Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2014) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

23rd International Meshing Roundtable (IMR23)

Delaunay quadrangulation by two-coloring vertices

Scott A. Mitchell[a,*], Mohammed A. Mohammed[b], Ahmed H. Mahmoud[b], Mohamed S. Ebeida[a]

[a]*Sandia National Laboratories, P.O. Box 5800 MS 1320, Albuquerque 87185, U.S.A.*

[b]*Alexandria University, El-Guish Road, El-Shatby, Alexandria 21526, Egypt*

Abstract

We introduce a *bichromatic Delaunay quadrangulation* principle by assigning the vertices of a Delaunay triangulation one of two colors, then discarding edges between vertices of the same color. We present algorithms for generating quadrangulations using this principle and simple refinements. The *global* vertex coloring ensures that only *local* refinements are needed to get all quads. This is in contrast to triangle-pairing algorithms, which get stuck with isolated triangles that require global refinement. We present two new sphere-packing algorithms for generating the colored triangulation, and we may also take as input a Delaunay refinement mesh and color it arbitrarily. These mesh non-convex planar domains with provable quality: quad angles in $[10°, 174°]$ and edges in $[0.1, 2]r$. The algorithms extend to curved surfaces and graded meshes. The "random" algorithm generates points with blue noise. The "advancing-front" algorithm produces large patches of boundary-aligned square tilings.

© 2014 The Authors. Published by Elsevier Ltd.
Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

Keywords: Delaunay; quadrangulation; graph coloring; disk packing; blue noise; random; advancing front

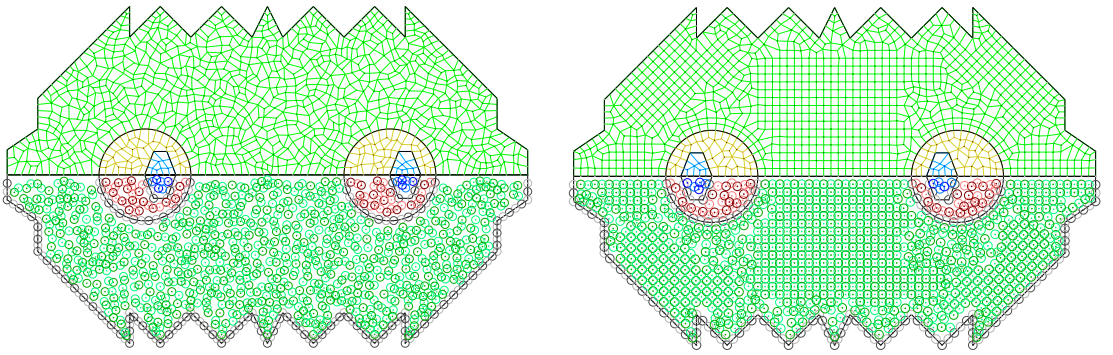


Fig. 1: Delaunay quadrangulation (top) from two-color disks (bottom light and dark circles). Left: random. Right: advancing front.

*Corresponding author. Tel.: +1-505-845-7594.
E-mail address: samitch@sandia.gov

1877-7058 © 2014 The Authors. Published by Elsevier Ltd.
Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

1. Introduction

This paper describes a novel quad meshing criterion and algorithm based on combining three well-established ideas from different communities: advancing front, sphere packing, and graph coloring. In this introduction we outline the algorithm for experts and practitioners, then briefly review these three key ideas. It is not feasible to fully review each idea and some of the algorithms we use as subroutines such as disk packing; knowledge of these may be obtained by reading the selected references. Further, the mesh-quality proofs span about 10 pages, and these proceedings do not admit appendices, so the proofs are available separately online as a Sandia tech report [1].

1.1. Algorithm summary

We generate a well-spaced point set with a good-quality Delaunay triangulation using one of our new disk-packing algorithms, or by traditional Delaunay refinement. All points are assigned one of two colors; dark and light in each region of Fig. 1. Our new algorithms surround each point by two disks, a large one that prevents same-colored points, and a small one that prevents opposite-colored points. Our first algorithm variant, "random," places points and colors randomly. The second algorithm, "advancing front," places points and colors in structured layers, then fills gaps with the random algorithm. Their outputs are contrasted in Fig. 1.

