SURFACE COMPUTATIONS WITH DECOUPLED INTRINSIC GEOMETRY

Randy Shoemaker
Advisor: Dr. Pieter Peers
College of William & Mary

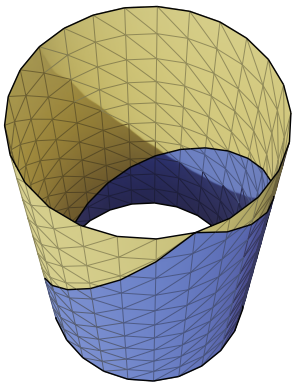
April 2023

Abstract

Many dynamic processes, such as heat flow, take place on the surface of a shape. Computations involved in simulating such processes are unnecessarily computationally intensive since they are computed using 3D data whereas a shape's surface only has two degrees of freedom. The 3D representation of a shape is its *extrinsic geometry*, and the 2D representation is called its *intrinsic geometry*. For computing purposes, the extrinsic geometry of a surface is typically represented by a mesh, a collection of flat triangles connecting points in three dimensions. Whereas intrinsic geometry can be represented by a mesh of triangles that *wrap around* the surface, specified only with edge lengths. We present a fast algorithm for computing a compact intrinsic triangulation of a given extrinsic shape with far fewer triangular elements than the extrinsic triangulation. We demonstrate the use of the simplified mesh for solving Partial Differential Equations. To the knowledge of the authors, we present the first intrinsic mesh simplification algorithm. We demonstrate the robustness and effectiveness of our algorithm on the benchmark *Thingi10k*[19] mesh data set. We show the effect of intrinsic simplification by solving Poisson equations on our intrinsically simplified meshes and give a comparison to existing extrinsic simplification methods.

Introduction

Geometric data encountered in the wild are often *"messy"* from a geometry processing perspective, necessitating the need for robustified processing methods[20,7,15,13,11]. Intrinsic triangulation frameworks[4,17,6] have been proposed as an alternative strategy for robust geometry processing in the wild. Intrinsic triangulations approach geometry processing from an intrinsic view where all operations are expressed as combinations of atomic operations (e.g., edge flipping, vertex insertion, etc.) defined on distances between points over the 2D manifold. While, existing intrinsic triangulation frameworks differ in data-structure and efficiency of certain atomic operations, they all share that each "extrinsic" vertex has an immutable counterpart in the intrinsic triangulation, and consequently, vertex removal of initial extrinsic vertices is not supported. Many in-the-wild triangle meshes are finely triangulated in order to faithfully approximate curved surfaces in $\mathbb{R}^3$ by piecewise planar surfaces. However, a significant portion of in-the-wild triangle meshes are designed with CAD tools or are the result of 3D scans of real-world surfaces formed by combining developable patches. Intrinsically, such developable parts are isometric to a plane. For example consider a cap-less cylinder which can be fully modeled using just two intrinsic triangles (see inset).



However, to faithfully capture the curvature, thousands of extrinsic triangles are needed in $\mathbb{R}^3$. Not only does blindly embedding a heavily tessellated developable surface incur significant overhead, it also impacts the efficiency of many downstream processing algorithms such as optimal Delaunay triangulations[2] and adaptive intrinsic mesh refinement[17].

We present a topology-preserving method for simplifying a triangle mesh directly on the intrinsic manifold. A key insight is that vertices with zero Gaussian curvature can be removed without impacting the accuracy of the metric defined on the embedding. Moreover, a locally developable approximation can be obtained by allowing vertices with small Gaussian curvature to be removed. However, classic vertex merging and edge collapse require updating of edge lengths in the intrinsic setting which can be non-trivial when removing a vertex with non-zero curvature. Instead, we introduce a novel intrinsic simplification method based on edge flipping, a stable atomic intrinsic operation. For each vertex we would like to remove, we perform edge flips until the vertex has valance 3 (or valance 2 for vertices on the boundary). We can then remove the vertex if the resulting triangulation remains valid. If the triangulation becomes invalid, we undo the edge flips in reverse order and reschedule the vertex for later evaluation. We process vertices in a greedy *"intrinsically-flattest-first"* order until no more vertices can be removed. In addition, we keep track of the intrinsic location of deleted vertices by their intrinsic barycentric coordinates. Compared to extrinsic simplification which approximates the surface with fewer planar triangles, intrinsic simplification can be seen as approximating the surface with developable patches.

We demonstrate and validate the robustness of our method on the *Thingi10k* dataset[19], and evaluate the impact of relaxing the Gaussian curvature threshold at which vertices can be removed.

