Eurographics Symposium on Geometry Processing 2017
J. A. Bærentzen and K. Hildebrandt
(Guest Editors)

Volume 36 (2017), Number 5

A Constrained Resampling Strategy for Mesh Improvement

Ahmed Abdelkader[1]*, Ahmed H. Mahmoud[2]*, Ahmad A. Rushdi[2], Scott A. Mitchell[3], John D. Owens[2], and Mohamed S. Ebeida[3]†

[1]University of Maryland, College Park [2]University of California, Davis, CA [3]Sandia National Laboratories, Albuquerque, NM

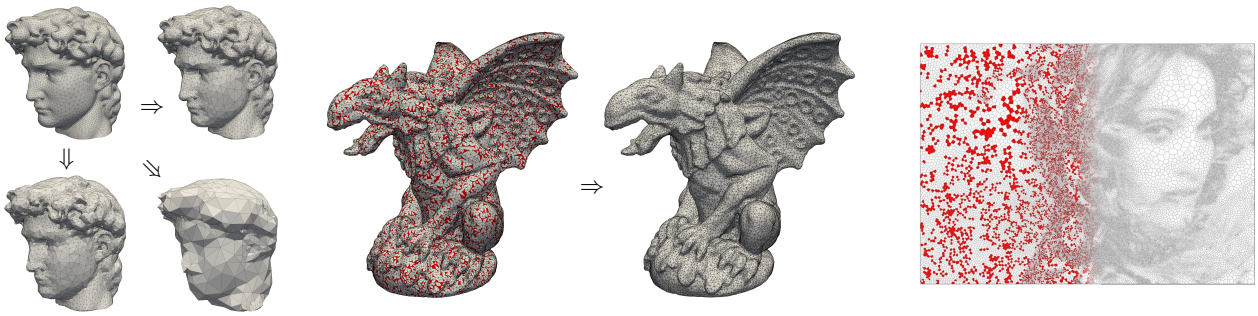


Figure 1: *Sample applications of the proposed strategy. Left figure shows the David Head model: initial mesh (top left),* 30% *sifted mesh (top right),* 60% *mildly-simplified mesh (bottom left), and* 95% *extremely-simplified mesh (bottom right). Middle figure shows the successful elimination of obtuse triangles on a challenging Gargoyle model. Right figure shows the elimination of short edges from a Voronoi mesh.*

Abstract

In many geometry processing applications, it is required to improve an initial mesh in terms of multiple quality objectives. Despite the availability of several mesh generation algorithms with provable guarantees, such generated meshes may only satisfy a subset of the objectives. The conflicting nature of such objectives makes it challenging to establish similar guarantees for each combination, e.g., angle bounds and vertex count. In this paper, we describe a versatile strategy for mesh improvement by interpreting quality objectives as spatial constraints on resampling and develop a toolbox of local operators to improve the mesh while preserving desirable properties. Our strategy judiciously combines smoothing and transformation techniques allowing increased flexibility to practically achieve multiple objectives simultaneously. We apply our strategy to both planar and surface meshes demonstrating how to simplify Delaunay meshes while preserving element quality, eliminate all obtuse angles in a complex mesh, and maximize the shortest edge length in a Voronoi tessellation far better than the state-of-the-art.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

A mesh is a discrete representation of a geometric domain, convenient for computing. Generating good quality meshes is a key step in geometry processing pipelines, e.g., graphics and visualization [AUGA08, LM15], finite element analysis [HL88] and computer-aided design [Lee99, TBG09, RMM*16].

In this paper, we address the problem of improving an input mesh in terms of a given set of quality objectives. While it is easier to achieve one objective at a time, improving one property without degrading others is a challenging problem for several graphics applications, including simplifying oversampled 3D scan data, level-of-detail (LoD) rendering, and isosurface extraction.

For example, mesh simplification attempts to reduce the size in order to decrease the computational cost of subsequent tasks such as rendering, simulation, animation, etc. In order to preserve visual fidelity, it is necessary to tolerate only marginal changes within some accepted tolerance. This class of problems includes mesh smoothing, mesh deformation, parameterization, surface approximation and mesh segmentation [BKP*10].

*Indicates equal contribution

†Corresponding author: msebeid@sandia.gov

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

Contribution. We describe a versatile strategy for mesh improvement by interpreting quality objectives as spatial constraints on resampling. Our main contribution is a derivation of a succinct spatial representation for points satisfying various objectives using a collection of geometric primitives, which allows us to extend known resampling operators to a larger class of problems and greatly simplifies the implementation of complex constraints.

Leveraging ideas from smoothing and transformation techniques, we obtain a practical approach to achieve multiple objectives simultaneously. We develop a toolbox of resampling operators that can be scheduled to achieve a wide range of quality objectives. These objectives include, but are not limited to, mesh simplification by removing vertices while preserving angle bounds; elimination of obtuse angles; improving angle, edge length and aspect ratio bounds; and elimination of short edges in Voronoi tessellations. Typical inputs to our strategy are produced by standard meshing packages. While we do not guarantee an improvement in terms of input quality, our strategy may not make much progress on random inputs; it is more of a clean-up process than a standalone mesher.

Figure 1 illustrates a few examples. The David Head model is used to demonstrate *Delaunay sifting*, i.e., simplification while preserving all angle, edge length, and smoothness bounds, on top of traditional *simplification* which trades-off smoothness for fewer vertices to yield lower LoDs. The Gargoyle model demonstrates the successful elimination of all obtuse angles from a complex model with highly detailed features. Finally, given a sizing function defined by a grayscale image, the initial Voronoi tessellation was processed to eliminate short Voronoi edges while preserving visual fidelity.

Improving a mesh greedily by a sequence of local updates, i.e., hill-climbing, is hardly novel and was previously explored, for example, in [Joe89, KS08]. What distinguishes our work is the ability to capture the feasibility regions to achieve a wide range of quality objectives by resampling, or detect that local improvement is not possible when the regions are empty, in contrast to other methods that rely on a few deterministic rules, e.g., Delaunay refinement and off-centers for Delaunay meshing. We believe the local resampling operators we develop can greatly enrich existing tools for mesh improvement, e.g., [BDK*03], as demonstrated by our results.

2. Related Work

We summarize the most related work under three categories. A more comprehensive account can be found in [AUGA08, BPK*07].

Mesh Improvement and Quality Remeshing: *Smoothing* methods represented by the Centroidal Voronoi Tessellation (CVT) [DFG99] and its variants [DGJ03, WHWB16, JFL14, DW03] work by moving vertices to optimize an energy function. Other optimization-based smoothing techniques for various quality objectives including angle bounds, edge length and triangle areas compute locally optimal moves [ABE99, Ren16]. On the other hand, *transformation* methods may add or remove vertices and work by vertex clustering [LT97, SW03], vertex removal [SZL92], edge and half-edge collapses [HDD*93, ATC*08], and incremental decimation [WK03, KCS98, EMA*13]. Our work is inspired by Poisson-disk sampling which we discuss in more details in Section 3; see the recent work in [AGY*16].

Mesh Simplification and Feature Preservation: It is often desired to reduce the size of data, e.g., during surface reconstruction [PGK02, MD03]. This often results in sacrificing critical features [CY16]. The Quadric Error Metric (QEM) methods achieve simplification by optimizing the position of the vertex to collapse into [GH97], with no guarantees on the angle bounds or smoothness of the output. Unlike the majority of remeshing algorithms that assume features are specified in advance, e.g., [GYJZ15], our constraints implicitly preserve features up to a smoothness parameter, which makes it closer to the approach in [LT98]. The recent work in [HYB*16] is similar to ours as they consider multiple objectives, i.e., angle bounds and the Hausdorff error, along with reducing the vertex count. Although their approach can be superior in Hausdorff error, our approach is simpler and more versatile.

Delaunay Refinement (DR): Starting with an initial triangulation, DR repeatedly refines any triangle with a small angle by inserting its circumcenter as a new vertex. Despite the theoretical guarantees on the asymptotic mesh size, unnecessarily high densities are often produced in practice, such as in *Triangle* [She96, She02]. For smaller meshes of better quality, *aCute* unifies vertex insertion schemes (circumcenter, sink, off-center) and incorporates smoothing [EÜ09b, EÜ09a]. Our work can be regarded as a generalization of these schemes as we define feasible regions for resampling rather than a few specific points. For surface remeshing, DR provably improves quality and can simplify with various guarantees [BO05] while preserving sharp features [CDS12]. To achieve multiple criteria in practice, interleaving DR with optimization turns out to be effective [TWAD09]. We adopt a similar paradigm using combinations of operators that work well in practice for different objectives. The limitations of the implementation reported here with respect to surface meshes are discussed in Section 4.

3. The Strategy

Our mesh improvement strategy is inspired by Maximal Poisson-disk sampling (MPS): once a point p_i is sampled, all future samples are constrained to lie outside a sphere of radius r at p_i, which guarantees an inter-sample distance at least r. On the other hand, a sampling is *maximal* if no more points can be sampled. Maximality guarantees for each point in the domain, there exists a sample no farther than r. Variants of MPS use different spatial constraints to provide different quality bounds [MREB12, YW12, EMA*13]. Observe that MPS only uses spheres as sampling constraints. In this work, we generalize this concept, tailoring new spatial constraints to capture a larger class of quality objectives.

We start with a few definitions: *bad elements* are mesh faces that fail to satisfy all quality objectives, as well as any of their constituent vertices and edges; a *patch* is a set of faces associated with a chosen vertex or edge (e.g. triangle fan or two opposite triangles); finally, a *void* Ω is the hole created in the mesh by removing some elements (e.g. vertices and associated faces) from a chosen patch.

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd.

