A Constrained Resampling Strategy for Mesh Improvement

https://escholarship.org/uc/item/5347s75h

Computer Graphics Forum, 36(5)

1467-8659

Abdelkader, Ahmed
Mahmoud, Ahmed H.
Rushdi, Ahmad A.
et al.

2017-07-01

10.1111/cgf.13256

https://github.com/Ahdhn/MeshImp

Peer reviewed
A Constrained Resampling Strategy for Mesh Improvement

Ahmed Abdelkader\textsuperscript{1*}, Ahmed H. Mahmoud\textsuperscript{2*}, Ahmad A. Rushdi\textsuperscript{2}, Scott A. Mitchell\textsuperscript{3}, John D. Owens\textsuperscript{2}, and Mohamed S. Ebeida\textsuperscript{3†}

\textsuperscript{1}University of Maryland, College Park \quad \textsuperscript{2}University of California, Davis, CA \quad \textsuperscript{3}Sandia National Laboratories, Albuquerque, NM

\textbf{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.

\section{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\textsuperscript{*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].

**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, KS80]. 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 [EU09b, EU09a]. 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. Maximal 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 $\Omega$ is the hole created in the mesh by removing some elements (e.g., vertices and associated faces) from a chosen patch.

### 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 $\Omega$.
2. Map each quality objective into a set of spatial constraints $\mathcal{C}$ on resampling over $\Omega$ (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 $\Omega$.
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 [EMP12], where a grid-based refinement was employed to track $\mathcal{F}$ up to machine precision or consider it empty.

![Figure 2](image1.png)

**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 $p_1$ to two fixed samples $p_1$ and $p_3$. 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.

![Figure 3](image2.png)

**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 [ERA16]. 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, if 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.

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 $\theta$ 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 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 $F$. For planar surfaces, we use the Simple MPS approach [EMP’12]. We construct an implicit background grid of quads enclosing $F$, determined by the bounding vertices and spatial constraints. We choose a coarse initial grid ($4 \times 4$ 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 the next grid cell. Otherwise, $p$ is rejected. A number of failed attempts proportional to the number of grid cells, we refine every cell to its vertices. Subcells completely outside $F$ 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 $F$ 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 $\alpha$ is the percentage decrease in the number of vertices.

5.1.1. Geometric Constraints and Operators

Delaunay sifting only uses the injection 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 silt (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. The meshes in Figure 5 were generated by Triangle for a minimum angle bound of $35^\circ$. Being the smallest angle bound for which Triangle guarantees termination, 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 achieve $\alpha \geq 72\%$ across different inputs while preserving all quality metrics including triangle quality $Q$ [YW16] ($Q = \frac{S}{\sqrt{A_P}}$, 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.

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] ($Q = \frac{S}{\sqrt{A_P}}$, where $S$ is the area, $h$ is the longest edge length and $P$ is half the perimeter).

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 $\theta^d$. To demonstrate how changing $\theta^d$ results in different resolutions, we gradually decrease $\theta^d$ 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 $\theta^d$ as a parameter to control the trade-off between the number of vertices and the approximation error; low $\theta^d$ achieves lower resolutions while tolerating larger deviations in terms of the Hausdorff distance.

![Figure 5: Delaunay sifting of planar meshes generated by Triangle with $\theta_{\text{min}} \geq 35^\circ$. These models feature high curvature, narrow regions and sharp corners. Sifting boundaries increases $\alpha$.](image)

![Figure 6: Delaunay sifting of planar meshes generated by aCute (sifting interiors and boundaries).](image)

![Figure 7: Delaunay sifting of surface meshes from different sources along with running times in seconds. $\{\theta_{\text{min}}, \theta_{\text{max}}, Q_{\text{min}}\}$ indicate the minimum angle, maximum angle and minimum triangle quality.](image)
Table 1: Delaunay sifting of planar meshes: Quality metrics before and after sifting for meshes generated by Triangle and aCute, including the number of vertices \( v \), number of triangles \( \Delta \), intrinsic Delaunay \( \theta_{\text{intra}} \) and restricted Delaunay \( \theta_{\text{max}} \) properties, in contrast to our algo-

