INTRINSIC MESH SIMPLIFICATION

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

April 2024

Abstract

Triangle meshes are useful for many geometry processing tasks. A triangle mesh consists of a set of points connected to form triangles. Many triangles are required to faithfully represent the shape of a surface. The amount of computation is adversely affected by the number of elements in a mesh, thus there is a trade off between mesh quality and computational efficiency. Mesh simplification algorithms have been developed to reduce the number of elements in a mesh while aiming to preserve features of its shape. Recently intrinsic mesh simplification was presented separately by Shoemaker et al.[15] and Liu et al.[11]. Intrinsic mesh simplification utilizes intrinsic triangulations to help navigate the trade off, yielding high quality geometric approximation. However, current current algorithms suffer from bent or kinked edges. We examine the issue of edge kinks and present an algorithm that avoids bent or kinked edges.

Introduction

Intrinsic triangulations have proved useful in many Geometry Processing tasks ranging from conformal mesh parameterization[1,6], to geodesic computation[13], and computing homeomorphisms between shapes[18]. Recently intrinsic triangulations have opened the door to a new class of mesh simplification algorithms[15,11] better able to preserve intrinsic geometry by operating in a larger space of triangulations[14]. Intrinsic mesh simplification yields a coarse intrinsic mesh with far fewer mesh elements that is suitable for many downstream geometry processing tasks. Computing on the surface of the coarse mesh is much faster since fewer elements are used to represent the geometry. Furthermore the intrinsic representation is decoupled from the original embedding of the surface, ignoring details that are unnecessary for intrinsic geometry processing. The coarse mesh produced by intrinsic simplification algorithms is useful for fast approximation of PDEs, geodesics, and Voronoi cells.

Current intrinsic simplification algorithms produce a mapping between the coarse and fine mesh. This map is required to translate computational problems defined on the fine mesh to one defined on the coarse and vice versa. Evaluating the map at mesh vertices is fast, however, evaluating the map at any other point is costly, since all coarsening operations affecting the point must be replayed. Furthermore, after significant coarsening, existing distortion measures can not be utilized to compute the geometric distortion induced by the mapping. This is because significant decimation introduces 'kinks' in the edges of triangles as shown in Figure 1. These bent edges render typical per-triangle distortion measures ill defined since such measures assume that a triangle from the original fine mesh remains a triangle under the mapping[12,8].

In this work we demonstrate that 'kinked' edges are caused by the interleaving of edge flipping and conformal flattening operations used by current approaches. We then present an algorithm that avoids kinks by carrying out all flattening operations prior to any flipping oper-

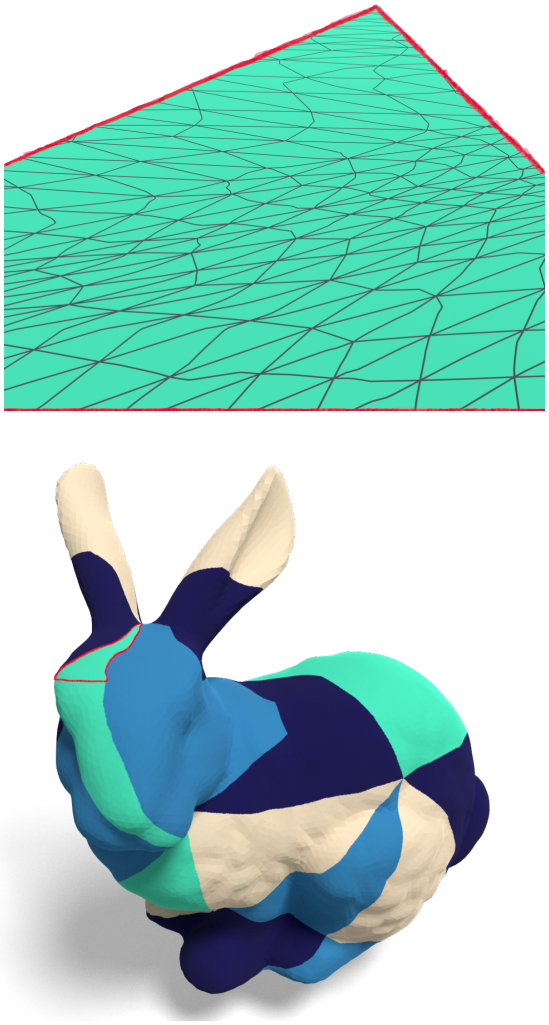

Figure 1: Significant decimation of the bunny model, reducing the number of vertices from 14290 vertices to 14, introduces 'kinks'. The aquamarine intrinsic triangle on the bunny (outlined with red in the bottom figure) contains the projections of many triangles from the original fine mesh (top), many of which are bent/kinked.

ations. The algorithm first applies the CETM algorithm[17] globally to flatten the mesh away from select cones. The algorithm then uses edge flipping to remove the flattened vertices to achieve the decimation target. By contrast current intrinsic simplification approaches interleave flattening and flipping operations by applying CETM locally to flatten the neighborhood of a vertex before immediately removing it via edge flips.