3.1. Algorithm Overview

Input: A mesh $\mathcal{M}$ and a set of quality objectives.

Steps:

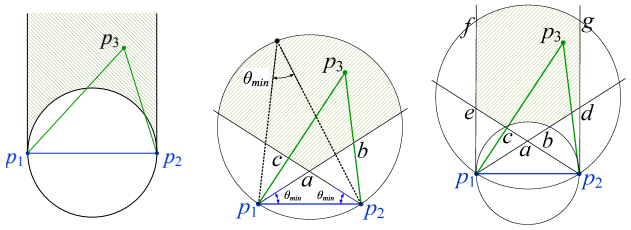
1. Pick a patch where quality objectives are not satisfied. Delete the elements of this patch, creating a void Ω.
2. Map each quality objective into a set of spatial constraints $\mathcal{C}$ on resampling over Ω (Section 3.2), using a collection of geometric primitives (e.g., half-spaces and spheres) or their complements. Define the feasibility region $\mathcal{F} = \cap\mathcal{C}$.
3. If $\mathcal{F}$ is empty, restore the original patch. Otherwise, sample from $\mathcal{F}$ (Section 3.3) and retriangulate Ω.
4. Iterate over all patches (sequentially or in parallel) until the objectives are satisfied or no further improvement is possible.

Output: An improved mesh $\mathcal{M}'$, at least as good as $\mathcal{M}$, since no degradation in quality is allowed w.r.t. the specified objectives.

In step (1), the patch is chosen by iterating over all mesh vertices or elements and testing the quality objective under consideration. If the quality is not satisfied, the patch is processed. Note that if the input mesh satisfies all objectives, we return the same input mesh.

3.2. From Quality Objectives to Spatial Constraints

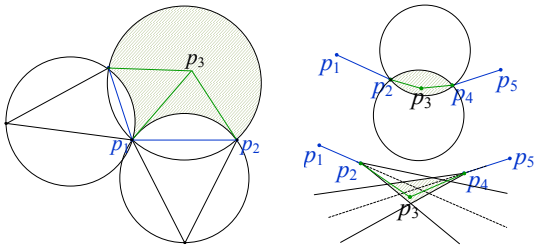
The goal of mapping quality objectives into spatial constraints in step (2) is to define the feasibility region $\mathcal{F}$. Spatial constraints can typically be classified into two types: *inclusion* and *exclusion*. The *inclusion region* $\mathcal{I}$ is the intersection of geometric primitives that must contain the new sample, while the *exclusion region* $\mathcal{O}$ is the union of primitives where the sample is not allowed. Clearly, $\mathcal{F} = \mathcal{I} \setminus \mathcal{O}$. We avoid an explicit construction of $\mathcal{F}$, which can be quite complex. Such feasibility regions $\mathcal{F}$ are similar to the voids created during the MPS process [EMP*12], where a grid-based refinement was employed to track $\mathcal{F}$ up to machine precision or consider it empty.



(a) *Nonobtuse.* **(b)** $\angle p_i \geq \theta_{\min}$. **(c)** *Both objectives.*

Figure 2: *Example primitives for resampling p_3 to form $\triangle p_1 p_2 p_3$: inclusion regions (green), exclusion regions (black boundaries).*

Figure 2 shows examples of mapping minimum and maximum angle bounds into spatial constraints and Figure 3(a) illustrates mapping the Delaunay property. In both figures, we connect one new sample p_3 to two fixed samples p_1 and p_2. In general, our algorithm has the flexibility to *relocate*, *add*, or *remove* multiple vertices (Section 3.3). To preserve sizing functions, we bound minimum and maximum edge lengths. The minimum edge length is controlled by the radii of inter-sample exclusion disks while the maximum edge length is achieved by ensuring that every domain point is included in some (potentially larger) disk. Domain coverage can be achieved by ensuring that every intersection point of two covering circles is covered by some third disk.



(a) *Delaunay disk-free property.* **(b)** *Smooth curvature.*

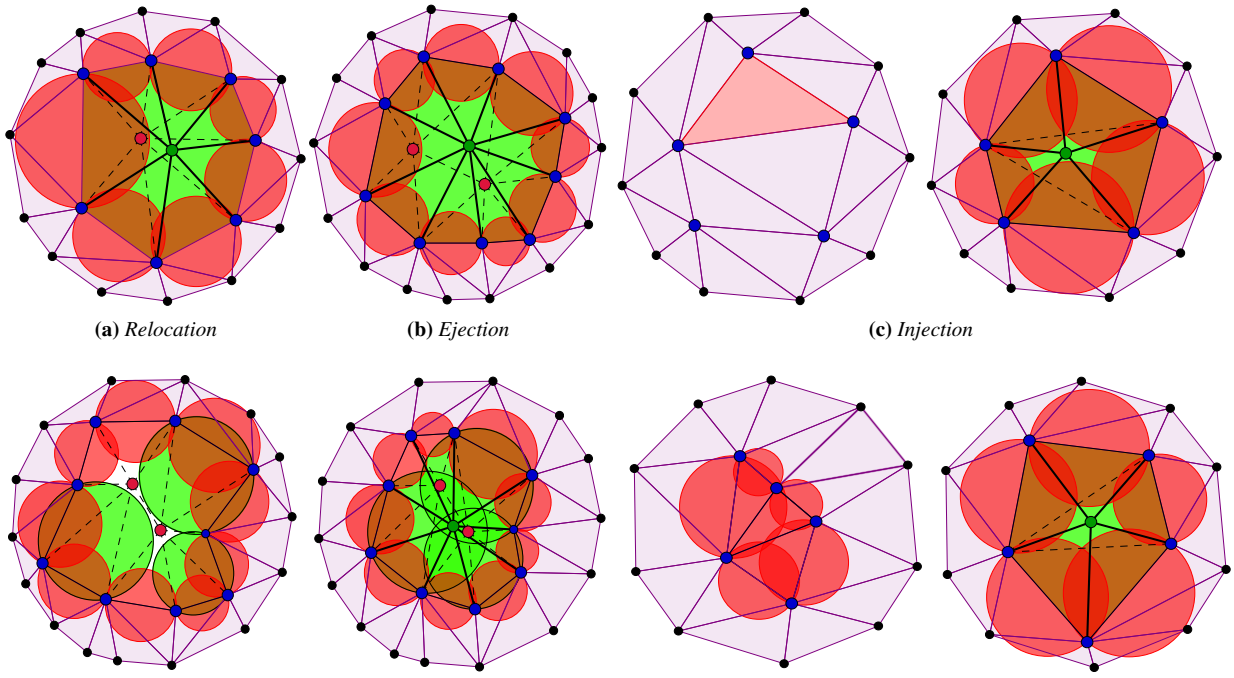
Figure 3: *Examples for resampling p_3 to (a) maintain the Delaunay disk-free property; p_3 should be outside other Delaunay circles (b) preserve a smooth boundary; (top) allowing a bounded deviation on $\angle p_3$ and (bottom) similar bounded deviation on $\angle p_2$ and $\angle p_4$*

3.3. Resampling Operators

In this work, we employ five resampling operators for step (3), summarized in Figure 4. This set of operators was previously used for tuning the density of a sphere packing [ERA*16]. We extend the operators to satisfy other constraints, allowing enough flexibility to achieve more objectives in practice. We emphasize that our strategy can accommodate additional operators as needed. The choice and scheduling of a specific subset of operators depends on the set of objectives required by the application at hand (Section 4.2).

1. **Relocation**: removes one vertex and fills the void by adding a new vertex, which can be viewed as moving the original vertex to a new location. Relocation represents the smoothing technique in our toolbox. This is the simplest operator; it is the least invasive making it the least powerful. We use it whenever it suffices to meet the objective locally. For example, it sometimes succeeds in achieving non-obtuse angles, bounding minimum angles and preserving the Delaunay property.
2. **Ejection**: removes two or three vertices and fills the void by adding a new vertex, with at least one of the ejected vertices being part of a bad-quality element. Ejection helps create a sparser patch where the mesh is locally dense.
3. **Injection**: destroys some bad elements by adding a new vertex. To ensure the new vertex is not irregular, i.e., 3-valent or 4-valent, we include two more triangles into the void by propagating through the edges of the bad triangle.
4. **Attractor Ejection**: ejects a vertex and relocates the vertices bounding the void towards the ejected vertex closing the void, i.e., a combination of ejection and relocation. When relocating the neighbor vertices, we discard the triangles connected to the ejected vertices and consider the quality of all surrounding triangles. We use attractor ejection when simpler operators fail, e.g., when ejection alone is not enough as the void is too large to be filled with a single vertex. This operator creates a denser patch, where subsequent ejections are more likely to succeed.

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd.



(a) *Relocation* **(b)** *Ejection* **(c)** *Injection*

(d) *Attractor Ejection (→ blue ←)* **(e)** *Repeller Injection (← blue →)*

Figure 4: *Sampling operators: red disks are exclusion regions; light green disks/polygons are inclusion regions; dotted lines and dark red vertices are the removed elements and vertices; thick dark lines and dark green vertices are the new elements and vertices; blue are the neighbor vertices. (a) Relocation of a vertex from the exclusion region to the feasible region. (b) Ejection of two vertices, one of which is associated with a bad element. (c) Injection of a new vertex. A low quality (red-ish) triangle is destroyed along with two other triangles such that the new vertex is not irregular (3-valent/4-valent). (d) Attractor Ejection where the intersection of inclusion disks is empty; relocating the neighbor vertices towards the ejected vertices to pack a feasible region. (e) Repeller Injection of new vertex where the exclusion disks cover the void completely; relocating the neighbor vertices farther away from the void creates a feasible region.*

5. **Repeller Injection**: destroys some bad elements, relocates the bounding vertices away to create a larger void, then fills the void by adding a new vertex. The neighbor vertices are relocated while respecting the quality objectives of all surrounding triangles/edges except those that will be destroyed by the newly added vertex. This operator is useful when only removing elements creates too small of a void to inject a new vertex.