Related Work

Mesh Simplification

There exists a vast body of work on *extrinsic* mesh simplification, however we will focus on seminal papers in this area and contrast them against our intrinsic mesh simplification method; we refer the reader to[8] for a comprehensive review. Extrinsic mesh simplification methods aim to reduce the number of vertices in the mesh such that some quality metric is best preserved. The most commonly preserved quality is the visual appearance of a mesh[14,5,12,10,3]. Vertex decimation[14] iteratively deletes vertices according to an extrinsic criterion and the resulting hole is carefully re-triangulated. Our method of vertex deletion is similar to vertex decimation, except that we employ intrinsic edge flips until the ring of a vertex is a triangle which can be removed without retriangulation. In their seminal work, Garland and Heckbert[5] greedily merge vertices to minimize a quadric error metric (QEM) via edge contraction. We also follow a greedy approach, but instead of QEM, we use Gaussian curvature to drive the simplification.

Recently, Lescoat et al.[9] proposed Spectral Mesh Simplification (SMS), a greedy extrinsic mesh simplification strategy, that aims to preserve the intrinsic geometry (i.e., minimize the change in the first k eigenvectors of the Laplace-Beltrami operator). Spectral mesh simplification is able to reduce the number of extrinsic triangles while minimizing errors when computing spectral distances. However, spectral mesh simplification is relatively computationally expensive and limited to reducing extrinsic surfaces. In contrast, intrinsic mesh simplification is computationally light weight and it is more efficient in reducing the number of elements in developable patches while preserving the intrinsic geometry.

Intrinsic Triangulations

Intrinsic triangulation frameworks[4,17,6] provide tools and atomic operations to perform geometry processing algorithms that only rely on intrinsic information directly on the intrinsic

mesh (e.g., geodesic distance[16], computing distortion minimizing homographies[18], or algorithms that rely on the Laplace-Beltrami operator[1]). All existing intrinsic triangulation frameworks support edge flipping, and the Signpost[17] and Integer Coordinate framework[6] support additional atomic operations such as adding vertices, repositioning (added) vertices, and computing common subdivisions. However, none of the existing intrinsic triangulation frameworks currently support the removal of an extrinsic's vertex's intrinsic counterpart. While we implement our method in the Signpost[17] framework to facilitate visualization of the resulting mesh, our method only relies on edge flipping to remove vertices, opening the door to possible adaptation to other current and future intrinsic triangulation frameworks.

Background

We briefly discuss the information required to support intrinsic triangulations relevant for introducing our intrinsic simplification method.

The intrinsic geometry of a shape is represented by a mesh $\mathsf{M} = \{\mathsf{V}, \mathsf{E}, \mathsf{F}\}$, consisting of a collection of vertices V, edges E, and triangles F, and additionally the lengths ℓ_{ij} of the ij-th edge in E between vertices i and $j \in \mathsf{V}$. The lengths ℓ_{ij} describe the shape of the triangles. For visualization purposes we also store signpost angles[17] φ_{ij} which describe the direction from vertex i to vertex $j \in \mathsf{V}$ along edge ij defined in a local polar coordinate system at i. It can be easily seen that these two pieces of additional information fully define the intrinsic geometry. Other relevant intrinsic information can be directly computed from the edge lengths:

$$\theta_i^{jk} = \arccos\left(\frac{\ell_{ij}^2 + \ell_{ik}^2 - \ell_{jk}^2}{2\ell_{ij}\ell_{ik}}\right), \quad (1)$$

$$A_{ijk} = \sqrt{s(s-\ell_{ij})(s-\ell_{jk})(s-\ell_{ki})}, \quad (2)$$

$$s = (\ell_{ij} + \ell_{jk} + \ell_{ki})/2 \quad (3)$$

where θ_i^{jk} is the interior angle at $i \in \mathsf{V}$ in the triangle $ijk \in \mathsf{F}$ and A_{ijk} is the area of the triangle $ijk \in \mathsf{F}$.

Intrinsic Simplification By Edge Flipping

Our goal is to remove intrinsic vertices that are part of a (near) developable patch in the corresponding extrinsic mesh. Removing such vertices will not alter the intrinsic geometry since a developable patch is isometric to a planar neighborhood. A surface is developable around a vertex i if it has zero Gaussian curvature: $\kappa_i = 2\pi - \alpha_i$, where $\alpha_i = \sum_{ijk} \theta_i^{jk}$ is the cone angle. Ideally, only vertices with zero Gaussian curvature should be removed such that the intrinsic geometry is not changed. However, developable surfaces are often highly tessellated for accurate approximation in $\mathbb{R}^3$. Depending on the exact triangulation and/or numerical round-off errors, an exact zero Gaussian curvature might not be reached. Therefore, in practice we try to remove all vertices with a Gaussian curvature less than some predetermined threshold κ_{max}. Setting κ_{max} to a larger threshold, allows us to obtain an intrinsic approximation where portions of the triangulation are replaced with developable patches.

