This algorithm, in its basic setting, is linear in time and memory with respect to the number of input primitives. However, it still succeeds to provide an *adaptive* sampling of the mesh, resulting in large polygons in flat areas and smallest around features. Adaptivity is a major property in several of the typical scenarios we have designed TOPSTOC for, such as instant network graphics, where the input mesh is instantly simplified after processing or editing, to be broadcasted toward heterogeneous devices, with limited network bandwidth, memory and computational power. Additionally, we propose extensions of the basic algorithm that help achieving guarantees on the vertex distribution as well as on the topology of the simplified mesh but make the

algorithm super-linear (see Section 4) – these extensions can be used when necessary for the application.

We have implemented this approach and analyzed the distribution of resulting meshes (see Section 5). It turns out that the method is generally very fast, and that the feature-driven stochastic vertex selection provides significant improvements in visual quality over similarly fast techniques. We substantiate this claim by measuring errors in differentials of the mesh, rather than only Hausdorff distance. Generally, stochastic selection and topological clustering are interesting linear approaches with good global properties. We provide more detailed conclusions in Section 6.

2. Previous Work

Quality-oriented simplification was initiated by the seminal work of Hoppe [1], introducing an optimization approach with a set of local mesh operators, among which the *edge-collapse/vertex-split* pair also proved instrumental for multiresolution mesh representations [6]. Garland and Heckbert [7] introduced the Quadric Error Metric (or QEM) for driving the simplification process. This metric computes and stores the error/importance of a vertex as the sum of its squared distances to the supporting planes of its incident triangles. QEM can be enhanced to consider the color information on surfaces [2], sped-up using *random multiple choices* [8] or GPUs [9], adapted to point sampled surfaces [10] and has been proved to be geometrically optimal in certain cases [11]. It still represents state-of-the-art quality-oriented simplification methods. However, as keeping a priority queue can be prohibitive for very large models, Wu and Kobbelt [12] used a random selection of vertices to be removed in the context of streaming to simplify large meshes.

Alternatively, *iterative relaxation methods*, such as Turk's *mesh re-tiling technique* [13] or Cohen-Steiner's *variational surface approximation* [3], rely on a global non-linear optimization process to find the best mesh with N polygons subsampling the input shape. The function to optimize may vary according to the application. For instance, Cohen-Steiner *et al.*introduced the $L^{2,1}$ error metric which is based on normal vectors of primitives – for identifying and preserving anisotropic features. Considering normals has now been understood to be more important for the visual quality than geometric errors, raising the general question of perceptual considerations in simplification. One attempt of addressing this is related to *mesh saliency* [14].

The most prominent and to our knowledge first example of almost instant simplification is spatial vertex clustering proposed by Rossignac and Borrel [4]: 3D space is partitioned into cells and each non-empty cell is represented by a single vertex. Triangles belonging to three non-empty cells are re-indexed to the representative vertices, while the rest of the triangles are discarded. The original technique used a grid as spatial partitioning structure and the centroid of vertices contained in the cell as representative vertex. Lindstrom [5] used the optimal vertex defined by the QEM [2] inside the grid cell. The spatial partitioning has been improved to adapt to surface geometry, using either BSP [15], Octree [16], or VS-Tree [17]. The mesh can also be constructed progressively, by adding representative vertices until an error bound is satisfied [18].

One of the main features of vertex clustering is its speed, with the complexity usually being linear in the number of input primitives (every primitive has to be touched at least once) and potentially super-linear only in its output size. The simple process on the input allows performing most computations out-of-core so that models of several hundreds of millions of primitives could be handled [16]. Note that we have not yet implemented an out-of-core version of TOPSTOC, however, our algorithm builds on principles (e.g. local selection, breadth-first traversing) which can be adapted to out-of-core and streaming methods [19] [20] [21].

It seems despite the improvements to the original vertex clustering approach, when it comes to the application of instant simplification to large meshes, a regular grid for partitioning is the preferred choice because it avoids costly adaptive data structures. This, however, has consequences for the output:

- the size of the output depends on the number of occupied cells, which is not known in advance.
- a regular grid is insensitive to the salient features of the geometry and results in almost uniform triangle sizes.
- the topology depends on the triangles intersecting edges of the regular grid; different components might be connected or the genus be changed.

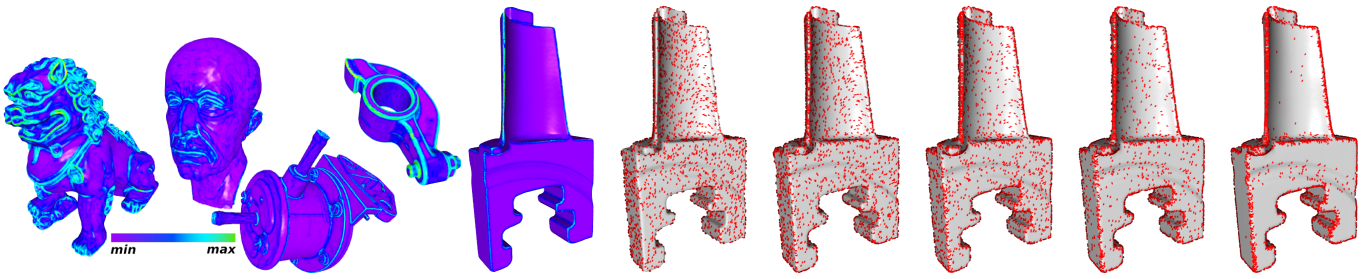
Depending on the application, these properties can be good or bad. In the context of interactively processing a large mesh, we feel it would be desirable to control the number of vertices, and be faithful to the geometric and topological features of the mesh.