We form the Delaunay triangulation of the points, discard monochromatic edges, then perform local refinement to generate all-quads, as follows. We retain Delaunay edges connecting opposite-colored points, and discard those between same-colored points. The retained Delaunay edges produce even-sided polygons. Any polygon with six (or more) sides contains one (or more) discarded monochromatic Delaunay triangles. By switching colors, the fraction of monochromatic triangles is reduced to about 2%. We add each monochromatic triangle incenter (not circumcenter) as an opposite-colored point and deterministically connect it to its triangle's vertices; this is guaranteed to convert many-sided polygons into quads. Quads with large angles are refined by a local one-to-five template.

1.2. Idea one, advancing front quad meshing

There has been an increasing demand to generate quadrilateral meshes for computational simulations and graphics [2]. In contrast to triangular mesh generation, most of the quadrilateral meshing algorithms are heuristic and may result in a mesh with poor quality. The triangle meshing literature is dominated by algorithms based on Delaunay triangulation, advancing front, and grid-based methods. Provably good triangular mesh generation methods are well developed for planar and curved surfaces. However, quadrilateral elements are sometimes preferred in finite element applications due to their superior performance.

Unstructured quadrilateral meshes are typically generated by one of two approaches: direct or indirect [3]. An indirect algorithm starts with a triangular or quad-dominant mesh and then converts all the triangles into quads [4–6]. A direct approach produces quads without any intermediate form of triangulation. The main direct approaches are domain decomposition, grid-based, and advancing front.

Advancing front techniques start with placing nodes on the domain boundaries. These represent the "initial front." Quadrilateral elements are then formed by projecting each edge on the front towards the interior. A new front is formed using the edges on the new boundary of the quadrilateral mesh. This process is repeated recursively until the domain is completely covered with quadrilaterals. Zhu [7] is among the first to propose a quadrilateral advancing front algorithm. In his approach, two triangles are created using the traditional advancing front methods then combined to form a single quadrilateral. Blacker and Stephenson [8] introduced the Paving algorithm in which they place a complete row of quads between two sharp angles. White and Kinney [9] enhanced the robustness of the paving algorithm through creating individual quadrilaterals rather than a complete row. These methods generate near-boundary elements with high quality. However, it is difficult to robustly close the gap between colliding fronts with high quality elements, especially if two overlapping elements have a large difference in size. These cases are resolved by heuristics that sometimes result in elements with poor quality. Moreover, the detection and resolution of the closure regions can be

very time consuming and sensitive to floating point errors. Bern and Eppstein [10] used an advancing-front circle packing to subdivide the domain directly into quadrilaterals with angles less than or equal to 120°, but with no guarantees on edge lengths or the smallest angle. It is advancing front in the sense that new disks are placed tangent to prior disks.

1.3. Idea two, random disk packings

Disk packing methods fill a domain with disks that are forbidden to overlap in some way. The packing is *maximal* when there is no room to add another disk without creating a forbidden conflict. The maximality ensures that no domain point is farther from a sample than the disk radius, else we could insert a disk center there. In a maximal packing where the disk radius is constant, or varies slowly, the points are *well-spaced*. This means that there is a maximum bound on the domain-point to nearest disk-center distance, a minimum bound on the distance between nearby disk-centers, and a maximum bound on the ratio of the two; further these all hold locally. Well-spaced points have Delaunay triangulations with good angles [11–13]. Traditional Delaunay refinement algorithms [12] flip the cause and effect, using the angles of an intermediate Delaunay triangulation to guide the generation of a well-spaced point set. Maximal Poisson-disk Sampling (MPS) is defined in terms of a process that sequentially creates disks and keeps a disk only if it is outside the prior disks. In the definition, the center of each disk is chosen uniformly at random, but many algorithms change the placement process slightly for efficiency. The important thing for graphics applications is that the final distribution of disk centers exhibits "blue noise," which is loosely used to mean a distribution whose Fourier spectrum has little low frequency content; see Section 4.

There are many algorithms for maximal and near-maximal Poisson-disk samplings from graphics [14–16]. Multi-class blue noise sampling [17] produces random samplings of multiple categories of disks interspersed. Each category is well-spaced by itself, and the combined set is also well-spaced. Each point has disks of multiple radii surrounding it, one for each category, for spacing the different categories different amounts apart. This paper is a special case where we have only two categories of points, and each has the same two radii. Our random placement algorithm is similar to multi-class sampling. Mitchell et al. [18] introduced a single category of disks with two radii; the effect of the radii ratio α on the point spacing and triangle quality is about the same as in our setting with two categories.