Although not strictly necessary, we start by performing an intrinsic Delaunay retriangulation to ensure a well behaved mesh. Next, we sort all vertices by their Gaussian curvature in a queue P by smallest Gaussian curvature first, and process the vertices in P until no vertices with a Gaussian curvature less than κ_{max} can be removed anymore. For each vertex, we perform edge flips on all incident edges until the desired valence (three for interior vertices or two for vertices on boundaries) is reached. We record each edge flip in a FIFO queue Q for additional post-processing detailed below. For vertices with zero Gaussian curvature, we are guaranteed to reach this goal[16]. For vertices with non-zero curvature the desired valence is not guaranteed. In practice we found that we usually can achieve the desired valance for small non-zero Gaussian curvature. When the desired valence is reached, simplification can be easily achieved by removing the vertex i and all incident edges ij, ik and il, and that the resulting triangle jkl forms a developable approximation

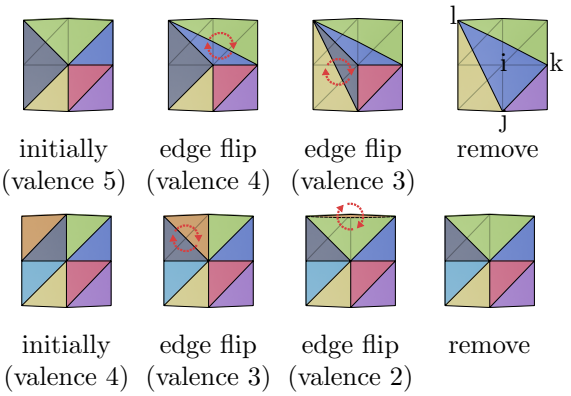


Figure 1: Illustration of intrinsic vertex removal by edge flipping for an interior (top) and a boundary (bottom) vertex.

(Figure 1). However, depending on the configuration of the 1-ring, removing the vertex can lead to an invalid triangulation. We therefore only remove vertices if the resulting triangle jkl strictly adheres to the triangle inequality.

When we are unable to reach the desired valence or if the triangle inequality is violated, we undo the edge flips recorded in Q in reverse order to restore the triangulation. It is possible that after removing more vertices, the triangulation is more favorable for removing the vertex. Therefore, we re-queue the vertex for reprocessing after all outstanding vertices with a Gaussian curvature less than κ_{max} have been processed.

If a vertex i can be successfully removed (i.e., it has the desired valence after edge flipping, and the containing triangle is valid), then we perform the following steps:

- We update the Gaussian curvature of the vertices of jkl in the processing queue P (either by updating the order if j, k, or l was in the queue, or by adding the vertex if not yet queued).

- During edge flipping to achieve the desired valence, it is sometimes possible to create degenerate triangles. Instead of trying to figure out a safe flipping order, we instead 'repair' the triangulation after vertex removal. For each edge recorded in Q, we check (in reverse order) if the resulting edges are Delaunay. If not, then we flip the edge, and add the four edges of the resulting triangles to Q. We repeat this process until Q is empty and the resulting triangulation meets the intrinsic Delaunay property again.

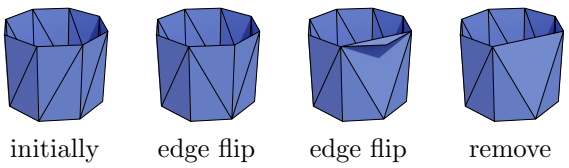


Figure 2: Applying extrinsic edge flips for vertex removal can result in degraded geometry. In contrast to an intrinsic edge flip producing an edge that 'rides' the surface, an an extrinsic edge flip changes the intrinsic geometry by altering distances along the surface.

Discussion

The above edge flipping strategy reduces the 1-ring polygon around each candidate vertex for removal to the trivial re-triangulation case; i.e., such that no re-triangulation is needed. We can apply this strategy only on the intrinsic triangulation, not in an extrinsic setting where it can significantly alter the shape (Figure 2).