TOPSTOC provides control over the size of the mesh, and retains features of the input at almost the same speed as clustering in a regular grid. One key idea is that stochastic sampling allows touching each vertex only once (and avoiding to partially order the vertex set) while introducing a feature sensitive vertex selector. The other observation is that clustering the topological (i.e. the surface) rather than the ambient space makes preserving topological features easier, while (perhaps surprisingly) being similarly fast in practice (see Section 3.2). Following Wu and Kobbelt [8], one can classify simplification methods roughly as follows:

	Clustering	**Incremental**
Deterministic	Spatial Clustering	Priority queue
Stochastic	TOPSTOC	Multiple-Choice

3. The Algorithm

The TOPSTOC algorithm is designed to operate on a simple triangle mesh $M = \{V, T\}$ with V the list of vertices and T the list of triangles indexed over V (standard representation in rendering engines of modeling tools). The main steps are selecting a subset $V' \subset V$ of size k, then partitioning V into k connected components covering the mesh, and identifying triangles $T' \subset T$ that belong to three different components in the



(a) Feature Estimator based on Normals

(b) Uniform to Adaptive Sampling Modulation

Figure 2: **Geometry-Aware Stochastic Sampling.**

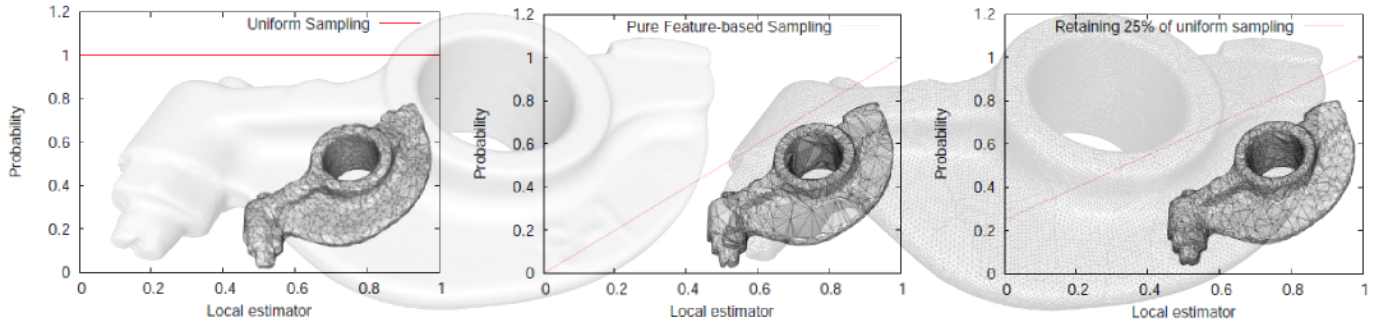


Figure 3: **Stochastic vertex sampling**. Uniform distributions (left) ignore geometric features. Purely adaptive (middle) ones miss slowly varying curved areas. Our choice offers a good trade-off in the vertex selection.

partition of V. The mesh $M' = \{V', T'\}$ is the output of the algorithm. Note that, in contrast to some variants of clustering, V' is a subset of V so one may view T' as an alternative triangulation of V, only with fewer triangles.

3.1. Stochastic Vertex Selection

Selecting the k "best" vertices to represent a mesh can be performed by attaching a "geometric importance" value to each vertex of V, and then ordering the vertices according to this value. We want to avoid this global approach, which inevitably results in super-linear asymptotic complexity and/or larger constants. Our idea is to approximate a global solution using tools from probability theory: *stochastic sampling* based on a probability distribution function.

Let $\mathscr{P} : [0,1] \rightarrow [0,1]$ be this probability distribution function. It defines the probability of the value x to appear in the stochastic sampling. In our application, we first assign a characteristic value $x^v \in [0,1]$ to each vertex v. Then we iterate over the vertices, generate a uniform continuous random variable r supported on the bounded interval $[0,1]$ for each vertex $v \in V$, and *select* the vertex if (and only if) $r < \mathscr{P}(x^v)$.

Obviously, the geometric distribution of the selection depends on two factors: the shape of $\mathscr{P}$ and how the characteristic value x^v of v is computed for each vertex. Note that a constant function $\mathscr{P}$ provides a purely random sampling of V (see Fig. 2(b) and Fig. 3), because each vertex would be selected independently of the value of x^v. We would like to adapt the sampling to the local features of the mesh, however, without imposing additional complex data structures or processing. A single iteration over T allows assigning any quantity available in the one-ring to a vertex. Also, indexed meshes in geometric modeling package often maintain per-vertex normal information for visualization purposes. This "one-ring of vertex normals" is the maximum local support we can expect in a standard mesh format to analyze a given vertex. We make use of this normal information because it describes the visual features better than geometric errors [18, 3]:

$$x^v = \frac{\sum_{u \in N_v} \sigma(\mathbf{n}_v^\intercal \mathbf{n}_u)}{|N_v|}$$

where N_v is the set of neighboring vertices, $\mathbf{n}_v$ the normal vector of vertex v, and $\sigma : [-1,1] \rightarrow [0,1]$ a monotonically decreasing function for weighting the difference of two normal vectors. This estimator has a higher value in high curvature regions and can be modulated using the σ function. In most of our experiments we simply use:

$$\sigma(d) = (1-d)/2$$

which leads to good results. Fig. 2(a) shows color code visualizations of x^v. Note that alternative feature estimators may be used.