Related Work

Intrinsic Triangulations

Our method, and previous intrinsic mesh simplification algorithms operate on intrinsic triangulations. Intrinsic triangulation data structures enable the decoupling of the triangulation used to represent the intrinsic geometry of a surface from the triangulation used to represent its embedding in $\mathbb{R}^3$. Decoupling the geometries allows algorithm to efficiently operate on intrinsic data by ignoring unnecessary extrinsic information. Furthermore, given a vertex set, the set of intrinsic triangulations is immense compared to the set of extrinsic triangulations. Operating in this larger space allows more freedom when computing a triangulation for a specific application[14]. Intrinsic triangulations frameworks provide a set of operations that can be used to modify an intrinsic triangulation. An early functional intrinsic triangulation framework, the incremental overlay, was introduced by Fisher et al.[3] and subsequent frameworks have expanded the set of operations available[14,5]. Example operations include vertex insertion, edge spits, and face splits. The key intrinsic operation required for existing intrinsic mesh simplification algorithms, as well as our method, is the intrinsic edge flip, illustrated in Figure 2. The frameworks of Sharp[14], and Gillespie[5] provided a means to *flip away*, i.e. remove vertices, but only if they were previously inserted. Prior to intrinsic mesh simplification algorithms, intrinsic triangulation frameworks did not provide a means of removing a vertex contained in the corresponding extrinsic triangulation.

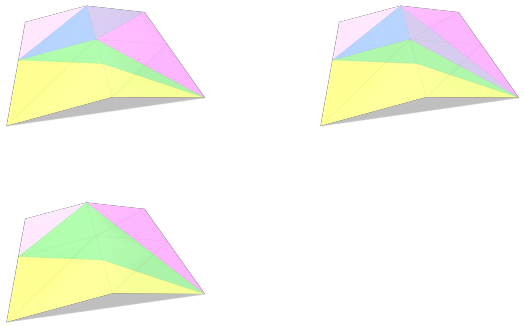

Figure 2: An example of an intrinsic edge flip. The edge between the grey and pink triangles is flipped. Notice that after the edge flip the central vertex has valence 3 and is contained in a triangle. Removing the central triangle yields the bottom Figure. We call the combination of flips and removal *flip away*.

Mesh Simplification

There is vast body of literature concerning mesh simplification. Here we only briefly discuss concepts necessary to motivate our work and refer the reader to[7] for an in depth survey. The goal of mesh simplification is to reduce the number of elements, i.e. vertices and faces, required to represent a surface. The number of vertices is reduced whilst a desired quality measure is maintained. The process of reducing the number of elements is referred to as *coarsening*. Classic, extrinsic mesh simplification algorithms coarsen the mesh while preserving some extrinsic measure, such as visual appearance[4]. In contrast, we present an intrinsic coarsening algorithm which aims to preserve the intrinsic geometry of the surface.

Recently, intrinsic mesh coarsening was introduced simultaneously by Liu et al.[11] (ICE) and our previous work, Shoemaker et al.[15] (IMS). Both methods achieve coarsening by greedily selecting vertices for removal before flattening the neighborhood of the vertex and removing it. In contrast, our method achieves coarsening by first flattening all desired vertices simultaneously before any vertex is removed. In order to minimize mesh distortion both ICE and IMS maintain a deletion queue for vertices ordered by an intrinsic metric. Our method, however, can remove vertices arbitrarily since only the flattening operation introduces distortion. Both IMS and ICE utilize edge flips to aid in vertex removal by reducing the valence, or number of incident edges, of a vertex to three. We refer to this operation as *flip away*. When a vertex is incident to three edges it is contained in a triangle formed by the three vertices that neighbor it, making removal trivial. Since IMS and ICE interleave flattening and edge flipping they both introduce kinked edges after significant decimation. These kinks make typical distortion measures ill defined Our method ensures that kinks do not occur, ensuring a higher quality map.