Furthermore, only the actual intrinsic vertex removal step alters the Gaussian curvature at the vertex (if not zero); Gaussian curvature remains unchanged under intrinsic edge flipping. Vice versa, while retriangulation is necessary in the extrinsic setting, it is unclear in the intrinsic domain how to compute the edge lengths when the Gaussian curvature is not zero. First, the 1-ring polygon needs to be projected to a developable surface, which can possibly entail changes to the intrinsic mesh beyond the polygon. Second, it is unclear what the updated edge lengths should be (i.e., the candidate vertex needs to be projected onto the developable approximation).

An additional benefit of our simplification algorithm is that it preserves the Euler characteristic (V-E+F) of the mesh. For a closed mesh (without boundary), each vertex removal step results in a deletion of 1 vertex, 3 edges,

and 3 faces, while adding 1 new face (ΔEuler: $-1+3-3+1=0$). In case there is a boundary, we remove 1 vertex, 2 edges and 1 face (ΔEuler: $-1+2-1=0$). By virtue of the Gauss-Bonnet theorem, we know that since the Euler characteristic is preserved, the sum of the Gaussian curvature is unchanged. Thus deleting a vertex i implies that its Gaussian curvature is redistributed to its neighbors. This also justifies why after each vertex removal, we update the Gaussian curvature sorted processing queue `P`.

Intrinsic-Extrinsic Correspondence

Similar to the Signpost data structure, we keep track of the correspondences between intrinsic and extrinsic vertices. Special care needs to be taken for tracking the correspondence of deleted vertices. This is achieved by maintaining *intrinsic* barycentric coordinates for each extrinsic vertex deleted from the intrinsic triangulation. There are three scenarios to consider:

1. *Defining the barycentric coordinate* (c_i^j, c_i^k, c_i^l) *of a deleted vertex* i inside a triangle jkl. When the threshold κ_{max} is not zero, the vertex is 'projected' onto a developable approximation. Since this projection is an approximation, a number of viable options exist. In our implementation, we opt for maintaining the relative ratio of triangle areas: $c_i^j = A_{ikl}/(A_{ikl}+A_{ilj}+Aijk)$, and similarly for c_i^k and c_i^l. When the threshold is set to zero, these correspond to the regular barycentric coordinates.

2. *Updating the barycentric coordinates of a vertex* v *that depends on a deleted vertex* i. When a vertex is deleted, it is possible that a previously deleted vertex's barycentric coordinates depends on it. Because our vertex deletion is only performed after the vertex has the desired valance, it follows that the dependent vertex must lie in one of the faces that will be merged. Hence, both the barycentric coordinates of the deleted and dependent vertex will depend on the same corner vertices after deletion. Thus, we can easily substitute the barycentric coordinates of the deleted vertex. For example, if $v \in ijk$, and i is deleted, then the updated coordinates are: $(c_v^j + c_v^i c_i^j, c_v^k + c_v^i c_i^k, c_v^i c_i^l)$.

3. *Updating the barycentric coordinate if the vertex is contained in a triangle whose edge is flipped.* In this case, we recompute the barycentric coordinates as ratios of the 2D areas after unfolding both the triangles in a local plane.

Results and Evaluation

We have implemented our intrinsic simplification method in `C++` using the Signpost data structure and a half-edge extrinsic mesh representation. We have validated our method on a subset of the *Thingi10k*[19] dataset containing all valid manifold and oriented triangle meshes. Processing all 7,129 triangle meshes takes approximately 4.25 hours to run the whole algorithm for 4 thresholds values: $\kappa_{max} = \{10^{-9}, 10^{-6}, 10^{-4}, 10^{-2}\}$ (in radians) on a Intel i5-8265U (1.60GHz) CPU with 16GB of memory (using a single core).

Figure 3 shows a selection of 3 meshes from the test set. For each mesh we show the original mesh, and 3 thresholds $\kappa_{max} = \{10^{-6}, 10^{-4}, 10^{-2}\}$. The visualizations of the simplified meshes are projected on the original extrinsic mesh, and thus any deformations due to non-zero Gaussian vertex deletion are not visualized. Note how our method simplifies most in regions that are developable or approximately developable. For example, in the *Fan Blades* example, we can see that significant simplification takes place on the backside of the fan blades and on the base in between the blades. The feet of the *Robot* consists of nearly developable surfaces, and thus show a significant reduction in intrinsic triangles. Note how some of the simplified intrinsic triangles form a curved surface in $\mathbb{R}^3$. Finally, the *Rods* example shows most simplification on the sides of the rods. As expected, for each example, a higher threshold removes move vertices.

Table 1: Mesh simplification statistics over a subset of 7,129 manifold and oriented triangle meshes in the Thingi10K dataset comparing the percentage of vertices with a Gaussian curvature less then κ_{max} versus the percentage removed after simplification.