4. Implementation Details

To complement the high-level description of the strategy in Section 3, we discuss some crucial aspects of our implementation.

4.1. Surface meshes

Many constraints for planar meshes can easily be extended to surface meshes by replacing circles with spheres, and 2D half-spaces with 3D ones. In addition, preserving smooth curvatures is crucial. Smoothness is measured by the dihedral angle θ^d between two adjacent triangles. Dihedral angles can be bounded by constraining neighboring samples to lie within half-spaces through triangle edges. This is analogous to the 2D example in Figure 3(b). In this paper, we do not handle meshes with sharp features.

We opted to control the deviation of the evolving mesh in terms of dihedral angles instead of explicitly checking Hausdorff errors. While this approach does not guarantee a bound on Hausdorff errors, which is a crucial measure in several applications, our results show that the resulting meshes are reasonably close although other methods can be superior in terms of Hausdorff errors. It is possible, however, to incorporate a suitable overestimator of the Hausdorff error committed by each potential update and reject the update if the error exceeds a given threshold as in [HYB*16]. One caveat is that bounding dihedral angles does not prevent the creation of needle-like features. However, this is highly unlikely and was not observed in any of our experiments on a variety of models. Note that bounding smoothness is related to bounding the number of vertices which is harder to control directly by a sampling approach.

4.2. Operator Scheduling

A *schedule* is a sequence of operators chosen from 3.3 according to the application at hand. Our implementation works in iterations till a stopping condition is met. In each iteration, the schedule starts by applying the first operator to all bad elements before switching to the next operator. Operators can be graded by the magnitude of change they introduce with relocation being the lowest and repeller

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd.

injection and attractor ejection the highest. We prefer to make as little change as possible to achieve the desired objectives. With this in mind, when sample count is not a primary objective, we start with relocation. As relocation does not change the connectivity, it has limited interaction beyond its local patch. This makes it a good choice for constraints that can be achieved locally without adding or removing vertices. However, many constraints, for instance non-obtuse remeshing and maximizing short edges in Voronoi meshes, can rarely be achieved by relocation only. Usually, we next turn to ejection or injection, depending on whether we expect to add or remove vertices. As a lighter mesh is typically preferred, ejection takes precedence. If both ejection and injection fail, e.g., due to complex geometries or very dense or sparse patches, we use our most aggressive operators; repeller injection or attractor ejection, which change both the connectivity and the position of neighbor vertices impacting a larger portion of the mesh. When sample density is an objective, as in mesh simplification or refinement, we start with ejection or injection accordingly.

4.3. Sampling from the Feasible Regions

As mentioned earlier, we avoid an explicit construction of the feasibility region $\mathcal{F}$. For planar surfaces, we use the Simple MPS approach [EMP*12]. We construct an implicit background grid of quads enclosing $\mathcal{F}$, determined by the bounding vertices and spatial constraints. We choose a coarse initial grid (4×4 cells). Then, we uniformly sample a point p from the grid cells and test it against each constraint; if all are satisfied then p is added and we proceed to another patch. Otherwise, p is rejected. After a number of failed attempts proportional to the number of grid cells, we refine every cell by dividing it into four subcells. Subcells completely outside $\mathcal{F}$ are discarded. We recurse, sampling uniformly across the current pool of cells. Because the hierarchy is always flat, the tree does not grow too large, and neither memory nor runtime becomes an issue. This process terminates when cell sizes reach machine precision or all subcells are discarded suggesting $\mathcal{F}$ is empty.

On curved surfaces, the input tessellation plays the role of the grid, similar to [CJW*09]. We pick a triangle t from the pool uniformly by area, sample a point p uniformly from t, then test p against the spatial constraints. If p passes, it is accepted and the patch is retriangulated. Otherwise, we continue as with the grid, refining triangles isotropically into four subtriangles i.e., conformal subdivision. When a sample p has a high probability of being accepted i.e., most of the grid subcells are within the acceptable region, this indicates a large feasible region. In such cases, we do not accept the first feasible p, but instead attempt to generate several, e.g., 10, and accept the best one. Here, "best" depends on the objectives we want to optimize and the bounds we are content to satisfy. For example, we may bound the minimum angle and maximize edge length. We may also use the same metric, e.g., maximizing the edge length locally while bounding the minimum length globally. In Delaunay Sifting, Mesh Simplification and Non-obtuse Triangulation, we use this concept where we accept the sample p with maximum minimum apex angle.

5. Applications

To demonstrate the versatility of our strategy, we develop custom algorithms for different problems. Each problem involves a distinct combination of geometric constraints, which require a suitable selection and scheduling of resampling operators. All experiments were conducted on a PC with Intel® Xeon® CPU E3-1280 v5 @3.70 GHz with 32 GB RAM. For all curved surface results, we used the popular Metro tool [CRS98] to estimate the Hausdorff distance between the original input surfaces and the improved ones. We opt for approximating Hausdorff distances to avoid the high cost of exact computations [BHEK10]; see [TLK09] for recent results on interactive approximations.

5.1. Delaunay Sifting

The goal of Delaunay sifting is to reduce the number of Steiner points from a given Delaunay mesh [AME14]. Steiner points are the set of vertices inserted to refine mesh elements in order to achieve the desired quality [EÜ09b]. Element quality is usually based on angle, edge length or aspect ratio bounds. Delaunay sifting preserves all quality metrics of input meshes along with the Delaunay property, while reducing the number of vertices, unlike standard mesh simplification. The *sifting ratio* α is the percentage decrease in the number of vertices.

5.1.1. Geometric Constraints and Operators

Delaunay sifting only uses the ejection operator, alternating between two variants scheduled as two passes. During the first pass, we iterate over all edges and attempt to eject the two end points forming a void to be retriangulated by resampling a single vertex. For the second pass, we iterate over all triangles attempting to eject all three vertices and resample a single vertex. The intuition is that the first pass helps bring down the density, which allows the more aggressive second pass to achieve even higher reductions. A pass terminates when no more vertices can be ejected.

The geometric constraints for this problem correspond to preserving the following for each triangle affected by the local update:

1. Minimum and maximum angle for both the base edge on the boundary of the void and the apex at the resampled vertex,
2. Delaunay property for both neighboring elements bounding the void and newly formed elements retriangulating it,
3. Smoothness of surface meshes for 3D models, e.g., with $\theta^d \geq 170^\circ$, and boundary edges for 2D models.

For 2D models, we can either disallow sifting boundary elements or restrict resampling to input edges and tolerate a bounded deviation, hence offering more controllability over the regions to sift (see Figure 5). When sifting near the boundary, if one or more of the ejected vertices are on a boundary edge, we ensure that the new vertex lies on the same edge within a margin (Figure 3(b)).

5.1.2. Results and Comparisons

We demonstrate the capability of our algorithm by sifting the output of state-of-the-art 2D Delaunay meshing software: Triangle [She96] and aCute [EÜ09b]. Our method was able to significantly reduce the number of vertices generated by both packages.

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd.

Method	Model	Input Mesh				Sifting Interior					Sifting Interior & Boundary				
		N	△	θ_{min}	θ_{max}	N	△	α	θ_{min}	θ_{max}	N	△	α	θ_{min}	θ_{max}
Triangle	Dolphin	3409	6482	35°	110°	836	1336	75%	35°	109°	466	766	86%	35°	109°
	Lake	2056	3587	35°	110°	1066	1607	48%	35°	109°	801	1225	61%	35°	109°
	Wavy	1164	2041	34°	109°	605	923	48%	34°	109°	242	378	79%	34°	109°
	Batman	859	1505	35°	110°	405	597	52%	35°	109°	271	409	68%	35°	109°
	Airfoil	839	1483	35°	109°	422	649	50%	35°	108°	168	275	80%	35°	108°
	Spiky	715	1207	35°	110°	427	631	40%	35°	108°	265	396	62%	35°	109°
aCute	Dolphin	1125	1939	40°	100°	1033	1755	8%	40°	100°	856	1499	24%	40°	100°
	Batman	580	948	40°	99°	527	842	9%	40°	99°	417	690	28%	40°	99°
	Spiky	552	881	40°	99°	494	765	11%	40°	99°	395	629	28%	40°	99°
	Face	443	715	40°	100°	377	603	13%	40°	100°	272	451	37%	40°	100°

Table 1: *Delaunay sifting of planar meshes: Quality metrics before and after sifting for meshes generated by Triangle and aCute, including the number of samples N, number of triangles △, sifting ratio α and angle bounds* $(\theta_{min}, \theta_{max})$.

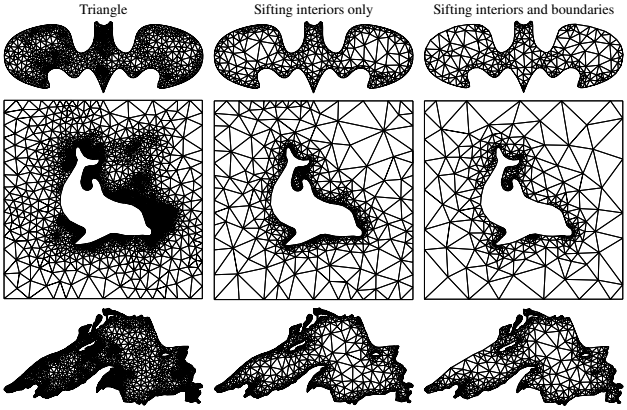


Figure 5: *Delaunay sifting of planar meshes generated by Triangle with* $\theta_{min} \geq 35°$. *These models feature high curvature, narrow regions and sharp corners. Sifting boundaries increases* α.