While x^v does capture local features, it cannot adequately reflect variations on a larger scale. Using only x^v for the selection of vertices would result in an undersampling of areas with only small curvature variation, with the effect of flattening large curved regions. Our main idea for coping with this problem without increasing the support of x^v (and in this way requiring

more costly preprocessing steps) is to exploit the influence on the selection process by modifying $\mathscr{P}$: while the selection of vertices with high values x^v has to be very likely, also selecting vertices with values x^v close to zero should be probable. In this spirit we define:

$$\mathscr{P}(x) = \tilde{k}\left(1+\alpha\left(\frac{x}{\overline{\{x^v\}}}-1\right)\right)$$

with $\alpha \in [0,1]$ the *adaptivity* coefficient (see Fig. 2(b) — we empirically observed good behavior on standard scanned meshes with $\alpha \in [2/3, 3/4]$), $\tilde{k}$ the target size of the subsampling, and $\overline{\{x^v\}}$ the arithmetic mean of x^v over V, acting as a global normalization.

Fig. 3 shows the influence of $\mathscr{P}$ on the subset of vertices being selected. The result of selecting vertices for which a random variable is smaller than $\mathscr{P}(x^v)$ yields the set V' with $k = |V'| \approx \tilde{k}$ elements. Note that enforcing $k = \tilde{k}$ would require the analysis of the distribution of x^v, which would significantly impact the performance. Similarly, alternative distribution functions may improve the adaptivity of the selection, taking into account multiscale features [14] for instance, but at the cost of much more expensive computations.

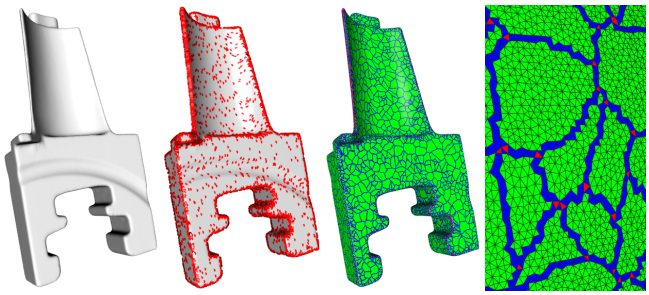
3.2. Topological vertex clustering

We partition M in k connected sub-meshes, by assigning each vertex of V to one of the vertices of V'. This is done by breadth-first traversing the vertices starting in all vertices of V' simultaneously. The procedure is described in detail in the algorithm 1.

Algorithm 1 breadth-first flood-filling from selection.

Require: $\overline{V}$ the set of selected vertices
Require: N_V an unordered vertex neighborhood structure
Require: Q a queue of vertex references
  for each vertex v in $\overline{V}$ **do**
    mark v with v
    append reference of v to Q
  end for
  while Q is not empty **do**
    $v :=$ head of Q
    for each vertex u in $N_V[v]$ **do**
      if u is not marked **then**
        mark u with mark of v
        append u to Q
      end if
    end for
    pop head of Q
  end while

Fig. 4 illustrates this flood-filling process. Breadth-first traversal of the edge graph means topological distances of the vertices rather than Euclidean or geodesic measures are considered. This avoids the need for a priority queue with (squared) distance values, which would make the process of inserting vertices at the traversal front logarithmic in time – while we process vertices in constant time during the traversal. We will later discuss a variant that trades part of this efficiency for better vertex distribution (see Section 4.1).



(a) Input (b) Selection (c) Regions (d) Close-up

Figure 4: **Topological vertex clustering**. Observe the close-up: the triangles are drawn in green when belonging to a single region, in blue for two regions and in red for three.

After traversal, each vertex belongs to exactly one connected subset of the mesh. These sets induce an equivalence relation $\sim$ on the vertices, i.e. two vertices are equivalent if they belong to the same set. Following the idea of *vertex clustering* [4], we re-index a subset of T to form the remeshed simplified connectivity T'. However, in contrast to spatial clustering, where vertices are equivalent if they are in the same grid cell, we use an equivalence relation generated by the traversal (see Fig. 5). A triangle $t \in T$ is classified according to the equivalence classes of its three vertices s_i, s_j, s_k:

i $s_i \sim s_j \sim s_k$: t is contained in a sub-mesh and degenerates to a point in the simplification; t is discarded

ii $s_i \sim s_j \nsim s_k$ (or similarly with the indices permuted): t intersects two sub-meshes and degenerates to a segment; t is discarded

iii $s_i \nsim s_j \nsim s_k \nsim s_i$: t covers the intersection of three sub-meshes and is re-indexed to the vertices $v_i, v_j, v_k \in V'$ with $v_i \sim s_i, v_j \sim s_j, v_k \sim s_k$; t is appended to T'.

We call this part of TopStoc *topological clustering* because the partitioning is performed in the topological space of the mesh (rather than the ambient space). This has immediate consequences for the topology of $\{V', T'\}$ relative to the original mesh: connected components are preserved, i.e. disconnected components (such as several close layers of different material in a CAD model) are not connected such as in spatial clustering

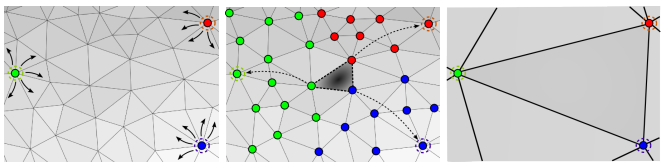


Figure 5: **Triangle re-indexing**. After selecting 3 seed vertices (red, blue and green), the breadth-first flood-filling process grows regions around them and triangles incident upon three regions are re-indexed to the seeds to form the simplified mesh.

approaches. In addition, the topology of the sub-meshes generated in the traversal implies the topology of the surface – this allows controlling the topology locally, which we discuss and exploit in the next section.