	Removable		Successfully Removed		Mean Time (s)		
κ_{max}	mean	std. dev.	mean	std. dev.	remove	track	total
10^{-9}	5.55%	13.81	99.54%	4.97	0.16s	0.13s	0.28s
10^{-6}	7.46%	16.16	99.32%	5.39	0.18s	0.18s	0.36s
10^{-4}	12.67%	20.24	96.92%	7.36	0.26s	0.30s	0.56s
10^{-2}	34.08%	33.55	93.66%	9.12	0.53s	0.42s	0.96s

To gain further insight in the efficacy of our intrinsic simplification method, Figure 1 reports the mean percentage and standard deviation of ideally removable vertices (i.e., those with a Gaussian curvature less than κ_{max}) and the mean percentage (and standard deviation) of actually removed vertices for four different thresholds $\kappa_{max} = \{10^{-9}, 10^{-6}, 10^{-4}, 10^{-2}\}$. As expected with an increasing threshold, more vertices can be removed, and in general vertex removal succeeds for almost all candidate vertices. In addition, we also report the mean running time (excluding data IO); vertex removal and updating the intrinsic barycentric coordinates take about the same amount of compute time.

To demonstrate the impact of removing nonzero Gaussian curvature vertices, we visualize the solution to a Poisson equation with a single source term centered on a chosen vertex on 2 selected meshes for 3 κ_{max} thresholds (Figure 4); we report the number of vertices and MSE error over the Poisson solution for each of the vertices in the original mesh (possible interpolated based on the intrinsic barycentric coordinates). We deliberately did *not* apply any refinement and directly solve the equation on the simplified mesh to better illustrate the impact of the simplification. In the first example, we deliberately placed the source term on a vertex with low Gaussian curvature. As expected, as the threshold κ_{max} increases, so does the error. In the second example, we placed the source term on a vertex with high Gaussian curvature, which results in a smaller increase in error because most intrinsic triangles around the source term will not be simplified. In a practice, for maximal accuracy, we would first apply optimal intrinsic Delaunay triangulation[17] or adaptive intrinsic mesh refinement[17]. While both method will again increase the number of intrinsic vertices, both methods would not be encumbered by suboptimally placed intrinsic vertices from the original mesh.

A key advantage of simplifying the intrinsic representation compared to first simplifying an extrinsic mesh and then constructing an intrinsic representation is that we can produce simplified triangles with "bend" edges, and thus better maintain the metric over the surface. To better understand the differences between different simplification methods, Figure 5 compares intrinsic mesh simplification with two extrinsic simplifications methods: QEM[5] and Spectral Mesh Simplification[9]. For each method, we perform an equal-vertex-count comparison (simplified from 14,290 to 1,715 vertices for the *Bunny*, and from 23,356 to 4,440 vertices for the *Frog*) to our intrinsic simplified mesh, and highlight the differences between all methods by visualizing the solution to a Poisson equation computed directly on the "raw" mesh with a source term placed at a low and high Gaussian curvature vertex. Note that Spectral Mesh Simplification optimizes for the intrinsic qualities of the mesh when removing vertices, and as such on average the solution to the Poisson equation is more accurate, albeit at a much higher computation cost (35 minutes versus 0.62 seconds for our method on the *Bunny*; for reference QEM

took 7.6 second seconds versus 0.85 seconds for our method on the *Frog*). However, as noted before, intrinsic mesh simplification is intended as a preprocessing step, and in practice we would apply an adaptive refinement or optimal Delaunay triangulation before solving the Poisson equation.

Future Work

Currently our simplification method utilizes signposts to encode directions along the surface of a mesh, which enables us to visualize the simplified mesh 'projected' on the extrinsic mesh. However when the threshold κ_{max} becomes large the signposts are no longer valid. A method for robustly updating the signposts in the presence of large κ_{max} would open the door to more downstream algorithms, such as vector field processing[17].

Understanding the effect of our simplification algorithm or a larger class of downstream applications and algorithms is crucial for building upon our framework and is a current area of research. Furthermore, giving users finer control over the speed/accuracy trade off inherent to all simplification methods (though less extreme in the intrinsic setting) is also a goal of future research.

Finally we think that adapting other extrinsic simplification methods, such as edge collapse[5], to the intrinsic setting could provide interesting avenues of future research.

Conclusion

We presented, to the best of our knowledge, the first intrinsic mesh simplification method. Our method leverages the benefits of intrinsic edge flipping to significantly simplify vertex deletion to two canonical cases (i.e., valence three for an internal vertex, and valence two for a boundary vertex). We use Gaussian curvature as the deletion criterion, which effectively projects vertices onto a locally developable approximation. We demonstrated the robustness and effectiveness of our method on the Thingi10k dataset.