Simple MPS [16] is perhaps the simplest and fastest MPS algorithm. It uses a quadtree where all active leaves have the same level to avoid bookkeeping. One nice thing about this approach is that it is fairly agnostic about the primitives used for deciding whether to discard a quad tree square, refine it, or accept a sample point in it. This paper uses the Simple MPS framework, with modified disk conflict and cell discard checks for the two categories and radii.

1.4. Idea three, two-coloring vertices for transforming triangles to quads

The famous "two-color theorem" shows that the edges of any quadrangulation are a bichromatic graph [19]. That is, it is possible to color the vertices with two colors, such that no edge connects two vertices of the same color.[1] This inspires our use of two colors (categories) of disks. A Delaunay triangulation connects two vertices if there is a circle through them that contains no other vertex; such vertices are nearby compared to the other vertices in about the same direction. Our two radii encourage disks of different colors to be close enough to one another to be connected by a Delaunay edge, and same-colored disks to be far apart and unconnected. Further, we enforce this by discarding same-colored Delaunay edges. To our knowledge, we are the first to define this bichromatic Delaunay criteria. [2]

[1] Quadrangulations are two-colorable for topologically "planar" surfaces, including meshes on geometrically curved surfaces that can be deformed into the plane, such as a sphere mesh by picking one quadrilateral to contain the point at infinity. For surfaces with topological handles, such as a torus, some quadrangulations are bichromatic, some not, depending on the existence of edge cycles with odd length. Having a quad mesh that is the boundary of a hex mesh does not settle the ambiguity, as it may have odd edge cycles that are null-homotopic in the complement of the hex mesh, e.g. for a torus, cycles must be even around the minor circumference but may be odd around the major circumference [20].

[2] A different use of the Delaunay criteria for finding quadrangulations is to lift points to three-dimensions, form a Delaunay tetrahedralization, then project back to the plane and make connectivity decisions based on the edge crossings [6].

One family of quad algorithms takes a triangulation and pairs triangles together to form quads [2,21]. Global matching algorithms tend to be expensive, e.g. quadratic. A bigger challenge is that there are often isolated unmatched triangles. These arise either geometrically or topologically: pairing would create a bad-quality quad, or a perfect matching over the whole mesh is impossible because of the graph topology. One way to resolve isolated triangles is to refine them into (e.g. three) quads, but this necessitates refining at least one adjacent quad, and its opposite-side neighbor, in a cascading chain that terminates at another unmatched triangle or the boundary [22]. That is, it requires global refinement to get all quads. This is in sharp contrast to our method with colored vertices, where refinement into all-quads can always be done locally without propagation. Triangle pairing is a local operation that leaves global difficulties, and two-coloring is a global operation that leaves only local difficulties.

2. Colored disk generation algorithms

We now describe our two new algorithms for generating disks: random, using Poisson-disk sampling; and advancing-front (or "biased") as in paving. We explain the basic steps here and Section 3.2 explores some heuristic modifications. In both algorithms we try to reach a maximal packing. Both algorithms are based on two colors of disks, which we will call red and blue. Each disk has two radii, and their ratio is a key algorithm parameter affecting the output quality, which we describe first.

2.1. Radii ratio α

Each disk is associated with two radii: r_s (small) and r_b (big). Disk centers of different colors are at least r_s apart, and same-colored disk centers are at least r_b apart. The parameter $\alpha = r_b/r_s$ controls the intermingling between the colors. We keep $\alpha \in [1, 3]$ and, empirically, $\alpha \approx 2.5$ works best. A value of 1 makes both radii the same. Non-colored, single-radius ($\alpha = 1$) maximal packings from other sources, such as Delaunay refinement, can be converted into a two-coloring by any coloring of the vertices.