4. Establishing guarantees

TopStoc is based purely on local stochastic selection. This makes the algorithm very fast, but also means the placement of vertices as well as the topology of the resulting surface might sometimes turn out to be inadequate. We offer a modification of the sampling so that selected vertices have specified minimum distances; and we introduce resampling of sub-meshes that violate the *closed-ball property* [22] so that M' can be enforced to be homotopy equivalent to M or to specifically remove topological features that fit into a small ball of specified size. Note that both parts are extensions slowing down the basic algorithm (topology control breaks the linear time complexity), and that all results in the paper are generated with the basic algorithm unless explicitly noted.

4.1. Density control

Bounding the sampling density allows to better control the shape of resulting triangles, improving isotropic distribution and preventing highly varying local edge length. We propose a strategy inspired by *Poisson disk* rejection sampling methods [23, 24]. We perform an *early* breadth-first traversal bound to a given euclidean distance to the seed vertices already during the selection process: once a vertex has been selected, we traverse the neighborhood until a maximum distance ε has been reached, and discard all encountered vertices from the upcoming selection. In the subsequent topological clustering step the traversal is continued from the already traversed vertices, i.e. the vertices are still only visited once during the traversal.

We have found this procedure adds roughly 40% processing time on average (mostly due to computing euclidean distances) but preserves linear time and memory complexity. Note that geodesic distances would be required for a more accurate density control, but this is computationally prohibitive in our application scenario. Fig. 6 shows the improvement in the density control provided by *early* flood-filling (all other pictures in this paper are generated **without** density control). Note also that a too large value for ε may conflict with the target resolution of M' and lead to a coarser simplification than expected. Nevertheless, our density control mechanism succeeds to rule the distribution of V' efficiently.

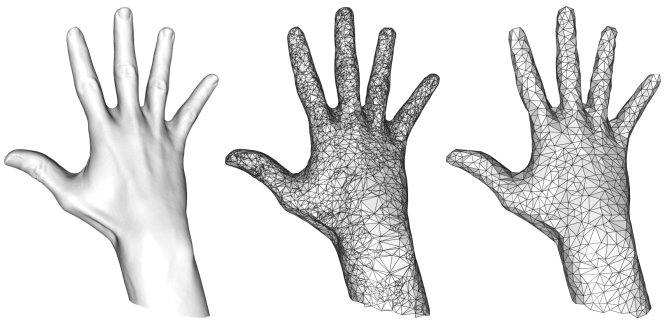


Figure 6: **Density control by early flood-filling**. Left: input model with 260k triangles. Middle: TopStoc simplification to 5% (166ms). Right: TopStoc with optional density control by early-flooding (228ms).

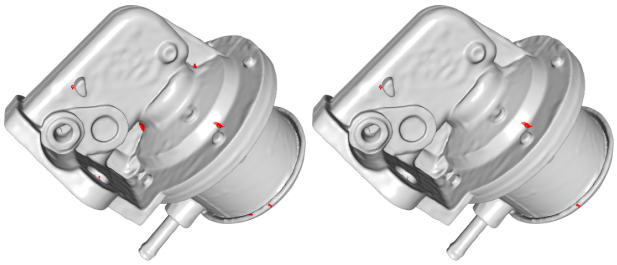


Figure 7: **Topological defects**. Multiply connected components are shown in red on a 0.5% (left) and a 2% (right) TopStocsimplification of the Oil Pump.

4.2. Topology control

If it is important to guarantee the homotopy between M and M', we can perform an optional *topological recovery* process, performed between clustering and triangle reindexing. For simplicity, we assume M is a globally orientable manifold mesh with or without boundaries; generalizations to non-orientable meshes follow the same ideas but require more elaborate tests.

Denote the k connected subsets in the dual of the edge graph as $C_i, i \in \{1,\ldots,k\}$. As dual vertices are faces, the $\{C_i\}$ cover the mesh surface (see Fig. 8). The *nerve* is the simplicial complex formed from the non-empty intersections of the subsets, i.e. edges for each non-empty intersection of two sets $\emptyset \neq C_i \cap C_j, i \neq j$, and triangles for each non-empty intersection of three sets $\emptyset \neq C_i \cap C_j \cap C_l, i \neq j \neq l$. By construction, T' consists of the triangles in the nerve. The *nerve lemma* applied to triangulations of closed surfaces [22] tells us that the triangulation T' is homotopy equivalent to M iff:

- **i** C_i is a closed 2-ball, i.e. simply connected, or homotopy equivalent to a manifold disk.
- **ii** the (non-empty) intersection between $C_i \cap C_j, i \neq j$ is a closed 1-ball, i.e. a single curve.
- **iii** the (non-empty) intersection between $C_i \cap C_j \cap C_l, i \neq j \neq l$ is a closed 0-ball, i.e. a single point.

This property has been termed the *closed-ball property*. Dyer et al. [25] recently showed that condition **(ii)** implies **(iii)** for connected components consisting of at least four subsets. We assume from now on that at least four sub-meshes have been generated and show how to check conditions **(i)** and **(ii)** for each of them. Note that the checks may appear to miss if two (or more) triangles are incident on the same three vertices – a non-manifold case that requires special attention if the mesh is simplified based on edge contractions [26]. This case is covered by condition **(iii)**, as three sub-meshes (which degenerate to vertices) would intersect in more than a single point (each intersection of three sub-meshes induces a triangle). As explained, we don't need to explicitly check condition **(iii)** as it is implied in our setting.