Acknowledgements

I would like to thank my Ph.D advisor Pieter Peers for all of his guidance. Many thanks go to my lab mate Sam Sartor for generating the beautiful images used in this manuscript. Finally, I would like to thank the Virginia Space Grant Consortium for supporting this research.

References

[1] M. Botsch et al. *Polygon Mesh Processing*. 2010. ISBN: 9781439865316.

[2] Long Chen and Jinchao Xu. "Optimal Delaunay triangulations". In: *Journal of Computational Mathematics* 22.2 (Mar. 2004), pp. 299–308.

[3] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. "Variational Shape Approximation". In: *ACM Trans. Graph.* 23.3 (2004), 905–914.

[4] Matthew Fisher et al. "An algorithm for the construction of intrinsic Delaunay triangulations with applications to digital geometry processing". In: *Computing* 81.2 (2007), pp. 199–213.

[5] Michael Garland and Paul S. Heckbert. "Surface Simplification Using Quadric Error Metrics". In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '97. 1997, 209–216.

[6] Mark Gillespie, Nicholas Sharp, and Keenan Crane. "Integer coordinates for intrinsic geometry processing". In: *ACM Trans. Graph.* 40.6 (2021).

[7] Yixin Hu et al. "Tetrahedral Meshing in the Wild". In: *ACM Trans. Graph.* 37.4 (2018).

[8] Dawar Khan et al. "Surface remeshing: A systematic literature review of methods and research directions". In: *IEEE TVCG* 28.3 (2020), pp. 1680–1713.

[9] Thibault Lescoat et al. “Spectral mesh simplification”. In: *Comp. Graph. Forum*. Vol. 39. 2. 2020, pp. 315–324.

[10] Jovan Popović and Hugues Hoppe. “Progressive Simplicial Complexes”. In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’97. 1997, 217–224.

[11] Yang Qi et al. “A bidirectional formulation for Walk on Spheres”. In: *Comp. Graph. Forum* 41.4 (2022), pp. 51–62.

[12] Jarek Rossignac and Paul Borrel. “Multi-resolution 3D approximations for rendering complex scenes”. In: *Modeling in computer graphics*. Springer, 1993, pp. 455–465.

[13] Rohan Sawhney and Keenan Crane. “Monte Carlo Geometry Processing: A Grid-Free Approach to PDE-Based Methods on Volumetric Domains”. In: *ACM Trans. Graph.* 39.4 (2020).

[14] William J Schroeder, Jonathan A Zarge, and William E Lorensen. “Decimation of triangle meshes”. In: *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. 1992, pp. 65–70.

[15] Silvia Sellén et al. “Solid Geometry Processing on Deconstructed Domains”. In: *Comp. Graph. Forum* 38.1 (2019), pp. 564–579.

[16] Nicholas Sharp and Keenan Crane. “You Can Find Geodesic Paths in Triangle Meshes by Just Flipping Edges”. In: *ACM Trans. Graph.* 39.6 (2020).

[17] Nicholas Sharp, Yousuf Soliman, and Keenan Crane. “Navigating intrinsic triangulations”. In: *ACM Trans. Graph.* 38.4 (2019).

[18] Kenshi Takayama. “Compatible Intrinsic Triangulations”. In: *ACM Trans. Graph.* 41.4 (2022).

[19] Qingnan Zhou and Alec Jacobson. “Thingi10K: A Dataset of 10,000 3D-Printing Models”. In: *arXiv preprint arXiv:1605.04797* (2016).

[20] Qingnan Zhou et al. “Mesh Arrangements for Solid Geometry”. In: *ACM Trans. Graph.* 35.4 (2016).