Conformal Parameterization

Our method, as well as previous mesh simplification algorithms utilize the CETM algorithm of Springborn et al.[17] to flatten vertices prior to removal. IMS and ICE both apply CETM locally, to flatten a single vertex at at time, whereas our algorithm flattens desired vertices simultaneously. CETM was designed for conformal mesh parameterization of a mesh. It does so by scaling the edges of a mesh such that its vertices have 0 Gaussian curvature except for a set of cones. The per edge scales are computed by minimizing an energy function via Newton's method.

Background

We briefly present the concepts and notation required for our exposition.

The intrinsic geometry of a surface is completely described by a mesh $\mathsf{M} = \{\mathsf{V}, \mathsf{E}, \mathsf{F}\}$ of vertices V, edges E, and F faces along with a function $\ell : \mathsf{E} \to \mathbb{R}^+$, called the *metric*, describing the lengths of edges. We typically denote vertices using single variables, usually i, j, k, and edges with two variables grouped together, for example ij. Faces are denoted by using three variables grouped together, as in ijk. This notation allows us to specify the relationship between mesh elements. For example suppose $i, j, k, m \in \mathsf{V}$ denote the vertices of triangles $ijk, jim \in \mathsf{F}$ as in Figure 3. Given ijk

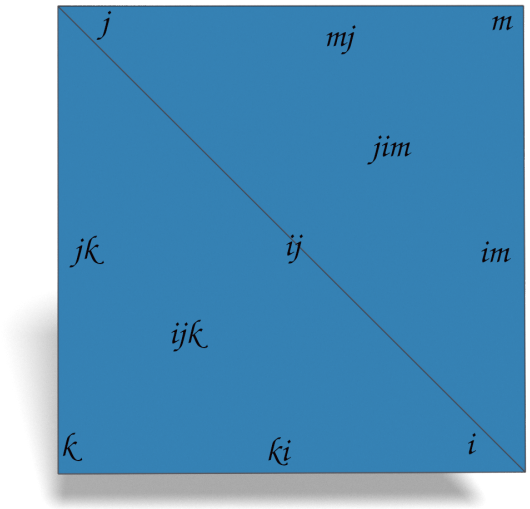

Figure 3: Neighboring triangles ijk and jim. Single variables represent vertices, a pair of variables represents an edge, and a tripple represents a face.

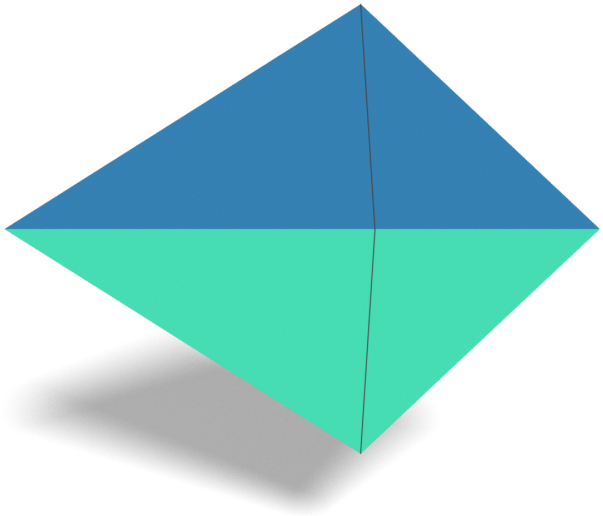

Figure 4: Interleaving intrinsic edge flips and edge scaling can introduce 'kinked' edges.

and jim, we may deduce that ijk connects the vertices i, j, and k and contains edges ij, jk, and ki. We can also deduce that ijk and jim share edge $ij \in \mathsf{E}$.

θ_i^{jk} denotes the corner angle in triangle ijk at vertex i and can be computed using edge lengths and the law of cosines. We denote the total angle around a vertex $\Theta_i = \sum \theta_i^{jk}$, and $\kappa_i = 2\pi - \Theta_i$ is the Gaussian curvature at i.

A pair of intrinsic triangulations with the same connectivity $\mathsf{M} = \{\mathsf{V}, \mathsf{E}, \mathsf{F}\}$ and edge lengths ℓ and $\tilde{\ell}$ respectively are related by a discrete conformal equivalence[17] iff there is a function $u : \mathsf{V} \to \mathbb{R}$ such that $\tilde{\ell}_{ij} = \ell_{ij} e^{(u_i + u_j)/2}$ for every edge $ij \in \mathsf{E}$.