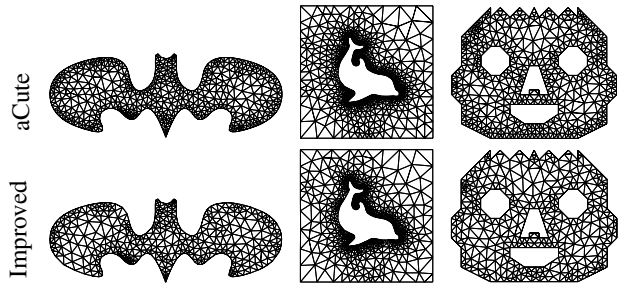


Figure 6: *Delaunay sifting of planar meshes generated by aCute (sifting interiors and boundaries).*

The meshes in Figure 5 were generated by Triangle for a minimum angle bound of 35°. Being the smallest angle bound for which Triangle is guaranteed to terminate, this forces Triangle to insert many Steiner points that turn out to be unnecessary. On the other hand, aCute produces locally optimal Steiner points which enables it to produce high quality meshes with fewer vertices. Still, our tests suggest there is still room for improvement, as shown in Figure 6 and summarized in Table 1. All angle bounds were preserved while achieving sifting ratios of $8\% - 13\%$ for aCute and $40\% - 75\%$ for Triangle when sifting only the interior, and $24\% - 37\%$ for aCute and $61\% - 86\%$ for Triangle when sifting both the interior and boundary. We note that due to selecting the next edge/triangle to process at random and the sampling employed by our operators, our strategy does not preserve the symmetries in the input meshes as can be seen in the batman example (Figure 6).

Next, we apply our algorithm to surface meshes, as shown in Figure 7 for meshes generated by MPS, Delaunay refinement (DR), frontal Delaunay (FD) [EI16], and CVT. Quality metrics before and after sifting are reported in Table 2. On average, we were able to achieve $\alpha \geq 72\%$ across different inputs while preserving all quality metrics including triangle quality Q [YW16] ($Q = \frac{6S}{\sqrt{3}hP}$, where S is the area, h is the longest edge length and P is half the perimeter). To the best of our knowledge, there is no available software for Steiner point removal from surface meshes while preserving the Delaunay property, angle bounds and smoothness.

5.2. Mesh Simplification

In several graphics applications, the user may trade-off the smoothness of a surface mesh, up to some threshold, for fewer vertices. In this application, we allow smoothness of the surface patches to degrade, in contrast to Delaunay sifting, achieving higher reduction ratios. Unlike standard mesh simplification, we require that other bounds, e.g., minimum angles and edge lengths, are preserved. As we discuss next, our current implementation does not explicitly preserve sharp features.

5.2.1. Geometric Constraints and Operators

We use the same geometric constraints and resampling operators as in Delaunay sifting, with a specified bound on θ^d. To demonstrate how changing θ^d results in different resolutions, we gradually decrease θ^d over an MPS mesh on the sphere model shown in Figure 8. We emphasize that angle bounds and the Delaunay property are preserved across the different resolutions. We use θ^d as a parameter to control the trade-off between the number of vertices and

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd.

Method	Model	v			$\triangle$			$\theta_{\min}$		$\theta_{\max}$		$Q_{\min}$		$\theta^d_{\max}$		$d_{RMS}(\times 10^{-2})$	$d_H(\times 10^{-1})$
		Input	Output	%	Input	Output	%	Input	Output	Input	Output	Input	Output	Input	Output		
Uniform MPS	Bunny	11.5K	3.4K	71	23K	7K	71	30	30	116	116	0.5	0.5	180	180	0.14	0.09
	Fertility	8.5K	2.3K	73	17K	5K	73	30	30	116	115	0.5	0.5	180	180	0.18	0.12
	Loop	10.7K	3.4K	68	22K	7K	67	30	30	117	117	0.49	0.49	178	175	0.22	0.15
Delaunay Refinement	Bimba	25.4K	6.3K	75	51K	13K	75	28	28	122	121	0.44	0.45	180	180	0.08	0.09
	Kiss	31.7K	8.5K	73	63K	17K	73	28	28	121	121	0.45	0.45	180	180	0.1	0.12
	Rocker	10.8K	2.5K	76	21K	5K	76	30	30	118	117	0.47	0.48	180	179	0.15	0.14
	Femur	4.1K	1.4K	66	8K	3K	66	30	30	118	116	0.48	0.49	180	177	0.15	0.12
Frontal Delauny	Bimba	24.4K	6.1K	75	49K	12K	75	28	28	121	121	0.46	0.46	180	180	0.09	0.11
	Kiss	30K	8K	74	61K	16K	74	29	29	121	120	0.45	0.46	180	180	0.11	0.16
	Rocker	10.2K	2.7K	74	20K	5K	74	32	32	114	114	0.52	0.52	180	180	0.14	0.14
	Femur	4.0K	1.3K	66	8K	3K	66	31	31	109	108	0.55	0.55	180	177	0.16	0.12
CVT	David Head	15.0K	4.9K	67	30K	9K	67	34	34	99	99	0.63	0.62	180	180	0.22	0.17
	Chine Dragon	30K	9.1K	70	60K	18K	70	39	39	103	103	0.59	0.59	180	180	0.14	0.15
	Omotondo	20K	5.2K	74	40K	10K	74	28	28	110	110	0.54	0.53	180	180	0.1	0.09

Table 2: *Delaunay sifting of surface meshes: Quality metrics before and after sifting, including the number of vertices v, number of triangles $\triangle$, angle bounds $(\theta_{min}, \theta_{max})$, minimum triangle quality (Q_{min}), maximum dihedral angle θ^d_{max}, root mean square distance d_{RMS} and Hausdorff distance d_H (estimated by Metro [CRS98] and normalized by the diameter of the bounding box).*

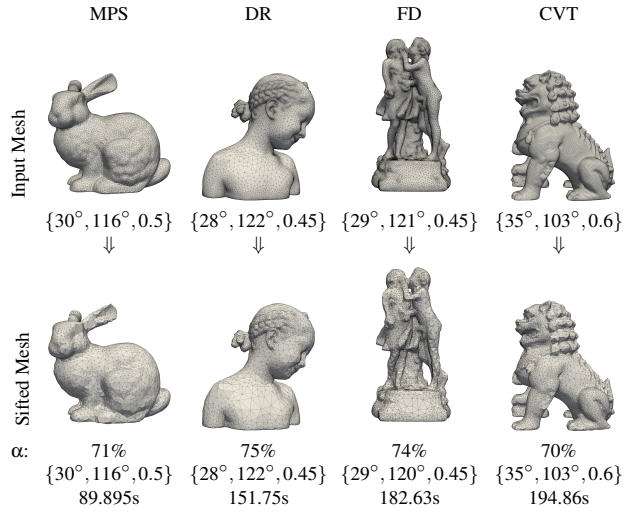


Figure 7: *Delaunay sifting of surface meshes from different sources along with running times in seconds. $\{\theta_{min}, \theta_{max}, Q_{min}\}$ indicate the minimum angle, maximum angle and minimum triangle quality.*

the approximation error; low θ^d achieves lower resolutions while tolerating larger deviations in terms of the Hausdorff distance.

5.2.2. Results and Comparison

We apply the algorithm to several surface meshes generated by different sources; see Figure 9 where different LoDs are achievable by setting different values for θ^d. We compare against the powerful mesh simplification technique; Quadratic Error Metric (QEM) [GH97] and Delaunay mesh simplification (DM) [LXFH15] which is built on top of QEM in order to simplify Delaunay-based meshes. It is worth noting that the DM algorithm preserves the *intrinsic Delaunay* property, in contrast to our algorithm which preserves the *restricted Delaunay* property. Recall that intrinsic Delaunay is based on the geodesic metric while restricted Delaunay is based on the Euclidean metric. In Table 3, we compare these three techniques. For this comparison, we run our algorithm until it converges with $\theta^d = 0$ and use the resulting vertex count as an input for QEM and DM. For the same reduction ratio, the comparison shows the superior performance of our algorithm in terms of preserving the quality metrics of the input mesh (e.g., θ_{mins} and Q_{min}), with the exception of the Hausdorff distance. QEM achieves a better Hausdorff distance since it maintains surface error approximations using the quadric-error metric [GH97]. Still, our algorithm produces competitive results and the visual fidelity of the simplified meshes is not hindered as shown in Figure 9 since all new vertices are sampled from the original input surface.

The closest work to ours which achieves multiple objectives is the recent work by K. Hu et al. [HYB*16]. In their work, the authors use hard constraint over the Hausdorff distance while maximizing minimum angle and reducing the mesh size. However, setting a large tolerance for the Hausdorff error, as required to achieve lower resolutions by simplification, the method in [HYB*16] starts producing self-intersections triangles [Hu17]. Additionally, since there is no guarantees on convergence for a specified bound on the minimum angle, a degradation in the minimum angle has been observed with a few tested models. For instance, with the Loop model, we used the default settings and specified a minimum angle equal to the minimum angle in the input (30°) and the output produced had $4K$ vertices but 7 triangles had an angle less than 30°. Our algorithm at least guarantees no degradation in quality while achieving similar mesh complexity.

5.3. Non-Obtuse Triangulation

In a non-obtuse triangular mesh, no angle is greater than 90°. This guarantees that triangle circumcenters lie within their elements which is a crucial property for some applications in computer graphics [EDD*95, KS98] and scientific computing [ÜS02]. We show how our resampling strategy can be applied to obtain a non-obtuse triangulation starting from given mesh of some model. To further demonstrate the potential of our resampling strategy, we consider applying the same approach to obtain acute triangulations and report some preliminary results.