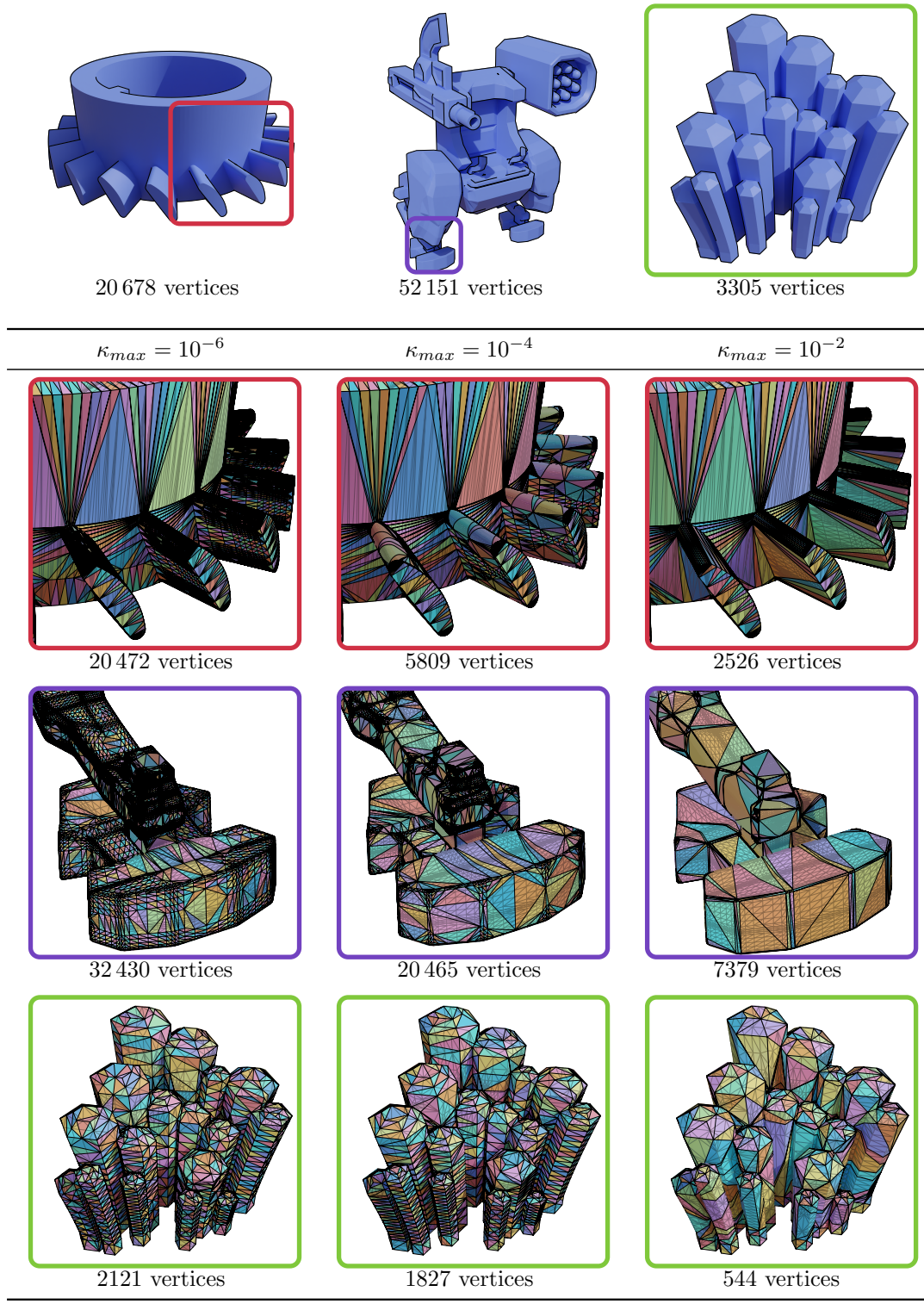


Figure 3: Intrinsic mesh simplification results on three meshes for different thresholds κ_{max}. Note that the simplified intrinsic mesh is projected on the original mesh, thus any deformations due to removal of intrinsic vertices with non-zero Gaussian curvature are not visualized.