Analysis and Method

Our goal is to coarsen a given mesh M_0 (described intrinsically or extrinsically), yielding a coarse intrinsic triangulation $\mathsf{M}_*$with a conformal mapping taking the triangles of M_0 to geodesic triangles embedded in the intrinsic triangulation M_*. We first describe the causes of kinked edges and afterwards present our algorithm avoiding them.

Avoiding 'Kinked' Edges

Kinked edges arise when the intrinsic operations of edge scaling and edge flipping are interleaved. Consider the triangulated square in Figure 3. We shall consider two scenarios. In the first scenario we shall apply an edge flip before a conformal scaling, and in the second scenario we shall apply a conformal scale prior to an edge flip.

In the first scenario we flip the interior edge and apply a conformal scale u_k at vertex k, scaling edges ℓ_{jk}, ℓ_{ki}, and l_{mk} by $e^{u_k/2}$ resulting in Figure 4. Note that flipping prior to scaling caused the original edge ij to have a slight bend. However scaling is performed prior to flipping then bending is avoided. This provides motivation for our algorithm; if edge flips appear after all conformal scaling bent edges can be avoided.

Coarsening via Decoupled Flattening and Removal

Recall our goal is to decimate a given mesh M_0 with metric ℓ to achieve a target number of vertices N. At a high level, Algorithm 1 describes our decimation strategy. We first apply the pro-

cedure `CETM` to flatten all but N vertices, yielding a per vertex function u. The conformal factors u are then used to scale the edges of M_0, flattening the metric at all vertices except N cones. Our algorithm then attempts to remove all flat vertices via the procedure `Flip_Away`, shown in Figure 2. `Flip_Away` applies intrinsic edge flips to a vertex until its valence is reduced to 3, at which time it is contained inside a triangle, making removal trivial. For a boundary vertex `Flip_Away` first reduces the valence to 2 before removal.

Algorithm 1 first applies the `CETM` algorithm to flatten all vertices except N cones. A natural question arises: how does one select which vertices to flatten, or equivalently, which ones do we choose as cones? Computing cones is non-trivial due to the fact that each choice of cone changes the global geometry. Cone selection is an active area of research with many approaches[10,2,9,16]. Computing globally optimal cones is computationally intensive, so to keep our method feasible for mesh simplification we must use a heuristic. Following Springborn et al.[17] we use automatic cone placement by taking a few newton steps and iteratively selecting the vertex with the largest conformal factor in magnitude. The vertex with the largest conformal factor has a larger impact on incident edges and so introduces the largest distortion in its neighborhood. When comparing to ICE or IMS we use the same cones to get an apples to apples comparison.

Algorithm 1

Input: Mesh $\mathsf{M}_0 = (\mathsf{V}_0, \mathsf{E}_0, \mathsf{F}_0)$;
Metric $\ell : \mathsf{E} \to \mathbb{R}^+$;
Desired number of vertices N
Output: Intrinsic Triangulation Mesh $\mathsf{M}_* = (\mathsf{V}_*, \mathsf{E}_*, \mathsf{F}_*)$ s.t. $|\mathsf{V}_*| = N$;

$\tilde{\ell} : \mathsf{E} \to \mathbb{R}^+$
$u \leftarrow \text{CETM}(M_0, \ell, N)$ // Compute conformal factors
for edge $ij \in \mathsf{E}_0$ **do**
  $\ell_{ij} \leftarrow e^{(u_i+u_j)/2}\ell_{ij}$ // Apply conformal factors
end for
$(\mathsf{M}_*, \tilde{\ell}) \leftarrow \text{Flip\_Away}(\mathsf{M}_0, \ell)$ // Remove flat vertices via edge flips
return $(\mathsf{M}_*, \tilde{\ell})$

Discussion

The CETM algorithm has many desirable properties. It formulates conformal flattening (and in general curvature prescription) as a convex optimization problem, ensuring it has a global minimizer. The gradient of the convex energy is simply half the difference between the desired total angle of a vertex $\hat{\Theta}$ and the total angle as a function of the conformal factors, which can be easily computed. Furthermore, the Hessian is half of the Cotan Laplacian, which is sparse, symmetric and easily computed. While CETM has many desirable properties, our choice of CETM is not without consequences.