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd.

Model	Method	v		$\triangle$		θ_{min}		θ_{max}		Q_{min}		θ^d_{min}		θ^d_{max}		$d_{RMS}(\times 10^{-2})$	$d_H(\times 10^{-2})$
		Input	Output	Input	Output	Input	Output	Input	Output	Input	Output	Input	Output	Input	Output		
Bunny (MPS)	DM	11.5K	≈153	23k	≈302	30	12	116	171	0.5	0.06	79	2.5	180	168	3.5	0.8
	QEM						6		165		0.11		54		163	**1.7**	**0.5**
	Our						**30**		**116**		**0.5**		59		172	4.6	1.8
Fertility (MPS)	DM	8.5K	≈390	17k	≈790	30	4.5	116	155	0.5	0.12	91	6	180	178	1.5	0.4
	QEM						4		168		0.08		61		179	**0.7**	**0.2**
	Our						**30**		**116**		**0.5**		55		179	4.86	0.9
Loop (MPS)	DM	10.7K	≈1.4K	22k	≈3K	30	10	117	159	0.48	**0.2**	78	0.8	178	171	1	0.2
	QEM						5.5		160		0.12		67		171	**0.5**	**0.1**
	Our						**30**		**117**		0.17		20		172	2.9	0.4
Bimba (DR)	DM	25.4K	≈180	51k	≈350	28	12	122	137	0.44	0.31	77	11	180	179	4.6	0.6
	QEM						6		161		0.12		60		179	**1.9**	**0.4**
	Our						**28**		**122**		**0.47**		76		165	4.7	1.5
Rocker (DR)	DM	10.8K	≈240	21k	≈485	30	13	118	135	0.47	0.31	102	72	180	167	2.3	0.4
	QEM						4		165		0.09		5		173	**1.1**	**0.3**
	Our						**30**		**114**		**0.5**		55		169	4.9	1.3
Bimba (FD)	DM	24.4K	≈270	49K	≈535	28	11	121	132	0.46	0.3	61	32	180	179	3.3	0.4
	QEM						5		167		0.08		19		171	**0.8**	**0.3**
	Our						**29**		**119**		**0.47**		64		180	4.8	1.2
Rocker (FD)	DM	10.2K	≈260	20.5k	≈520	32	9	114	130	0.52	0.24	106	69	180	177	1.9	0.4
	QEM						5		167		0.09		20		171	**0.8**	**0.3**
	Our						**32**		**112**		**0.52**		50		179	4.9	1.1
Chinese Dragon (CVT)	DM	30K	≈1.5K	60k	≈3.1K	34	11	103	138	0.6	0.27	72	4	180	179	1.5	0.2
	QEM						3		174		0.04		13		179	**0.7**	**0.1**
	Our						**34**		**103**		**0.6**		46		180	2.7	0.4
David Head (CVT)	DM	15K	≈660	30k	≈1.3K	33	8	107	151	0.56	0.2	50	12	180	180	1.1	**0.2**
	QEM						7		158		0.15		11		175	**1.9**	**0.2**
	Our						**33**		**107**		**0.56**		50		180	4.9	0.9
Omotondo (CVT)	DM	20K	≈260	40k	≈530	28	12	110	140	0.54	0.3	90	59	180	167	2.9	0.5
	QEM						7		142		0.18		48		171	**1.1**	0.3
	Our						**28**		**110**		**0.53**		77		180	1.8	**0.2**

Table 3: *Mesh simplification: Comparison against the Quadratic Error Metric (QEM) [GH97] and Delaunay mesh simplification (DM) [LXFH15] across different quality metrics including: the number of vertices v, number of triangles $\triangle$, angle bounds $(\theta_{min}, \theta_{max})$, triangle quality Q_{min}, dihedral angle bounds $(\theta^d_{min}, \theta^d_{max})$, root mean square distance d_{RMS} and Hausdorff distance d_H (estimated by Metro [CRS98] and normalized by the diameter of the bounding box). The best result for each measure is shown in **bold** face.*

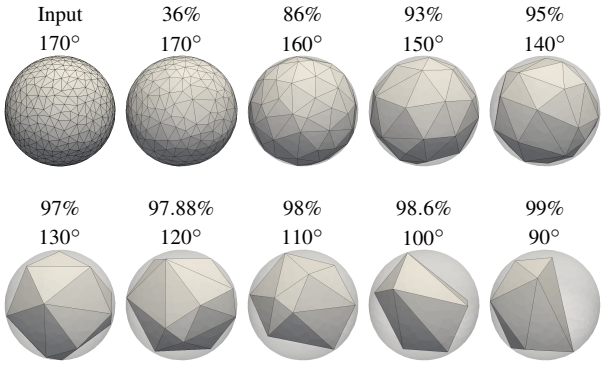


Figure 8: *Tuning the upper bound on dihedral angles θ^d and the corresponding reduction ratio for an MPS mesh on a sphere model (top left).*

5.3.1. Geometric Constraints and Operators

We use the same geometric constraints as in Delaunay sifting, with the addition of bounding angles by 90°. Each iteration uses all five operators from 3.3 in the following order: relocation, ejection, attractor ejection, injection, then repeller injection. For this application, the relocation operator is applied to all mesh vertices as a smoothing phase. Smoothing helps achieve convergence faster as it locally improves the locations of all vertices. A more conservative approach is to apply smoothing only to the neighborhoods of bad triangles, but we do not consider tuning the size of these neighborhoods in the experiments reported here and apply smoothing to all mesh vertices. A patch is chosen by iterating over all triangles and testing for obtuse angles. Relocation suffices to get rid of most obtuse angles, while ejection and attractor ejection help create sparser patches for subsequent relocation to succeed. For certain arrangements, inserting a new vertex is needed for which we use injection and, more aggressively, repeller injection. The algorithm terminates when all obtuse triangles are eliminated.

5.3.2. Results and Comparison

We apply our algorithm to several planar meshes shown in Figure 10 and report the relevant quality metrics in Table 4. The meshes in Figure 10 were generated by Triangle [She96] for a minimum angle bound in $[34^\circ, 35^\circ]$. Being the smallest angle bound for which Triangle is guaranteed to terminate, this is likely to result in tight feasible regions making it challenging for a local approach.

Next, we apply our algorithm to surface meshes, as shown in Figure 11 and report the relevant quality metrics in Table 5. For all input models, our algorithm succeeds in eliminating all obtuse triangles without degrading the minimum angle bound while improving the triangle quality and, as shown by our results, with marginal change in terms of Hausdorff errors. To further demonstrate the capability of our approach, we test our algorithm on the Gargoyle model, deemed an "unsatisfactory example" in [YW16], where 143 obtuse angles were not eliminated. The method in [YW16] fails in the presence of noise and rapid changes in density, since it is inherently a smoothing technique. Our strategy successfully eliminated

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd.

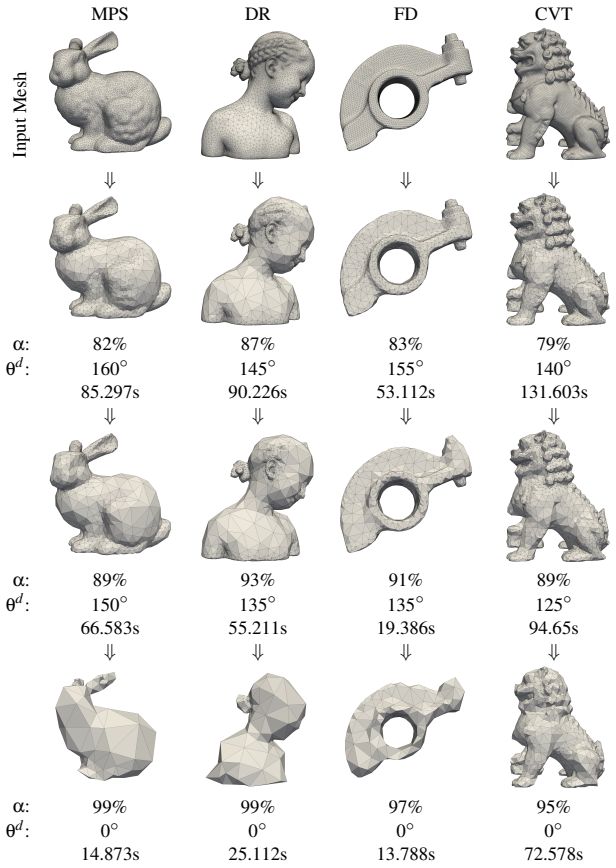


Figure 9: *Mesh simplification of surface meshes from different sources. The minimum dihedral angle θ^d and the reduction ratio α are reported between different simplification levels along with running times in seconds.*

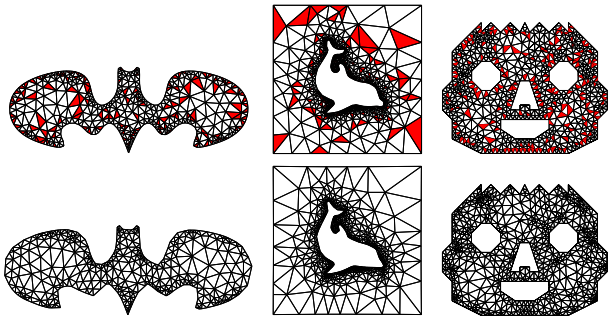


Figure 10: *Non-obtuse remeshing using our algorithm on planar meshes dominated by obtuse triangles (red).*

all obtuse angles while preserving all prominent features, as shown in Figure 1, thanks to the flexibility in applying relocation, addition or removal of vertices to achieve the desired objective. Additionally, the same input model was used to compare against the recent of work of A. G. M. Ahmed et al. [AGY*16] where the density function was computed based on the curvature for adaptive remeshing. However, due to rapid changes in the curvature between different regions, their method could not converge [Guo17]. One might consider using estimates of the local-feature size, instead of the curvature, to estimate the density function to use with [AGY*16]. In that case, the method converged producing a non-obtuse triangulation, but, as seen in Figure 12, some features were smoothed out [Guo17].