2.2. Random Poisson-disk sampling algorithm

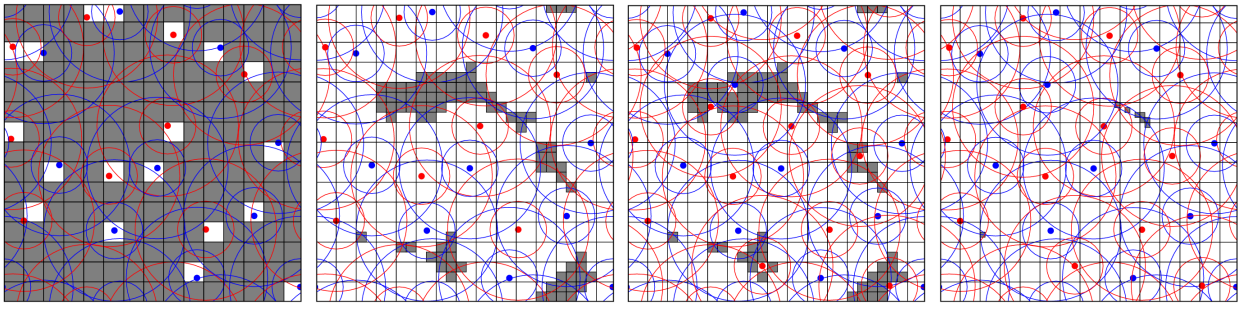
The random algorithm is based on the framework of Simple MPS [16], with modified primitives for our two-radii and bichromatic disks; see Fig. 2. We start with constructing a base grid of square cells. Each cell can contain at most one sample: cell diagonals are r_s. These cells are the initial pool of active cells.

We pick a random cell, then a random point inside the cell, and give it a random color. We gather the nearby extant samples using the base grid, and check for conflict using r_s, r_b, and the colors of the disks. If a sample is in conflict, it is rejected. Otherwise, we accept the sample and remove its cell from the active pool. After a number of attempts proportional to the beginning size of the active pool [16], we proceed to the next iteration.

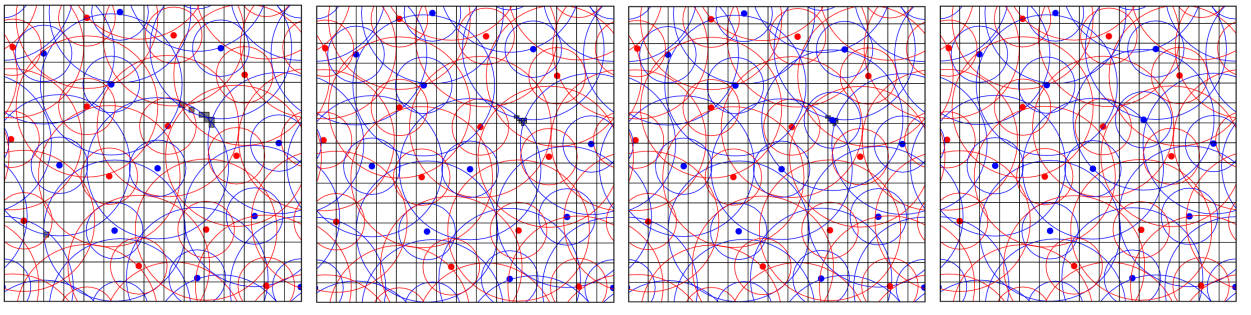
To advance the iteration, we refine all active cells, and discard child cells that are too close to extant samples to admit a point. In particular, we discard a cell if it is inside an r_s-radius disk, or if it is inside an r_b disk of *each* color. This test is conservative, in that we never discard a cell that could admit a point. However, we often retain a square that has no conflict-free area; this is handled later when we refine it and discover we can discard its children. The maximum refinement depth is machine precision, and in practice the number of active squares decreases geometrically (quickly) with iteration count (equal to refinement depth).

2.3. Advancing front algorithm

We place disks on the domain boundary. We then advance the front using *vertical* advancement, perpendicular to the front or domain boundary, until it can be advanced no further; then *horizontal* advancement, parallel to the boundary,



(a) Iteration 0 End (b) Iteration 1 Start (c) Iteration 1 End (d) Iteration 2 Start



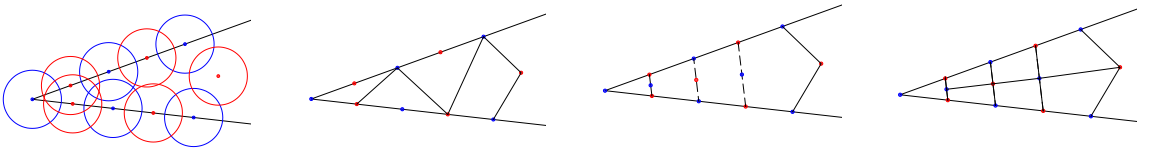
(e) Iteration 2 End (f) Iteration 3 Start (g) Iteration 3 End (h) Final