First, for CETM to converge it may need to traverse a configuration of conformal factors that induces degenerate triangles into the mesh. Springborn et al.[17] provide a means to extend the acceptable values of u to all of $\mathbb{R}^{|V|}$. However, due to numerical issues, the method can fail to converge, especially when encountering challenging geometries and may even fail on 'nice' triangulations. Another issue is that the conformal factors minimizing the energy may themselves induce degenerate triangles. This latter problem is handled by Gillespie et al.[6], and separately by Campen et al.[1] by modeling the problem with ideal hyperbolic triangulations, enabling the use of Ptolemy edge flips and careful projection schemes. However, both methods interleave edge flipping and scaling, leaving open the possibility of kinked edges. Since our goal is to address the issue of kinked edges we disallow edge flips prior to conformal scaling, and are susceptible to the shortcomings of CETM. Navigating the interplay between convergence and bent edges is an active and ongoing area of research.

A further consequence of using CETM is computational cost. The CETM energy is minimized by taking newton steps, each of which involves solving a linear system involving the Hessian and gradient. In practice the cost of solving a system involving the Laplacian is lin-

ear in the number of vertices so each newton step is roughly linear, which may become prohibitively expensive for large meshes. The issue will be more pronounced with poor triangulations. Adopting a local or local-global hybrid approach to more efficiently avoid kinked edges is an ongoing focus of research.

Results and Evaluation

To understand the trade offs of our algorithm, we fist compare and contrast it to a previous intrinsic coarsening algorithm, ICE[11]. Afterwords, we show some examples of our algorithm with high decimation levels.

We ran our algorithm and ICE on a diverse set of 50 meshes from the Thingi10k[19] dataset. We use three per-triangle distortion measures to compare the two methods. The first two measures are the Anisotropic and Area distortion measures of Khodakovsky et al.[8], and the third measures the relative change in the length of the edges from the original mesh. In order to compute these distortion measures we must first compute the length of an extrinsic edge as it is embedded in the coarse intrinsic triangulation. For our algorithm we can easily compute the edge lengths from the conformal factors. Computing the edge lengths for ICE (and IMS) is not trivial. Due to the presence of kinks, the embedding of an extrinsic edge is not guaranteed to be a shortest geodesics, or even geodesics, in the coarse mesh. We get an estimate of the length of an embedded edge by tracking the embedding of the edge during simplification. We accomplish this by inserting pilot points on the original edge before any simplification occurs. The pilot points are tracked as ICE progresses and after simplification the length is estimated as the sum of the segments between the pilot points. Clearly if one could track *every* point of an edge then the true length could be computed. In practice we insert hundreds of points per extrinsic edge, depending on its length.

As we noted previously, due to the presence of kinked edges, per triangle distortion measures are unable to give a faithful estimate of distortion. So additionally we keep a count of when the embedding of a triangle violates the triangle, as can occur in the presence of kinked edges. Table 1 shows a comparison our Algorithm 1 to ICE using Anisotripic, Area, and length distortion measures. Our algorithm performs much better than ice when it comes to anisotropic distortion. The presence of heavy anisotropic distortion is consistent with the presence of kinked edges. However, ICE is better able to preserve area than our algorithm. It appears that ICE is, on average better able to preserve the metric, but this appears to be due to the presence of an outlier as the medians are comparable.

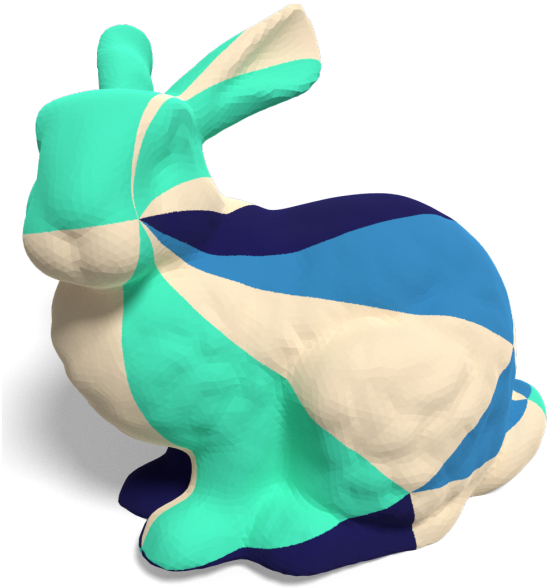

Figure 5: An example of our algorithm with high coarsening. The original mesh contained $14,290$ and the coarsened meshes each contain 14.

Figures 5 and 6 show two meshes by heavily coarsening with our algorithm. Notice that no kinked intrinsic edges occur as is the case with the mesh produced by ICE in Figure 1.

