Steiner Point Reduction in Planar Delaunay Meshes

Ahmed Abdelkader* Scott A. Mitchell† Mohamed S. Ebeida†

Abstract

We develop a mesh simplification strategy that preserves angle bounds. Preliminary results for a sample of planar Delaunay meshes are then presented. We demonstrate significant improvements of Triangle output, in terms of the number of Steiner points needed for a required angle bound, specially for large bounds where Triangle is known to possibly perform poorly.

1 Introduction

Given an input Delaunay mesh, one goal of many mesh improvement algorithms is to reduce the number of points while preserving angle bounds, e.g., [1]. In this paper, we develop a sampling-based technique that achieves this goal by replacing a pair of neighboring points with one point. We introduce a set of constraints for the location of the new point based on the desired minimum angle and compute an explicit representation of the solution region, which we then sample from to find the replacement point, as shown in Figure 1. This strategy generalizes edge collapse, as it possibly combines edge swaps to update the mesh after replacement.

The simplification concept we adopt is called *sifting* and was first presented in [2]. While in this paper we deal with arbitrary Delaunay meshes, e.g., output of Delaunay Refinement (DR) [3], the study in [2] focused mainly on *explicit sizing functions* and used constraints based on separation of points and maximal coverage of the domain. Thanks to the angle bounds, the number of constraints is bounded by a constant and all updates are local. With the sampling region defined, a replacement sample can be found in constant time [4]. In the same spirit, we call our method *Delaunay Sifting* (DS).

We develop the sift algorithm in Section 2. Then, in Section 3, we show improvement of several Delaunay meshes generated by Triangle [3]. Finally, we conclude and highlight ongoing work in Section 4.

2 The Delaunay Sifting Algorithm

We define the sampling region as the set of valid replacement points that preserve the minimum angle constraints. We denote the minimum angle by α.

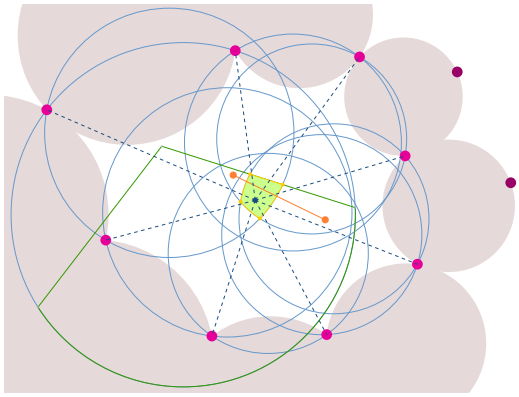


Figure 1: Sifting example (orange: candidate, purple: neighbors, green: sampling region, blue: replacement point, dashed lines: retriangulation, other: constraints).

Given an edge, removing all edges incident on its two end points creates a *gap* in the mesh. The gap is a polygonal region that can be partitioned into triangles by connecting all vertices on the boundary to the replacement of the removed pair of points. Shrinking the gap, by creating *ear* triangles when possible, can also help make sifting possible. Below, we list the constraints that limit the location of valid replacement points.

2.1 Constraint(1) - Neighboring Circumcircles

Neighboring triangular faces surrounding the gap each gives rise to a circumcircle. By definition of the Delaunay triangulation, such circumcircles should be empty.

2.2 Constraint(2) - No Thin Triangles

If we think of any given gap edge as the base of a new triangle and the sample as its apex, then we need to enforce the lower bound for both the apex angle and base angles. For the apex angle, we can simply require the sample point to fall within a circle having the edge as a chord with a central angle of 2α. As for the base angles, the sample point has to fall in the intersection of two half-planes, each making an angle of α with the base.

2.3 Constraint(3) - Boundary Segments

If the candidate edge is incident on a boundary segment, sifting must ensure the boundary face remains the same.

*University of Maryland, College Park, akader@cs.umd.edu
†Sandia National Laboratories, : msebeid@sandia.gov

If only one end of the edge is a boundary vertex, it *cannot* be a corner on the boundary. If the edge as a whole is a boundary segment, it must be on a larger segment with boundary neighbors on both sides. In both cases, the new sample will be constrained to lie on the larger boundary segment.

2.4 Putting It All Together

Applying all constraints simultaneously to all gap edges, we compute the desired sampling region, as in Figure 1. We further classify these constraints into two types: inclusive and exclusive constraints. From 2.1, we get a set of exclusive circles and from 2.2 and 2.3, we get two inclusive sets of arc-gons and possibly a line segment. This reduces to a set of boolean conditions that can be easily checked to test any candidate replacement point.