Fig. 2: The random algorithm in action. Within an iteration, we repeatedly pick an active cell (gray), pick a point in that cell and color it, and accept it if it isn't conflicting. Between iterations, all active cells are refined, and the covered children are discarded. The algorithm terminates with a maximal sampling when no active cells remain. Here $r_s = 0.095$ and $\alpha = 2.5$ and the domain is a periodic unit square.

to exhaustion. We repeat these two types of advancement until neither makes progress. We fill any remaining small gaps using the random algorithm. We now describe these steps and boundary sampling in more detail.

2.3.1. Boundary sampling

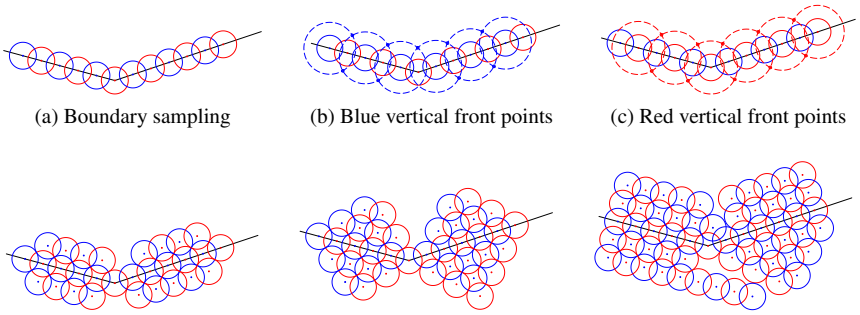
If the domain has no boundary because it is periodic or closed, then place two same-colored (blue) disks with centers $\sqrt{2}r_b$ apart. Otherwise, cover the boundary with disks; see Fig.s 3 and 4a. Place blue disks at all domain vertices. Place blue disks on the curves at least $2r_s$ apart; the ideal spacing is $\sqrt{2}r_b$. This is not possible for all curve lengths, so get as close as possible, rounding up or down. Now place red disks half-way between the blue disks.



(a) Sharp corner with conflicting boundary disks. (b) Flat boundary angles. (c) Opposite-color disks between conflicting disks. (d) No flat boundary angles remain.

Fig. 3: Extra disks near sharp corners and narrow regions remove flat angles.

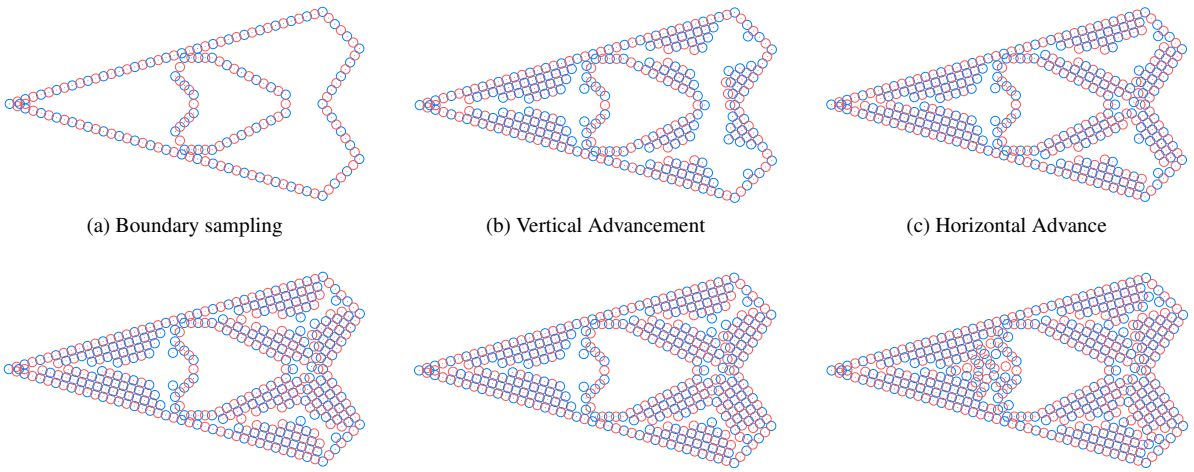
If the input has small angles, or narrow regions (compared to the disk radii), the boundary disks may conflict. Extra disks, also in conflict, are introduced to avoid flat boundary angles; see Fig. 3. Conflicts and poor quality only occur locally. Interior large angles are fixed after the packing is maximal, in Section 3. For the random algorithm, boundary sampling and handling sharp features are similar, except the spacing is random and more varied.