	Input Mesh			Output Mesh		
Model	v	θ_{min}	θ_{max}	v	θ_{min}	θ_{max}
Airfoil	839	35°	109°	844	35°	90°
Batman	859	35°	109°	857	35°	90°
Dolphin	3409	35°	109°	3411	35°	90°
Spiky	715	35°	109°	720	35°	90°
Wavy	1164	34°	109°	1171	34°	90°

Table 4: *Quality metrics of non-obtuse remeshing on planar meshes. v indicates the number of vertices, θ_{min} and θ_{max} are the lower and upper angle bounds respectively.*

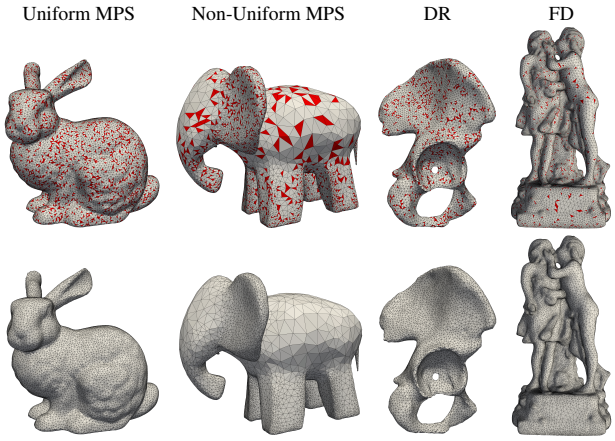


Figure 11: *Example models for eliminating obtuse triangles (red) using our non-obtuse remeshing algorithm.*

5.3.3. Preliminary results for acute remeshing

In order to challenge our algorithm, we explore the possibility of generating *acute* triangulation where all angles are strictly below 90°. We first re-run all the models listed in Table 5 with maximum angle bound of 85°. In this experiment, our algorithm was able to converge without degrading the minimum angle bound and within similar approximate error as shown in Table 5. However, this comes at the cost of running the algorithm longer; it took at most 5 iterations to converge. With maximum angle bound of 80°, our algorithm was not able to converge with any model; with marginal improvement between different iterations.

5.4. Voronoi Meshing without Short Edges

In this application, we aim to produce a Voronoi mesh that has no short edges as required in several applications utilizing explicit numerical simulations. In such applications, short edges lead to small

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd.

Method	Model	v		$\triangle$		$\triangle_{obt}$		$\theta_{\min}$		$\theta_{\max}$		$Q_{\min}$		$d_{RMS}(\times 10^{-3})$	$d_H(\times 10^{-2})$	Time(s)
		Input	Output	Input	Output	Input	Output	Input	Output	Input	Output	Input	Output			
Uniform MPS	Bunny	11.5K	11.4K	23.1K	22.8K	3.5K	0	30.5	30.5	116.2	90	0.5	0.64	0.43	0.49	2.934
	Loop	10.7K	10.6K	21.6K	21.4K	3.1K	0	30.2	30.2	117.4	90	0.48	0.64	0.74	0.59	2.824
	Fertility	8.4K	8.3K	16.9K	16.7K	2.5K	0	30.3	30.3	116.0	90	0.5	0.64	0.48	0.38	2.116
Non-uniform MPS	Elephant1	12.7K	12.5K	25.4K	25.1K	4.2K	0	21.9	21.9	123.4	90	0.4	0.5	0.41	0.65	3.19
	Homer	4.4K	4.3K	8.8K	8.7K	1.4K	0	21.4	21.4	126.1	90	0.4	0.5	0.84	0.83	1.108
Delauny Refinement	Kiss	31.7K	31.3K	63.4K	62.7K	8.2K	0	28.5	28.5	121.6	90	0.45	0.6	0.26	0.21	7.907
	Bimba	25.3K	24.9K	50.7K	49.9K	6.5K	0	28.5	28.5	122.1	90	0.44	0.6	0.23	0.29	6.33
	Rocker	10.9K	10.7K	21.7K	21.4K	2.3K	0	30.0	30.0	118.7	90	0.47	0.63	0.41	0.39	2.592
	Hip	6.4k	6.3K	12.8K	12.5K	1.5K	0	28.6	28.6	119.1	90	0.47	0.62	0.65	0.71	1.533
	Elephant2	6.3K	6.2K	12.7K	12.5K	1.3k	0	30.4	30.4	118.1	90	0.48	0.64	0.67	0.756	1.528
	Femur	4.1K	4.1K	8.3K	8.1K	1.0K	0	30.6	30.6	118.4	90	0.48	0.64	0.41	0.32	0.997
Frontal Delauny	Kiss	30.3K	30.2K	60.7K	60.5K	2.7K	0	28.6	28.6	121.4	90	0.45	0.6	0.26	0.18	2.297
	Bimba	24.4K	24.3K	48.8K	48.6K	2.1K	0	28.5	28.5	120.9	90	0.46	0.6	0.22	0.30	1.81
	Rocker	10.2K	10.2K	20.4K	20.4K	0.32K	0	31.6	31.6	113.7	90	0.52	0.65	0.41	0.38	0.69
	Elephant2	6.0K	6.0K	12.0K	12.0K	0.32K	0	30.6	30.6	112.4	90	0.5	0.64	0.67	0.53	0.457
	Femur	3.9K	3.9K	7.9K	7.9K	0.14K	0	31.1	31.1	109.2	90	0.55	0.65	0.36	0.21	0.267

Table 5: *Non-obtuse remeshing on curved surfaces: Quality metrics include the number of vertices v, number of triangles $\triangle$, number of obtuse triangles $\triangle_{obt}$, angle bounds $(\theta_{min}, \theta_{max})$, root mean square distance d_{RMS}, Hausdorff distance d_H (estimated by Metro [CRS98] and normalized by the diameter of the bounding box), and running time in seconds.*

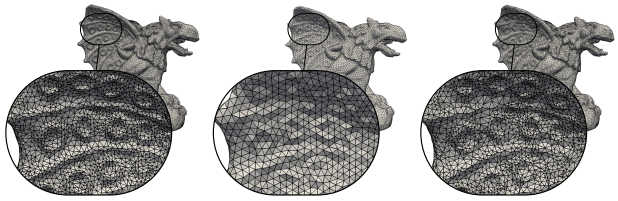


Figure 12: *Non-obtuse remeshing results for the Gargoyle model (left). While [AGY*16] (middle) caused important features to be smoothed out, ours (right) preserved features satisfactorily.*

time steps, significantly increasing the computational cost. Using well-spaced sample points, either random or structured, as Voronoi seeds to decompose the interior of a domain provides upper bounds on the aspect ratio of the resulting cells. However, these cells typically have problematic short edges, whenever two Voronoi seeds come too close to each other. We call an edge e_1 *short* if it belongs to a cell with a longer edge e_2 such that $|e_1|/|e_2| < 10\%$. This definition is particularly useful when the underlying domain exhibits rapid variations in the sizing function. Our strategy provides a principled approach to bound the minimum inter-seed distance and eliminate short edges using more constraints on sampling.

5.4.1. Geometric Constraints and Operators

We alternate between two sampling operators: relocation and ejection. We only update seeds whose cells contain short edges. We use relocation for the first pass before incorporating ejection. The reason behind this sequence is that relocation suffices in most cases while ejection helps expand the feasible regions where relocation failed. The geometric constraints are defined in terms of the dual Delaunay mesh to preserve: (1) Delaunay property for both neighboring elements and newly created elements, (2) Minimum Delaunay edge length and (3) Sizing function.

We bound the minimum Delaunay edge length by an exclusion sphere around each vertex. This prevents Voronoi seeds from getting too close to one another which improves the aspect ratio of Voronoi cells. In order to bias resampling to maximize the distance between new Delaunay vertices, we sample 10 vertices and pick the candidate that is farthest from all surrounding circumcircles.

5.4.2. Planar Results

The only work we know about that directly targets this problem was presented in Sieger et al. [SAB10], where short edges were defined by the ratio to the mean edge length and only edges in the range $1\% - 5\%$ were considered. In contrast, our definition of short edges has a better chance of producing longer short edges resulting in better elements. Unfortunately, due to lack of a readily available implementation or data set, we do not include a comparison against [SAB10].

We apply our algorithm on the 2D geometries in Figure 13, spanning different types of difficulties. Figure 1 shows an optimized Voronoi diagram where the sizing function is implicitly defined by a grayscale image. The process started by sampling the image and using the intensity of the sampled pixel to infer the sizing function. Using these samples as Voronoi seeds, the resulting tessellation had 14272 bad cells. The improved output has no bad cells while being visually similar to the input, as sizing was preserved.

6. Guarantees and Limitations

Our proposed strategy uses local resampling in order to globally optimize the input mesh with respect to the declared quality objectives. In practice, solving global optimization problem on meshes has a very high computational complexity [Epp01]. The common approach to obtain faster solvers to such instances is sampling [JPS93]. Consequently, we use constrained local resampling from feasible regions to make faster progress towards an optimal configuration. Thanks to the extra degrees of freedom provided by local resampling, our strategy terminates at local minima which are typically better than other deterministic alternatives [YW16, SAB10]. With these guarantees, our proposed strategy strikes the right balance between a principled approach with guarantees ensuring strict improvement and an efficient way to explore the solution space probabilistically by means of sampling.