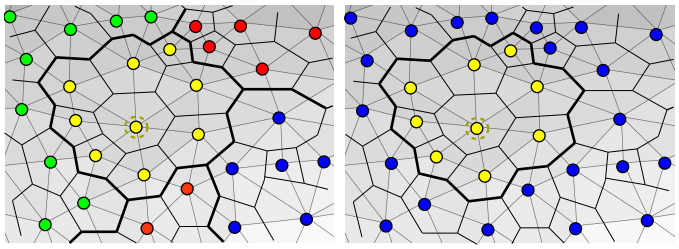


Figure 8: **Dual graph and violation of the second condition**. The characteristic of a sub-mesh can be computed from the elements incident on its primal vertices. Both illustrations show violations of the second condition: in the first example more than two triangles are induced between the red and yellow region; in the second example no triangles are induced.

First condition of the closed-ball property. We first note that it is sufficient to check the Euler characteristic of a sub-mesh: there are only two types of surface patches with Euler characteristic 1, a topological disk and a crosscap. Because we have assumed M to be globally orientable no connected subset of the mesh can be unorientable and have the topology of a crosscap. Thus, Euler characteristic of 1 is identifying topological disks in our setting.

Rather than actually considering C_i as comprised of dual primitives, we make use of the following observations (see also Fig. 8): the number of dual faces is equal to the number of (primal) vertices; the number of dual edges, respectively dual vertices is equal to the number of primal edges / primal faces connecting to only vertices within the sub-mesh (interior edges and vertices) plus those incident upon at least one vertex in the sub-mesh (boundary edges and vertices). Because the number of vertices and edges on the boundary (or boundaries) of the sub-mesh is equal, it suffices to count the number of vertices v_{C_i}, edges e_{C_i}, and faces f_{C_i} that are contained in the sub-mesh and check that

$$v_{C_i} - e_{C_i} + f_{C_i} = 1.$$

Second condition of the closed-ball property. Note that each open intersection curve between two components C_i and C_j induces two triangles: each dual edge separating the sub-meshes corresponds to a primal edge with vertices from different sub-meshes. At each end of the edge-path the triangle incident on the primal edge must be incident on a vertex from another sub-mesh, as otherwise the edge would also induce a dual edge separating C_i and C_j. These triangles are incident on vertices from three different sub-meshes and induce triangles in the coarse triangulation.

So, checking that the intersection of C_i with C_j is a single open curve means that C_i and C_j induce exactly two triangles. If the curve was open it would induce no triangles, and if there was more than one intersection curve it would induce more than two triangles (see Fig. 8). Thus, we count the number of occurrences of neighboring sub-meshes in the triangles induced around C_i and reject the component if any number is not equal to two.

Meshes with boundaries. If the surface has boundaries additional tests are necessary to ensure that their topology is preserved:

- the intersection between a sub-mesh and the boundary of M is empty or a single closed curve
- the intersection between two sub-meshes and the boundary of M is empty or a single point.

We assume boundary vertices in the original mesh are tagged. For testing we only consider C_i if it contains boundary vertices. The first condition means that the boundary vertices are connected by boundary edges and exactly two boundary vertices are endpoints of this vertex-edge path. We identify these endpoints as boundary vertices connected to another boundary vertex that is assigned to another sub-mesh C_j. The first test is satisfied if exactly two such vertices exist in C_i. The second test boils down to checking that these two vertices are connected to boundary vertices in $C_j \neq C_k$.

It turns out that in most cases very few sub-meshes have to be rejected, meaning most parts of the mesh are homotopy equivalent to the original one (see Fig. 10 and Fig. 7). Note that this is usually not the case in spatial clustering.

Topology correction. When a sub-mesh fails one or both of the two tests, we add it to a queue. Then, for each element of the queue, we perform a local (and slightly simplified) version of TopStoc: acting on the sub-mesh only, two vertices are *randomly* selected, generating two sub-meshes. Their topology is checked (as before) and they might be inserted to the queue for further subsampling. By construction, this process is guaranteed to converge, because components contain fewer and fewer vertices, and in the worst case, this will lead to the selection of all vertices in a sub-mesh, reproducing the input mesh locally. In the (usual) case of orientable surfaces, this process makes M' homeomorphic to M. Results are shown in Fig. 9(a).

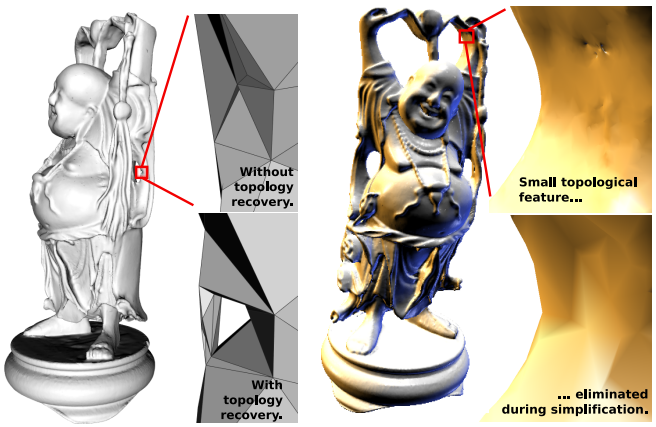


Figure 9: **Topology recovery**. (a) Happy Buddha with a close-up on a strong simplification with TopStoc, a hole disappears. With optional topology recovery, the genus of the original surface is preserved in its simplification. (b) Small topological events, such as this tunnel, can be ignored in the topological recovery process.