<table>
<thead>
<tr>
<th>Method</th>
<th>Model</th>
<th>Input Mesh</th>
<th>Sifting Interior</th>
<th>Sifting Interior &amp; Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( N )</td>
<td>( \Delta )</td>
<td>( \theta_{\text{intra}} )</td>
<td>( \theta_{\text{max}} )</td>
</tr>
<tr>
<td></td>
<td>( v )</td>
<td>( \Delta )</td>
<td>( \theta_{\text{intra}} )</td>
<td>( \theta_{\text{max}} )</td>
</tr>
<tr>
<td>Triangle</td>
<td>Dolphin</td>
<td>3409</td>
<td>6482</td>
<td>35°</td>
</tr>
<tr>
<td></td>
<td>Lake</td>
<td>2056</td>
<td>3587</td>
<td>35°</td>
</tr>
<tr>
<td></td>
<td>Wavy</td>
<td>1164</td>
<td>2041</td>
<td>34°</td>
</tr>
<tr>
<td></td>
<td>Batman</td>
<td>859</td>
<td>1505</td>
<td>35°</td>
</tr>
<tr>
<td></td>
<td>Airfoil</td>
<td>839</td>
<td>1483</td>
<td>35°</td>
</tr>
<tr>
<td></td>
<td>Spiky</td>
<td>715</td>
<td>1207</td>
<td>35°</td>
</tr>
<tr>
<td>aCute</td>
<td>Dolphin</td>
<td>1125</td>
<td>1939</td>
<td>40°</td>
</tr>
<tr>
<td></td>
<td>Batman</td>
<td>580</td>
<td>948</td>
<td>40°</td>
</tr>
<tr>
<td></td>
<td>Spiky</td>
<td>552</td>
<td>881</td>
<td>40°</td>
</tr>
<tr>
<td></td>
<td>Face</td>
<td>443</td>
<td>715</td>
<td>40°</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Model</th>
<th>Input Mesh</th>
<th>Output Mesh</th>
<th>%</th>
<th>Input Mesh</th>
<th>Output Mesh</th>
<th>%</th>
<th>RMS ( d_{\text{H}} \times 10^{-3} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform MPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delaunay Refinement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frontal Delaunay</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2.3. 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 \( \theta^2 \). 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 algo-

The closest work to ours which achieves multiple objectives is the recent work by K. Hu et al. [HYB16]. 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 [HYB16] 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 [EDD98, KS98] and scientific computing [US02]. We show how our resampling strategy can be applied to obtain a
Mesh simplification: Comparison against the Quadratic Error Metric (QEM) [GH97] and Delaunay mesh simplification (DM) [LXFH15] across the addition of bounding angles by 90°. We use the same geometric constraints as in Delaunay sifting, with non-obtuse triangulation starting from given mesh of some model. To further demonstrate the potential of our resampling strategy, we present some preliminary results.

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, more aggressively, repeller injection. The algorithm terminates when all obtuse triangles are eliminated.

![Image](https://www.example.com/image.png)

Figure 8: Tuning the upper bound on dihedral angles θ⁴ and the corresponding reduction ratio for an MPS mesh on a sphere model (top left).

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 Δ, angle bounds (θₘᵢₙ, θₘₐₓ), triangle quality Qₘᵢₙ, dihedral angle bounds (θₐₘᵢₙ, θₐₘₐₓ), root mean square distance dₙₑᵦᵦ and Hausdorff distance dₜ (estimated by Metro [CRS98] and normalized by the diameter of the bounding box). The best result for each measure is shown in bold face.