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd.

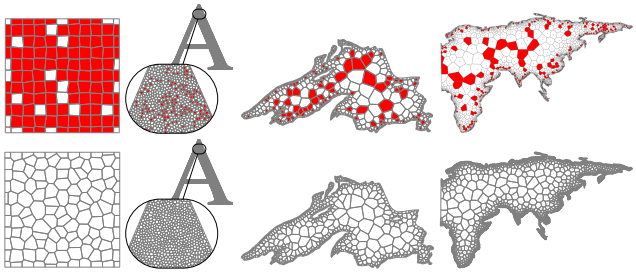


Figure 13: *Elimination of short edges in planar Voronoi meshes. From left to right: a jittered grid where most cells contain short edges, totaling 98 bad elements; a dense mesh with a constant sizing function and 1666 bad elements; two meshes with rapid changes in grading with 139 and 541 bad elements, respectively.*

We choose uniform sampling instead of deterministic rules (e.g., gradients) because such methods are likely to bias towards extreme configurations with less degrees of freedom, get stuck sooner and require costly evaluations. Unless this bias makes it easier to guarantee convergence into restricted families of meshes of higher quality, it is unclear how such regimes would be chosen over uniform sampling which is able to reach larger classes of meshes achieving very satisfactory results as shown in our experiments.

6.1. Guarantees

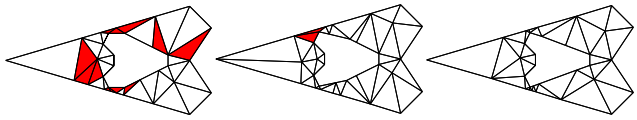
By construction, the algorithm in 3.1 is guaranteed to terminate without degrading quality; patches are only remeshed if quality is preserved with no degradation of neighbor quality. This is demonstrated in practice by Table 2, Table 3, and Table 5. By requiring strict improvement, say lexicographic minimum quality, we can also guarantee termination and no repeating scenarios. Moreover, a patch is never visited more than once unless its topology or geometry has changed towards improvement.

For curved surfaces, new samples are picked from the input triangulation. This guarantees an upper bound on the Hausdorff distance between the input and output meshes, depending on the resolution of the input mesh, as shown in Table 2, Table 3, and Table 5.

6.2. Limitations

The main limitation is the potential of getting stuck in local minima. For example, in Figure 14, we do not achieve a non-obtuse triangulation. Our algorithm handled most of the obtuse angles in the input mesh (a), except for two elements reaching a dead-end, as shown in (b), where no operator can improve the red patch. The mesh in this example allows little degrees of freedom as it is rather coarse. This dead-end, however, is rare in practice; visiting patches in a different order resolved this problem as in (c).

To determine the frequency of dead-ends, we test our non-obtuse remesher on 10^6 points in a unit box. The random seed changes the sequence of visited patches, and the resampled point locations. We use an overly strict criterion and count a run as a failure if a non-obtuse mesh was not obtained after three iterations. Only 5 runs out of 30 failed, and each had *only one* obtuse triangle.



(a) *Input.* **(b)** *Dead-end.* **(c)** *Success.*

Figure 14: *Limiting scenario for the non-obtuse triangulation of a mesh (a). A rare dead-end (b) might be reached, and can be resolved (c) by changing the order of visiting mesh elements.*

Hence, even though it is possible to get stuck, with such a high success rate, and thanks to the low overhead of the approach, rerunning the algorithm a few times quickly produces a handful of improved meshes to choose from.

7. Concluding Remarks

We introduced a versatile constrained resampling strategy for mesh improvement. We started by deriving the spatial representation of various quality objectives and developed a toolbox of local resampling operators that strictly improve or preserve quality. Our resampling approach levarages ideas from both smoothing and transformation methods and generalizes popular point insertion schemes like Delaunay refinement and off-centers. We demonstrated the successful application of our strategy to a number of important problems on both planar and surface meshes, where we were able to achieve multiple objectives simultaneously and outperform state-of-the-art. Our tests on a collection of models of varying complexity always achieved the required quality objectives. Failures are rare, but possible. We presented a basic empirical quantification of the failure rate, arguing that the speed of the proposed approach compensates for any occasional failures.

To extend this work, more applications can be considered, e.g., improving the angle bounds of unstructured Delaunay and Voronoi meshes. This application will likely require more sophisticated sequences of sampling operators. Another direction is to handle unstructured quadrilateral, fixed-topology and anisotropic meshes. Regarding failure rates, a richer set of operators may allow stuck patches to progress, possibly by resampling neighboring patches. Our preliminary results on acute remeshing are promising and a more comprehensive study of the power and limitations of the proposed approach for this challenging problem would be very exciting. Last but not least, our current implementation does not explicitly handle sharp features and may fail on challenging test cases. As the preservation of triangle quality is in conflict with the preservation of sharp features, it would be interesting to attempt a more robust implementation and compare the achievable improvements on more challenging models with what was reported here.

Acknowledgements

The authors would like to thank Dong-Ming Yan, Kaimo Hu [HYB*16] and Jianwei Guo [AGY*16] for providing valuable input and helping run some additional tests, and Darren Engwirda for providing the FD models [EI16].

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), Applied Mathematics Program. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[ABE99] AMENTA N., BERN M., EPPSTEIN D.: Optimal point placement for mesh smoothing. *Journal of Algorithms 30*, 2 (Feb. 1999), 302–322. doi:10.1006/jagm.1998.0984. 2

[AGY*16] AHMED A. G. M., GUO J., YAN D. M., FRANCESCHI J. Y., ZHANG X., DEUSSEN O.: A simple push-pull algorithm for blue-noise sampling. *IEEE Transactions on Visualization and Computer Graphics PP*, 99 (December 2016), 1–1. doi:10.1109/TVCG.2016.2641963. 2, 9, 10, 11

[AME14] ABDELKADER A., MITCHELL S. A., EBEIDA M. S.: Steiner point reduction in planar Delaunay meshes. In *Symposium on Computational Geometry (SoCG), Young Researchers Forum* (2014). available from http://www.computational-geometry.org/YRF/cgyrf2014.pdf 5

[ATC*08] AU O. K.-C., TAI C.-L., CHU H.-K., COHEN-OR D., LEE T.-Y.: Skeleton extraction by mesh contraction. *ACM Transactions on Graphics (TOG) 27*, 3 (Aug. 2008), 44:1–44:10. doi:10.1145/1360612.1360643. 2

[AUGA08] ALLIEZ P., UCELLI G., GOTSMAN C., ATTENE M.: *Recent Advances in Remeshing of Surfaces*. Springer, 2008, pp. 53–82. 1, 2

[BDK*03] BREWER M. L., DIACHIN L. F., KNUPP P. M., LEURENT T., MELANDER D. J.: *The Mesquite Mesh Quality Improvement Toolkit*. Springer Berlin Heidelberg, 2003, pp. 3–23. 2

[BHEK10] BARTOÅĹ M., HANNIEL I., ELBER G., KIM M.-S.: Precise hausdorff distance computation between polygonal meshes. *Computer Aided Geometric Design 27*, 8 (2010), 580 – 591. Advances in Applied Geometry. URL: http://www.sciencedirect.com/science/article/pii/S016783961000049X, doi:http://dx.doi.org/10.1016/j.cagd.2010.04.004. 5

[BKP*10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., LÉVY B.: *Polygon mesh processing*. AK Peters, October 2010. 1

[BO05] BOISSONNAT J.-D., OUDOT S.: Provably good sampling and meshing of surfaces. *Graphical Models 67*, 5 (2005), 405 – 451. Solid Modeling and Applications. URL: http://www.sciencedirect.com/science/article/pii/S1524070305000056, doi:http://dx.doi.org/10.1016/j.gmod.2005.01.004. 2

[BPK*07] BOTSCH M., PAULY M., KOBBELT L., ALLIEZ P., LÉVY B., BISCHOFF S., RÖSSL C.: Geometric modeling based on polygonal meshes. In *ACM SIGGRAPH 2007 Courses* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. URL: http://doi.acm.org/10.1145/1281500.1281640, doi:10.1145/1281500.1281640. 2

[CDS12] CHENG S.-W., DEY T. K., SHEWCHUK J.: *Delaunay Mesh Generation*, 1st ed. Chapman & Hall/CRC, 2012. 2

[CJW*09] CLINE D., JESCHKE S., WHITE K., RAZDAN A., WONKA P.: Dart throwing on surfaces. *Computer Graphics Forum 28*, 4 (June 2009), 1217–1226. 5

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174. URL: http://dx.doi.org/10.1111/1467-8659.00236, doi:10.1111/1467-8659.00236. 5, 7, 8, 10

[CY16] CHEN Y., YUE L.: A method for dynamic simplification of massive point cloud. In *2016 IEEE International Conference on Industrial Technology* (Mar. 2016), ICIT 2016, IEEE, pp. 1690–1693. doi:10.1109/ICIT.2016.7475017. 2

[DFG99] DU Q., FABER V., GUNZBURGER M.: Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review 41*, 4 (Dec. 1999), 637–676. doi:10.1137/S0036144599352836. 2

[DGJ03] DU Q., GUNZBURGER M. D., JU L.: Constrained centroidal Voronoi tessellations for surfaces. *SIAM Journal on Scientific Computing 24*, 5 (2003), 1488–1506. doi:10.1137/S1064827501391576. 2

[DW03] DU Q., WANG D.: Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *International Journal for Numerical Methods in Engineering 56*, 9 (Mar. 2003), 1355–1373. doi:10.1002/nme.616. 2

[EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W.: Multiresolution analysis of arbitrary meshes. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques* (1995), SIGGRAPH '95, ACM, pp. 173–182. 7