Enabling topology simplification. It is easy to limit topological recovery to geometrically large components. This can be useful if the input mesh is expected to have small, unwanted

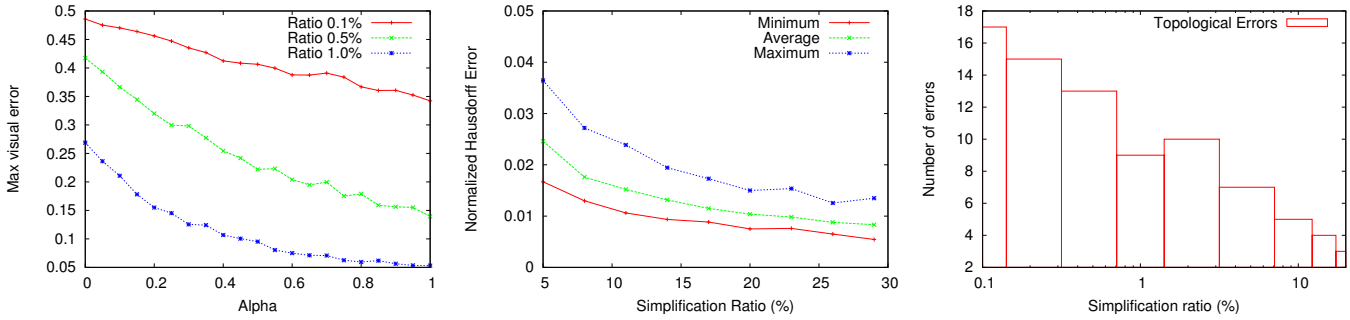


Figure 10: Statistics for each parameter value (Oil Pump model). Left: Influence of α on the curvature defect (over 1000 simplifications). Middle: Hausdorff error with original surface (over 1000 simplifications). Right: number of topological errors without topological recovery.

topological features, e.g. resulting from triangulating scanned data (see Fig. 9(b)). We only apply topological recovery to submeshes that either fail the second test or fail the first test while being larger than a given threshold.

Obviously, these extra steps decrease the overall performance. In our experiments, we observed an extra computation time ranging from 50 to 150% compared to the simplification without topological correction. As this part of our implementation is not optimized we see potential for improvement. For instance, the simplification showed in Fig. 9(a) requires 571ms for standard TopStoc simplification, and 898ms when enabling the *topological recovery*.

5. Implementation and results

TopStoc has been designed for aggressive in-core simplification, providing an alternative to grid clustering [4, 5], yet adapting to the topology and features of models in a CAD or geometry processing environment. Despite of its adaptivity, TopStoc is linear, both in time and memory. It only requires a fixed and small amount of memory for a given object, stored in a standard indexed mesh format, with unordered one-ring vertex references – in particular, no higher order mesh or hierarchical data structures are necessary. Consequently, TopStoc is very easy to implement (about 100 lines of code for the core algorithm).

For the very common task of down-sampling meshes of a few million primitives to several thousands, TopStoc needs less than a second on standard hardware. Table 1 summarizes timing and Hausdorff errors (measured with Metro [27] on an INTEL Q6600, 4GB, single thread) for different simplification ratios on various models. In Fig. 13, we compare TopStoc with two different algorithms (QSlim and Spatial Clustering), residing at each extremum of the *speed-vs-quality* spectrum. Despite the speed, visually important features are well preserved, as can be seen in Fig. 11 and Fig. 12. To substantiate this claim we have measured an error inspired by the definition of mesh saliency [14]. We compute a *visual error* as the mean curvature defect of samples between original and simplified triangulation. For each value of the adaptivity parameter α (ranging from 0 to 1), we perform a thousand of TopStoc simplifications and measure the maximum curvature defect observed. Fig. 10 shows averaged measures for 3 typical simplification ratio. We observe how α improves the situation compared to a uniform selection. For the sake of completeness, we also measured the evolution of Hausdorff error w.r.t. the sampling ratio.

Models	**Input** *triangles*	**Output** *triangles*	**Time**	**Hausdorff Error**
Rocker Arm	80354	2991	30ms	2.39%
		11913	45ms	0.65%
Chinese Dragon	305608	12083	134ms	2.96%
		45815	142ms	0.40%
Julius Caesar	774164	11721	268ms	2.04%
		115004	209ms	0.78%
Carter	1069048	27203	504ms	1.73%
		161869	410ms	0.24%
Grog	1752348	39239	545ms	2.23%
		263411	452ms	0.17%
David 2mm	7227031	131815	2.433s	1.54%
		1082707	1.799s	0.11%

Table 1: TopStoc performance measure. Hausdorff errors expressed relatively to the bounding box diagonal.

Fig. 14 illustrates how sharp geometric structures, important in CAD, are preserved in the output. Note that spatial clustering, either uniform or adaptive, produces hundreds of very visible topological errors on these examples, while TopStoc generates less than ten in its basic form, which can be fixed with the *topology recovery*. It is difficult to control the size of the output mesh in grid clustering, and often important features in CAD or scanned models are missed. In contrast, our approach adapts its sampling density to features, preserves topology, and is able to output large polygons in flat areas. Fig. 12 shows a set of levels of details from the Digital Michelangelo's David data [28]. The number of vertices in the output is within 1% of the target number for small input and output number of vertices, and naturally better for larger settings. For all applications we are aware of this is sufficient and significantly easier to control than uniform spatial clustering methods and more efficient than adaptive ones.