Conclusion

We examined the issue of kinked edges produced by current intrinsic mesh simplification algorithms, including our previous work. Intrinsic mesh simplification algorithms are able

Table 1: Mesh simplification statistics over a subset of 50 triangle meshes from the Thingi10K dataset comparing our Algorithm 1 to ICE.

	Anisotropic		Area		Length	
	OURS	ICE	OURS	ICE	OURS	ICE
mean	0.40	1.33e+06	1.02	0.90	1.48e+45	9.03e-01
median	0.09	1.49+03	1.05	3.81e-3	9.90e-01	1.85e-01
std	1.50	3.22e+06	20.7	1.62	1.70e+46	1.58e+00
min	0.00	4.16e-07	-110	-1.58	0.00e+00	1.34e-07
max	13.6	2.28e+07	29.4	7.00	1.96e+47	1.10e+01

Figure 6: An example of our algorithm with high coarsening. The original mesh contained $14,290$ and the coarsened meshes each contain 14.

to achieve coarsening by scaling mesh edges until a vertex is flat, when it can be safely removed by intrinsic edge flips. These steps are repeated until the desired decimation level is achieved. We illustrated that bent, or kinked extrinsic edges occur when intrinsic edge flips are interleaved with conformal scaling of mesh edges. We presented our algorithm for intrinisic mesh simplification that avoids causing kinked extrinsic edges by first conformally scalling all edges until a desired number of vertices are flat. Our algorithm then safely removes flat vertices via edge flips until the desired decimation level is achieved. We compared our method to ICE, an existing mesh simplification algorithm, and exhibited a few meshes to illustrate that our method avoids kinked edges.

Acknowledgements

I would like to express my appreciation for my Ph.D. advisor Pieter Peers for his patience and guidance. I also wish to express my gratitude to the Virginia Space Grant Consortium for funding this research.

References

[1] Marcel Campen et al. “Efficient and robust discrete conformal equivalence with boundary”. In: *ACM Transactions on Graphics (TOG)* 40.6 (2021), pp. 1–16.

[2] Qing Fang et al. “Computing sparse cones with bounded distortion for conformal parameterizations”. In: *ACM Trans. Graph.* 40.6 (2021). ISSN: 0730-0301. DOI: 10.1145/3478513.3480526. URL: https://doi.org/10.1145/3478513.3480526.

[3] 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.

[4] 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.

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

[6] Mark Gillespie, Boris Springborn, and Keenan Crane. “Discrete Conformal Equivalence of Polyhedral Surfaces”. In: *ACM Trans. Graph.* 40.4 (2021).

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

[8] Andrei Khodakovsky, Nathan Litke, and Peter Schröder. “Globally Smooth Parameterizations with Low Distortion”. In: *ACM Trans. Graph.* 22.3 (2003), 350–357. ISSN: 0730-0301. DOI: 10.1145/882262.882275. URL: https://doi.org/10.1145/882262.882275.

[9] Mo Li et al. “Computing sparse integer-constrained cones for conformal parameterizations”. In: *ACM Trans. Graph.* 41.4 (2022). ISSN: 0730-0301. DOI: 10.1145/3528223.3530118. URL: https://doi.org/10.1145/3528223.3530118.

[10] Mo Li et al. “Efficient Cone Singularity Construction for Conformal Parameterizations”. In: *ACM Trans. Graph.* 42.6 (2023). ISSN: 0730-0301. DOI: 10.1145/3618407. URL: https://doi.org/10.1145/3618407.

[11] Hsueh-Ti Derek Liu et al. “Surface simplification using intrinsic error metrics”. In: *ACM Transactions on Graphics (TOG)* 42.4 (2023), pp. 1–17.

[12] Pedro V Sander et al. “Texture mapping progressive meshes”. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001, pp. 409–416.

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

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

[15] Randy Shoemaker, Sam Sartor, and Pieter Peers. *Intrinsic Mesh Simplification*. CoRR, abs/2307.07115. 2023. DOI: https://arxiv.org/abs/2307.07115.

[16] Yousuf Soliman, Dejan Slepčev, and Keenan Crane. “Optimal Cone Singularities for Conformal Flattening”. In: *ACM Trans. Graph.* 37.4 (2018).

[17] Boris Springborn, Peter Schröder, and Ulrich Pinkall. “Conformal Equivalence of Triangle Meshes”. In: *ACM Trans. Graph.* 27.3 (2008), 1–11. ISSN: 0730-0301. DOI: 10.1145/1360612.1360676. URL: https://doi.org/10.1145/1360612.1360676.

[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).