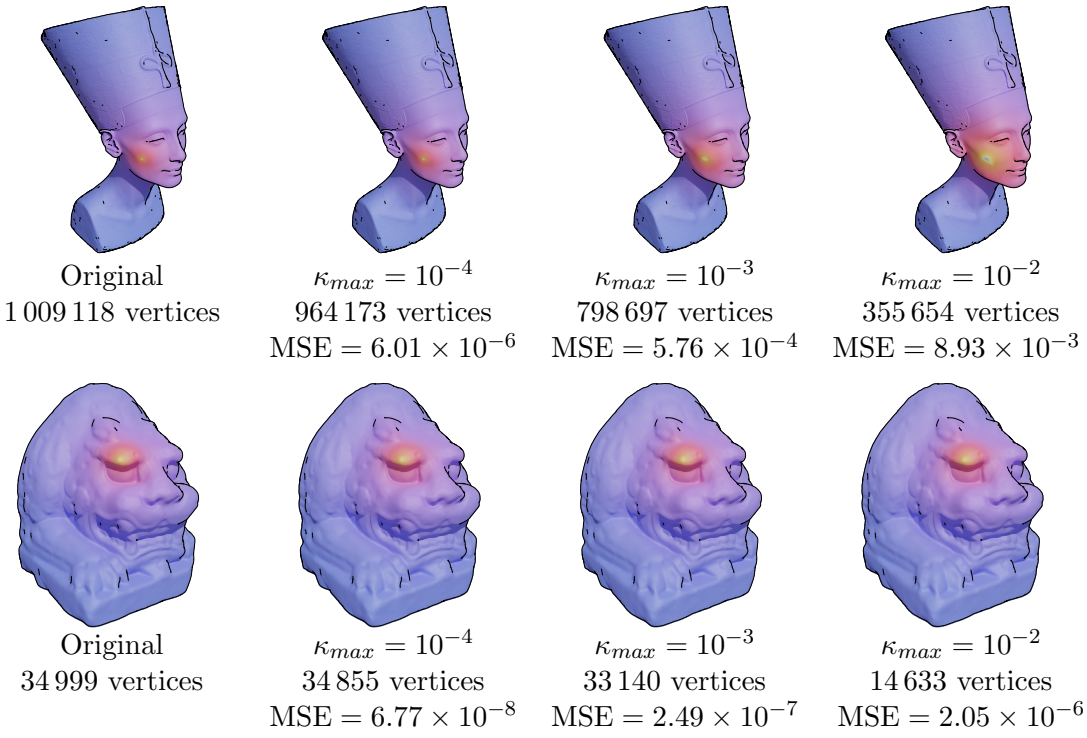


Figure 4: Visualizations of the solution of a Poisson equation with the source placed on a near-developable vertex (top) or on a vertex with high curvature (bottom). The equation is solved directly on the "raw" simplified mesh to better show the impact of simplification.

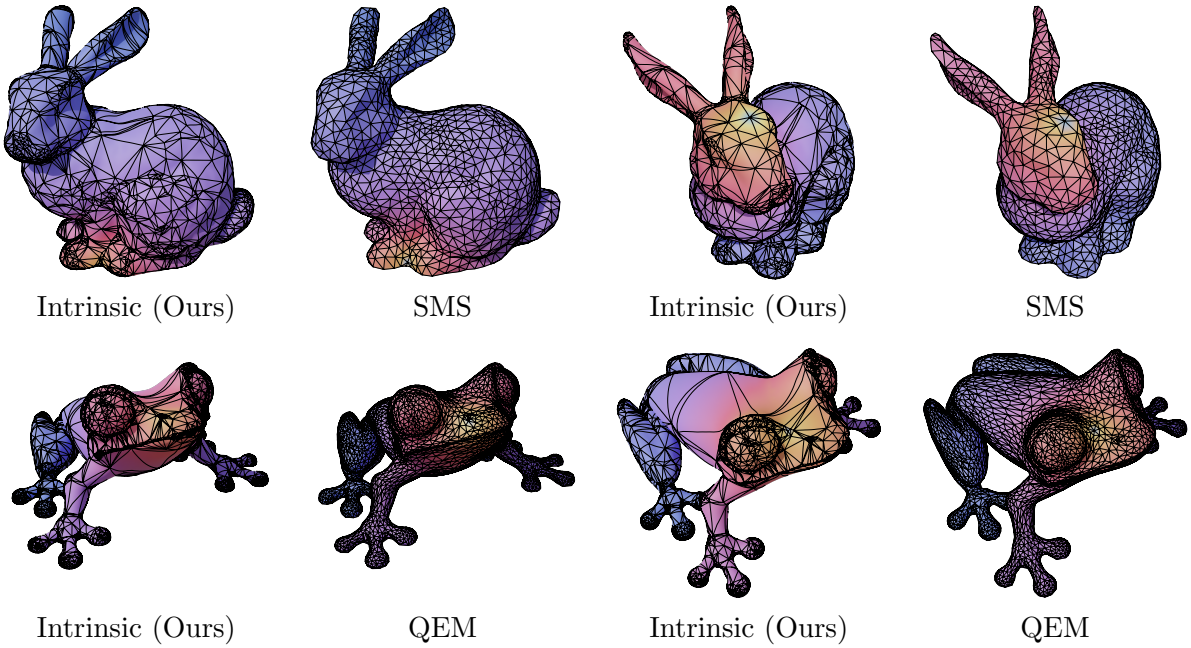


Figure 5: Solutions of a Poisson equation computed on simplified meshes obtained with our intrinsic simplification algorithm, Spectral Mesh Simplification (SMS), and Quatric Error Metrics (QEM). All solution are computed on the "raw" simplified mesh to better show the impact of simplification.