Limitations. A limitation stemming from how features are detected is resilience to noise: the purely local nature of feature estimation cannot distinguish noise from features and, consequently, meshes with geometric noise will be subsampled uni-

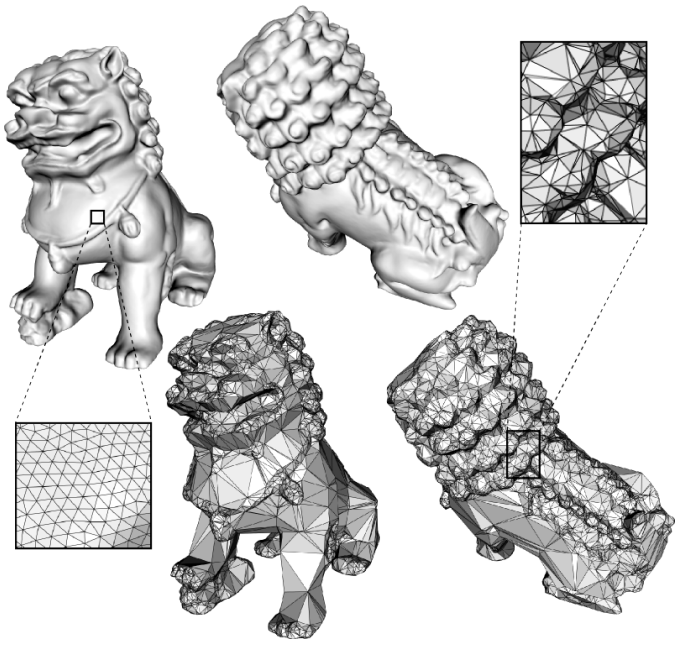


Figure 11: Chinese Dragon, simplified from 307k (top) to 7k triangles (bottom) in 0.3s with TOPSTOC.

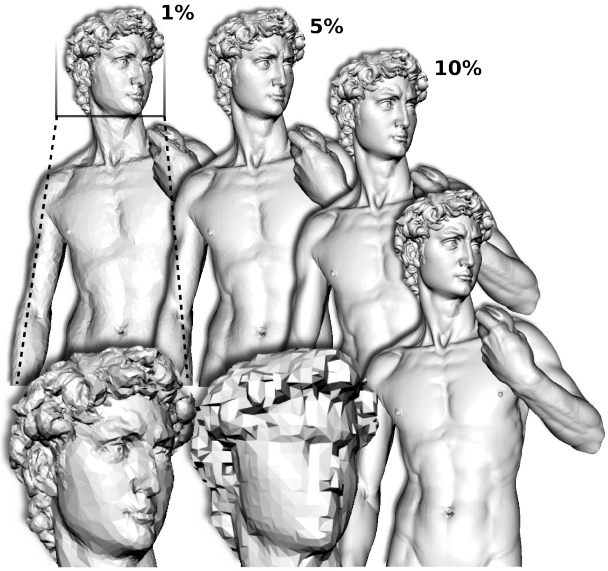


Figure 12: Multi-triangulation with TOPSTOC for generating a LOD over the David Model (7.2M tri.). The close-up compares the 1% TOPSTOC simplification with spatial clustering.

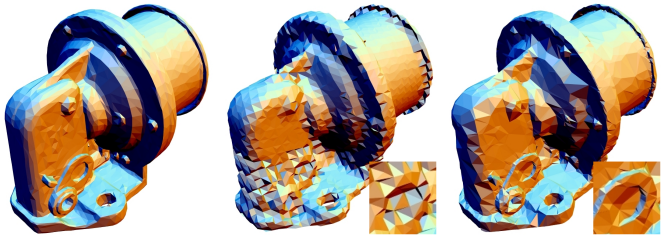
formly. This might not be ideal but still results in a reasonable approximation of the mesh. Note that recent advances in surface reconstruction methods include noise filtering in the mesh generation process. Similarly, if the mesh has already undergone a curvature-based adaptive resampling method, the influence of our feature estimation is less critical and uniform selection might be enough. Another limitation comes from the selection principle: TOPSTOC provides only an alternative, coarser triangulation of the original vertex set and it may be "biased" by a very non-uniform vertex distribution in the input. Note that the use of topological distances is usually not a problem for the typical dense meshes provided as input, in particular meshes generated using Marching Cubes. However, anisotropy could be improved by taking curvature into account during the (early) flood-filling process. Last, our topological recovery is unlikely to lead to an *optimal* homotopy equivalent triangulation, in the sense of satisfying both the sampling density required from a geometric point of view and the necessary number of triangles for homotopy equivalence.

6. Conclusion and Future Work

TOPSTOC is a simple but nevertheless very useful mesh simplification algorithm. By combining simple statistic and probabilistic tools, we design a geometry aware stochastic sampling that preserves important geometric features without requiring global sorting of the surface elements (spatial, error-driven, etc). With topological clustering, TOPSTOC preserves the topology of its input 3D surfaces, and in contrast to space partitioning approaches, avoids the collapsing of multiple surface layers. Moreover, it provides control over the preservation of topological features, with possibility of enforcing homotopy equivalence or simplification of geometrically small topological features. Although not optimal from a geometric point of view (i.e. the case of *extreme* simplification), TOPSTOC fits practical simplification scenarios, reducing multi-millions polygon models by 2 or 3 orders of magnitude in less than a second.

Both parts of the algorithm are interesting in their own right and could be used in different settings. Quite generally, stochastic sampling may be useful to produce globally reasonable vertex distributions without the need of non-linear computational complexity and it has already proven its effectiveness in other graphics areas [29, 30]. Topological clustering could be used with more elaborate distance metrics to generate better triangulations, with the benefit of having control over the topology of the resulting surface.

Ackowledgements: We thank Nina Amenta for early discussion on the Nerve lemma. We also thank reviewers for helping improving this paper. Models are courtesy Aim@Shape Network and Stanford University.



(a) QSlim:11.85s (b) Clustering:0.21s (c) TOPSTOC:0.36s

Figure 13: The Oil Pump model with originally 1.14M triangles simplified to 10K triangles.