3 Implementation and Results

We opt for a maximal random sifting strategy and apply the DS operation in a uniformly random manner. At each iteration, we generate a random permutation of edges and attempt to sift them in that order. Whenever sifting is successful, we start a new iteration. Otherwise, no more sifting is possible. Other execution strategies can be considered as future work, e.g., smallest-angles first. Random order allows us to explore different evolution paths in our experiments, as we seek the minimum possible number of points, and it also has proven advantages, from an efficiency standpoint. In addition, choosing replacement points randomly preserves desirable spectral properties of the mesh, e.g., blue noise.

In Figure 2, we show a sample of results for our experiments with planar meshes. For each model, we show the input domain, the output of `triangle -q35`[1] and the sifted mesh. For each mesh, we indicate the number of sample points and the sifting ratio achieved. The sifting ratio is defined as the ratio between the reduced number of points and the original number of points. Table 1 shows a summary of more settings.

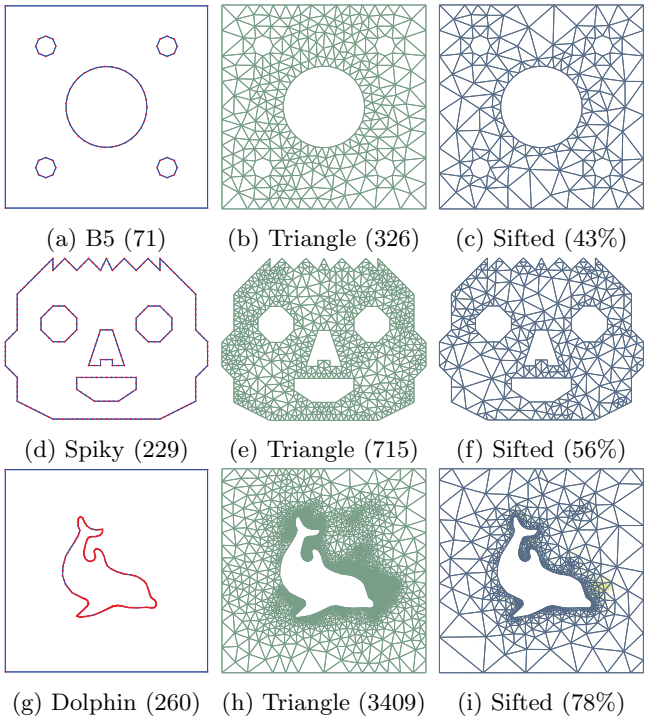
4 Conclusions and Future Work

Our current implementation can perform a few thousand DS attempts per second. The total number of attempts for a given mesh was observed to be linear in the number of edges. Equivalently, it is linear in the number of vertices when taking into account the minimum angle constraints, which bound the degree of vertex connectivity. A slight decrease in the median of angles was also observed. Currently, we are working on extensions to curved surfaces and the derivation of analytical models of both improvement and convergence.

[1]Generate a Delaunay mesh with angles no less than 35 °.

./triangle -q		20		30		35	
`B5`	71	139	17%	197	29%	326	43%
`Spiky`	229	330	54%	505	59%	715	56%
`Dolphin`	260	471	31%	865	49%	3409	78%

Table 1: Size of Triangle outputs and achievable sifting.



(a) B5 (71) (b) Triangle (326) (c) Sifted (43%)
(d) Spiky (229) (e) Triangle (715) (f) Sifted (56%)
(g) Dolphin (260) (h) Triangle (3409) (i) Sifted (78%)

Figure 2: Model, triangle -q35 and sifted meshes.

References

[1] Alper Üngör. Off-centers: A new type of steiner points for computing size-optimal quality-guaranteed delaunay triangulations. In *LATIN 2004: Theoretical Informatics*, pages 152–161. Springer, 2004.

[2] Mohamed S Ebeida, Ahmed H Mahmoud, Muhammad A Awad, Mohammed A Mohammed, Scott A Mitchell, Alexander Rand, and John D Owens. Sifted disks. In *Computer Graphics Forum*, volume 32, pages 509–518. Wiley Online Library, 2013.

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

[4] Mohamed S Ebeida, Andrew A Davidson, Anjul Patney, Patrick M Knupp, Scott A Mitchell, and John D Owens. Efficient maximal poisson-disk sampling. In *ACM Transactions on Graphics (TOG)*, volume 30, page 49. ACM, 2011.