[EI16] ENGWIRDA D., IVERS D.: Off-centre Steiner points for delaunay-refinement on curved surfaces. *Computer-Aided Design 72* (2016), 157 – 171. 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation. URL: http://www.sciencedirect.com/science/article/pii/S0010448515001608, doi:http://dx.doi.org/10.1016/j.cad.2015.10.007. 6, 11

[EMA*13] EBEIDA M. S., MAHMOUD A. H., AWAD M. A., MOHAMMED M. A., MITCHELL S. A., RAND A., OWENS J. D.: Sifted disks. *Computer Graphics Forum 32*, 2 (May 2013), 509–518. URL: https://cfwebprod.sandia.gov/cfdocs/CCIM/docs/SiftedDisks_final.pdf, doi:10.1111/cgf.12071. 2

[EMP*12] EBEIDA M. S., MITCHELL S. A., PATNEY A., DAVIDSON A. A., OWENS J. D.: A simple algorithm for maximal Poisson-disk sampling in high dimensions. *Computer Graphics Forum 31*, 2 (May 2012), 785–794. URL: http://www.cs.sandia.gov/~samitch/papers/eurographics_mps-final-with-appendix.pdf, doi:10.1111/j.1467-8659.2012.03059.x. 3, 5

[Epp01] EPPSTEIN D.: Global optimization of mesh quality. *Tutorial at the 10th International Meshing Roundtable 10* (2001), 13. https://www.ics.uci.edu/~eppstein/pubs/Epp-IMR-01.pdf. 10

[ERA*16] EBEIDA M. S., RUSHDI A. A., AWAD M. A., MAHMOUD A. H., YAN D.-M., ENGLISH S. A., OWENS J. D., BAJAJ C. L., MITCHELL S. A.: Disk density tuning of a maximal random packing. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 259–269. 3

[EÜ09a] ERTEN H., ÜNGÖR A.: Computing triangulations without small and large angles. In *Sixth International Symposium on Voronoi Diagrams* (June 2009), ISVD '09, IEEE, pp. 192–201. 2

[EÜ09b] ERTEN H., ÜNGÖR A.: Quality triangulations with locally optimal Steiner points. *SIAM Journal on Scientific Computing 31*, 3 (2009), 2103–2130. 2, 5

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 97* (Aug. 1997), pp. 209–216. 2, 7, 8

[Guo17] GUO J.:. personal communication, April 2017. 9

[GYJZ15] GUO J., YAN D.-M., JIA X., ZHANG X.: Efficient maximal poisson-disk sampling and remeshing on surfaces. *Computers & Graphics 46* (2015), 72 – 79. 2

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), SIGGRAPH '93, ACM, pp. 19–26. doi:10.1145/166117.166119. 2

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd.

[HL88] HO-LE K.: Finite element mesh generation methods: a review and classification. *Computer-Aided Design 20*, 1 (Jan./Feb. 1988), 27–38. 1

[Hu17] HU K.:. personal communication, April 2017. 7

[HYB*16] HU K., YAN D. M., BOMMES D., ALLIEZ P., BENES B.: Error-bounded and feature preserving surface remeshing with minimal angle improvement. *IEEE Transactions on Visualization and Computer Graphics PP*, 99 (2016), 1–1. doi:10.1109/TVCG.2016.2632720. 2, 4, 7, 11

[JFL14] JIE Y.-X., FU X.-D., LIU Y.: Mesh generation for FEM based on centroidal Voronoi tessellations. *Mathematics and Computers in Simulation 97* (Mar. 2014), 68–79. doi:10.1016/j.matcom.2013.05.014. 2

[Joe89] JOE B.: Three-dimensional triangulations from local transformations. *SIAM Journal on Scientific and Statistical Computing 10*, 4 (1989), 718–741. 2

[JPS93] JONES D. R., PERTTUNEN C. D., STUCKMAN B. E.: Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications 79*, 1 (1993), 157–181. 10

[KCS98] KOBBELT L., CAMPAGNA S., SEIDEL H.-P.: A general framework for mesh decimation. In *Proceedings of the Graphics Interface 1998 Conference* (June 1998), vol. 98, pp. 43–50. URL: http://graphicsinterface.org/wp-content/uploads/gi1998-6.pdf. 2

[KS98] KIMMEL R., SETHIAN J. A.: Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences 95*, 15 (1998), 8431–8435. 7

[KS08] KLINGNER B. M., SHEWCHUK J. R.: Aggressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable* (Seattle, Washington, oct 2008), Springer, pp. 3–23. 2

[Lee99] LEE F.: *Principles of CAD/CAM/CAE systems*. Addison-Wesley Longman Publishing Co., Inc., 1999. 1

[LM15] LAVOUÉ G., MANTIUK R.: Quality assessment in computer graphics. In *Visual Signal Quality Assessment*. Springer, 2015, pp. 243–286. 1

[LT97] LOW K.-L., TAN T.-S.: Model simplification using vertex-clustering. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (Apr. 1997), I3D '97, pp. 75–82. doi:10.1145/253284.253310. 2

[LT98] LINDSTROM P., TURK G.: Fast and memory efficient polygonal simplification. In *Visualization 1998. Proceedings* (1998), pp. 279–286. 2

[LXFH15] LIU Y.-J., XU C.-X., FAN D., HE Y.: Efficient construction and simplification of Delaunay meshes. *Transactions on Graphics 34*, 6 (2015), 174. 7, 8

[MD03] MOENNING C., DODGSON N. A.: A new point cloud simplification algorithm. In *Proc. Int. Conf. on Visualization, Imaging and Image Processing* (2003), pp. 1027–1033. 2

[MREB12] MITCHELL S. A., RAND A., EBEIDA M. S., BAJAJ C.: Variable radii Poisson-disk sampling, extended version. In *Canadian Conference on Computational Geometry (CCCG)* (2012), pp. 1–9. long color online version. 2

[PGK02] PAULY M., GROSS M., KOBBELT L. P.: Efficient simplification of point-sampled surfaces. In *Proceedings of the Conference on Visualization '02* (Oct. 2002), VIS '02, pp. 163–170. URL: http://dl.acm.org/citation.cfm?id=602099.602123. 2

[Ren16] RENKA R. J.: Two simple methods for improving a triangle mesh surface. *Computer Graphics Forum 35*, 6 (2016), 46–58. URL: http://dx.doi.org/10.1111/cgf.12731, doi:10.1111/cgf.12731. 2

[RMM*16] RUSHDI A. A., MITCHELL S. A., MAHMOUD A. H., BAJAJ C. L., EBEIDA M. S.: All-quad meshing without cleanup. *CAD 85* (August 2016), 83–98. doi:10.1016/j.cad.2016.07.009. 1

[SAB10] SIEGER D., ALLIEZ P., BOTSCH M.: Optimizing Voronoi diagrams for polygonal finite element computations. In *International Meshing Roundtable*. Springer, 2010, pp. 335–350. 10

[She96] SHEWCHUK J. R.: Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator. In *Applied computational geometry towards geometric engineering*. Springer, 1996, pp. 203–222. 2, 5, 8

[She02] SHEWCHUK J. R.: Delaunay refinement algorithms for triangular mesh generation. *Computational geometry 22*, 1 (2002), 21–74. 2

[SW03] SCHAEFER S., WARREN J.: Adaptive vertex clustering using octrees. In *Proceedings of SIAM Geometric Design and Computing* (2003), pp. 491–500. 2

[SZL92] SCHROEDER W. J., ZARGE J. A., LORENSEN W. E.: Decimation of triangle meshes. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (July 1992), SIGGRAPH '92, pp. 65–70. doi:10.1145/133994.134010. 2

[TBG09] THAKUR A., BANERJEE A. G., GUPTA S. K.: A survey of CAD model simplification techniques for physics-based simulation applications. *Computer-Aided Design 41*, 2 (2009), 65–80. 1

[TLK09] TANG M., LEE M., KIM Y. J.: Interactive hausdorff distance computation for general polygonal models. In *ACM SIGGRAPH 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 74:1–74:9. URL: http://doi.acm.org/10.1145/1576246.1531380, doi:10.1145/1576246.1531380. 5

[TWAD09] TOURNOIS J., WORMSER C., ALLIEZ P., DESBRUN M.: Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph. 28*, 3 (July 2009), 75:1–75:9. doi:10.1145/1531326.1531381. 2

[ÜS02] ÜNGÖR A., SHEFFER A.: Pitching tents in space-time: Mesh generation for discontinuous galerkin method. *International Journal of Foundations of Computer Science 13*, 02 (2002), 201–221. 7

[WHWB16] WANG L., HÉTROY-WHEELER F., BOYER E.: A hierarchical approach for regular centroidal Voronoi tessellations. *Computer Graphics Forum 35*, 1 (Feb. 2016), 152–165. doi:10.1111/cgf.12716. 2

[WK03] WU J., KOBBELT L.: A stream algorithm for the decimation of massive meshes. In *Graphics interface* (2003), vol. 3, pp. 185–192. 2

[YW12] YAN D.-M., WONKA P.: Adaptive maximal Poisson-disk sampling on surfaces. In *SIGGRAPH Asia 2012 Technical Briefs* (2012), SA '12, pp. 21:1–21:4. 2

[YW16] YAN D.-M., WONKA P.: Non-obtuse remeshing with centroidal Voronoi tessellation. *IEEE Transactions on Visualization and Computer Graphics 22*, 9 (Sept. 2016), 2136–2144. doi:10.1109/TVCG.2015.2505279. 6, 8, 10

© 2017 The Author(s)
Computer Graphics Forum © 2017 The Eurographics Association and John Wiley & Sons Ltd.