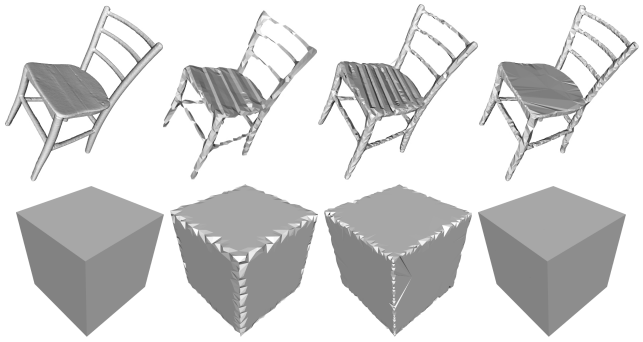


Figure 14: Simplification of a scanned chair (435k triangles, top) at 1% with uniform spatial clustering (135 ms, middle-left), adaptive spatial clustering (683 ms, middle-right) and TOPSTOC (186 ms, right). Note how sharp features and topology are preserved with TOPSTOC, while the other fast simplification methods produce strong topological and geometrical artifacts. We have added a synthetic example on the bottom line to illustrate the nature of visual artifacts of each method.

References

[1] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Mesh Optimization, in: ACM SIGGRAPH, 19–26, 1993.

[2] M. Garland, P. Heckbert, Simplifying Surfaces with Color and Texture using Quadric Error Metrics, in: IEEE Visualization, 263–269, 1998.

[3] D. Cohen-Steiner, P. Alliez, M. Desbrun, Variational shape approximation, in: ACM SIGGRAPH, 905–914, 2004.

[4] J. Rossignac, P. Borrel, Multi-Resolution 3D Approximation for Rendering Complex Scenes, Modeling in Computer Graphics (1993) 455–465.

[5] P. Lindstrom, Out-of-core simplification of large polygonal models, in: ACM SIGGRAPH, 259–262, 2000.

[6] H. Hoppe, Progressive Meshes, ACM SIGGRAPH .

[7] M. Garland, P. S. Heckbert, Surface simplification using quadric error metrics, in: ACM SIGGRAPH, 209–216, 1997.

[8] J. Wu, L. Kobbelt, Fast Mesh Decimation by Multiple-Choice Techniques, in: Vision, Modeling, Visualization, 241–248, 2002.

[9] C. DeCoro, N. Tatarchuk, Real-time Mesh Simplification using the GPU, in: Proc. of the ACM Symposium on Interactive 3D, 2007.

[10] M. Pauly, M. Gross, L. Kobbelt, Efficient simplification of point-sampled surfaces, in: IEEE Visualization, 163–170, 2002.

[11] P. Heckbert, M. Garland, Optimal Triangulation and Quadric-Based Surface Simplification, Journal of Computational Geometry: Theory and Applications 14 (1) (1999) 49–65.

[12] J. Wu, L. Kobbelt, A Stream Algorithm for the Decimation of Massive Meshes, in: Graphics Interface, 185–192, 2003.

[13] G. Turk, Re-Tiling Polygonal Surfaces, Computer Graphics (SIGGRAPH 92) 26 (2) (1992) 55–64.

[14] C. H. Lee, A. Varshney, D. Jacobs, Mesh Saliency, in: ACM SIGGRAPH, 659–666, 2005.

[15] E. Shaffer, M. Garland, Efficient adaptive simplification of messive meshes, in: IEEE Visualization, 127–134, 2001.

[16] S. Schaefer, J. Warren, Adaptive Vertex Clustering Using Octrees, in: Proceedings of SIAM Geometric Design and Computing, 491–500, 2003.

[17] T. Boubekeur, W. Heidrich, X. Granier, C. Schlick, Volume-Surface Trees, Computer Graphics Forum (Special issue on Eurographics) 25 (3) (2006) 399–406.

[18] D. Brodsky, B. Watson, Model Simplification through Refinement, in: Proceedings of Graphics Interface, 221–228, 2000.

[19] P. Cignoni, C. Rocchini, C. Montani, R. Scopigno, External Memory Management and Simplification of Huge Meshes, IEEE TVCG 9 (4) (2003) 525–537.

[20] M. Isenburg, P. Lindstrom, S. Gumhold, J. Snoeyink, Large Mesh Simplification using Processing Sequences, in: IEEE Visualization, 465–472, 2003.

[21] M. Isenburg, P. Lindstrom, Streaming Meshes, IEEE Visualization (2005) 231–238.

[22] H. Edelsbrunner, N. R. Shah, Triangulating topological spaces, in: Proc. of the 10th Symposium on Computational Geometry, 285–292, 1994.

[23] R. L. Cook, Stochastic sampling in computer graphics, ACM Transactions on Graphics 5 (1) (1986) 51–72.

[24] D. Dunbar, G. Humphreys, A Spatial Data Structure for Fast Poisson-Disk Sample Generation, in: ACM SIGGRAPH, 503–508, 2006.

[25] R. Dyer, H. Zhang, T. Möller, Surface sampling and the intrinsic Voronoi diagram, in: ACM Symposium on Geometry Processing, 1393–1402, 2008.

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

[27] P. Cignoni, C. Rocchini, R. Scopigno, Metro: measuring error on simplified surfaces, Computer Graphics Forum 17 (2) (1998) 167–174.

[28] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, D. Fulk, The Digital Michelangelo Project : 3D Scanning of Large Statues, in: ACM SIGGRAPH, 131–144, 2000.

[29] A. Kalaiah, A. Varshney, Statistical geometry representation for efficient transmission and rendering, ACM Trans. Graph. 24 (2) (2005) 348–373, ISSN 0730-0301, doi:http://doi.acm.org/10.1145/1061347.1061356.

[30] R. L. Cook, J. Halstead, M. Planck, D. Ryu, Stochastic simplification of aggregate detail, ACM Trans. Graph. 26 (3) (2007) 79.