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>v</th>
<th>Δ</th>
<th>θₘᵢₙ</th>
<th>θₘₐₓ</th>
<th>Qₘᵢₙ</th>
<th>θₐₘᵢₙ</th>
<th>θₐₘₐₓ</th>
<th>dₑᵦₑ(×10⁻²)</th>
<th>dₜ(×10⁻²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bunny (MPS)</td>
<td>DM</td>
<td>11.5K</td>
<td>153</td>
<td>25k</td>
<td>302</td>
<td>30</td>
<td>116</td>
<td>171</td>
<td>0.06</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>QEM</td>
<td>8.5K</td>
<td>300</td>
<td>17k</td>
<td>790</td>
<td>40</td>
<td>116</td>
<td>165</td>
<td>0.5</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>166</td>
<td>0.5</td>
<td>111</td>
<td>165</td>
<td>0.1</td>
<td>79</td>
</tr>
<tr>
<td>Fertility (MPS)</td>
<td>DM</td>
<td>4.5</td>
<td>155</td>
<td>0.12</td>
<td>6.5</td>
<td>0.5</td>
<td>116</td>
<td>165</td>
<td>0.08</td>
<td>116</td>
</tr>
<tr>
<td></td>
<td>QEM</td>
<td>8.5K</td>
<td>300</td>
<td>17k</td>
<td>790</td>
<td>40</td>
<td>116</td>
<td>165</td>
<td>0.5</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>166</td>
<td>0.5</td>
<td>111</td>
<td>165</td>
<td>0.1</td>
<td>79</td>
</tr>
<tr>
<td>Loop (MPS)</td>
<td>DM</td>
<td>10.7K</td>
<td>1.4K</td>
<td>22k</td>
<td>3K</td>
<td>30</td>
<td>117</td>
<td>160</td>
<td>0.48</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>QEM</td>
<td>10.7K</td>
<td>1.4K</td>
<td>22k</td>
<td>3K</td>
<td>30</td>
<td>117</td>
<td>160</td>
<td>0.48</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>166</td>
<td>0.5</td>
<td>111</td>
<td>165</td>
<td>0.08</td>
<td>116</td>
</tr>
<tr>
<td>Biniba (DR)</td>
<td>DM</td>
<td>25.4K</td>
<td>180</td>
<td>51k</td>
<td>350</td>
<td>28</td>
<td>6</td>
<td>122</td>
<td>0.44</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>QEM</td>
<td>25.4K</td>
<td>180</td>
<td>51k</td>
<td>350</td>
<td>28</td>
<td>6</td>
<td>122</td>
<td>0.44</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>166</td>
<td>0.5</td>
<td>111</td>
<td>165</td>
<td>0.08</td>
<td>116</td>
</tr>
<tr>
<td>Rocker (DR)</td>
<td>DM</td>
<td>10.8K</td>
<td>240</td>
<td>21k</td>
<td>485</td>
<td>30</td>
<td>13</td>
<td>155</td>
<td>0.31</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>QEM</td>
<td>10.8K</td>
<td>240</td>
<td>21k</td>
<td>485</td>
<td>30</td>
<td>13</td>
<td>155</td>
<td>0.31</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>166</td>
<td>0.5</td>
<td>111</td>
<td>165</td>
<td>0.08</td>
<td>116</td>
</tr>
<tr>
<td>Biniba (FD)</td>
<td>DM</td>
<td>24.4K</td>
<td>270</td>
<td>49k</td>
<td>535</td>
<td>28</td>
<td>7</td>
<td>121</td>
<td>0.46</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>QEM</td>
<td>24.4K</td>
<td>270</td>
<td>49k</td>
<td>535</td>
<td>28</td>
<td>7</td>
<td>121</td>
<td>0.46</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>166</td>
<td>0.5</td>
<td>111</td>
<td>165</td>
<td>0.08</td>
<td>116</td>
</tr>
<tr>
<td>Rocker (FD)</td>
<td>DM</td>
<td>10.2K</td>
<td>260</td>
<td>20.5k</td>
<td>520</td>
<td>32</td>
<td>9</td>
<td>130</td>
<td>0.24</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>QEM</td>
<td>10.2K</td>
<td>260</td>
<td>20.5k</td>
<td>520</td>
<td>32</td>
<td>9</td>
<td>130</td>
<td>0.24</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>166</td>
<td>0.5</td>
<td>111</td>
<td>165</td>
<td>0.08</td>
<td>116</td>
</tr>
<tr>
<td>Chinese Dragon (CVT)</td>
<td>DM</td>
<td>30K</td>
<td>1.5K</td>
<td>60k</td>
<td>3.1K</td>
<td>34</td>
<td>3</td>
<td>103</td>
<td>0.04</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>QEM</td>
<td>30K</td>
<td>1.5K</td>
<td>60k</td>
<td>3.1K</td>
<td>34</td>
<td>3</td>
<td>103</td>
<td>0.04</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>166</td>
<td>0.5</td>
<td>111</td>
<td>165</td>
<td>0.08</td>
<td>116</td>
</tr>
<tr>
<td>David Head (CVT)</td>
<td>DM</td>
<td>15K</td>
<td>660</td>
<td>30k</td>
<td>1.3K</td>
<td>33</td>
<td>7</td>
<td>107</td>
<td>0.56</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>QEM</td>
<td>15K</td>
<td>660</td>
<td>30k</td>
<td>1.3K</td>
<td>33</td>
<td>7</td>
<td>107</td>
<td>0.56</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>166</td>
<td>0.5</td>
<td>111</td>
<td>165</td>
<td>0.08</td>
<td>116</td>
</tr>
<tr>
<td>Omotondo (CVT)</td>
<td>DM</td>
<td>20K</td>
<td>260</td>
<td>40k</td>
<td>530</td>
<td>28</td>
<td>7</td>
<td>140</td>
<td>0.3</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>QEM</td>
<td>20K</td>
<td>260</td>
<td>40k</td>
<td>530</td>
<td>28</td>
<td>7</td>
<td>140</td>
<td>0.3</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>166</td>
<td>0.5</td>
<td>111</td>
<td>165</td>
<td>0.08</td>
<td>116</td>
</tr>
</tbody>
</table>

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°, 35°]. 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...
Figure 9: Mesh simplification of surface meshes from different sources. The minimum dihedral angle $\theta_d$ and the reduction ratio $\alpha$ 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).

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 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 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].

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 $90^\circ$. We first re-run all the models listed in Table 5 with maximum angle bound of $85^\circ$. 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 $80^\circ$, our algorithm was not able to converge with any model; with marginal improvement between different iterations.
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 [JPS03]. Consequently, we use constrained local resampling from feasible regions to make faster progress towards an optimal configuration. Thanks to the extra degrees of freedom pro-
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.

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 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.

Hence, even though it is possible to get stuck, with such a high success rate, and thanks to the low overhead of the approach, running 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 leverages 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 [E116].

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


[BKP’10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., LÉVY B.: Polygon mesh processing. AK Peters, October 2010. 2


[EUO9a] ERTEN H., ÜNGAR O.: Computing triangulations without small and large angles. In Sixth International Symposium on Voronoi Diagrams (June 2009), ISVD ’09, IEEE, pp. 192–201. 2


A. Abdelkader et al. / A Constrained Resampling Strategy for Mesh Improvement


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