(a) Boundary sampling (b) Blue vertical front points (c) Red vertical front points

(d) One level of vertical advancement (e) Adjacent pyramids from vertical advancement (f) Disks after first horizontal advancement

Fig. 4: One vertical and horizontal advancement iteration from a two-sided boundary.



(a) Boundary sampling (b) Vertical Advancement (c) Horizontal Advance

(d) Vertical Advancement (e) Horizontal Advance (f) Final random packing

Fig. 5: Typical configurations after the different advancing front phases.

2.3.2. Vertical advancement

We find two overlapping big disks of the same color and at the same *level.* The disks on the boundary are all level zero. The level is the number of advancement steps preceding its generation. We seek to add another disk at their intersection points, at the next level and of the same color; see Fig. 4b. (Disks have two intersection points; each may be outside the domain or covered by disks from a lower level or another patch.) The intersection points define the "front" of the advancing front. We check the candidate at the intersection point for the standard conflict criteria defined by its two disk radii, and reject it if necessary. It is also rejected if it contains a second intersection point in its big disk.

Fig. 5b illustrates the output after vertical advancement can make no more progress. Starting with a row of disks on a single curve, the next row at the next level tends to have one fewer disk. This tends to result in pyramid-shaped arrangements as in Fig. 4e. Each row is advanced until it peters out to a single disk, or collides with a pyramid growing from some other curve, as in Fig. 5b. The order in which the front is advanced will determine the rate at which the patch for each curve grows. The algorithm is relatively insensitive to advancement order. Deterministically visiting all the intersection points at a given level for one color, then all the ones at the same level for the other color, works well. We continue advancing the front until all patches cannot grow. Depending on the angle between curves, there is often

a large uncovered area between adjacent patches. Fractured pyramids and gaps also occur due to surface curvature when meshing curved surfaces.

2.3.3. Horizontal advancement

See Fig. 4f and Fig. 5c. The disks on the end of each row are called *terminal disks*. Except for a single-disk pyramid apex, they have exactly one opposite-colored neighbor. For a blue terminal disk with a red neighbor, we attempt to add another red disk opposite the neighbor. The neighbor is at center distance $d \approx \sqrt{2}r_b/2$ from the terminal disk. The candidate is also at distance d from the terminal disk, and centered on the line through the terminal and neighbor centers. If the candidate is in conflict, we stop. We advance the terminal disks (the front) horizontally, in deterministic round-robin order, until all candidates are in conflict.

2.3.4. Filling large gaps

After neither advancement is possible, if there is still a large area uncovered, we pick one of the disks bounding it, and find a direction into it, in which we can add a same-colored disk at distance $\sqrt{2}r_b$. Also add an overlapping pair of opposite-colored disks. These seeds start a new structured patch, which is then advanced as before. After filling all large gaps, we fill any small gaps using the random algorithm.

3. Getting all-quads with good shape

We start with a maximal two-color disk packing. This may have been generated by the random algorithm, the advancing-front algorithm, or some other source such as Delaunay refinement. We construct its Delaunay triangulation with no special constraints. Then we reduce the connectivity by discarding edges connecting two vertices of the same color. Right away, this produces mostly quadrilaterals. The other *many-sided* cells contain a discarded monochromatic triangle, because of the two-coloring.

So, our problematic structures are monochromatic triangles, and non-convex quadrilaterals. We have a deterministic template-based technique that refines the monochromatic triangles, then the non-convex quads. This gives guaranteed quad quality and termination. With $\alpha = 1$, the provable guarantees are that all quad angles are in $[10.8°, 173.3°]$ and edges are in $[0.1, 2]r_s$. We emphasize that these resolution steps are local, and only affect adjacent triangles. This is in sharp contrast to the triangle pairing family of algorithms, where an unmatched triangle leads to a refinement that propagates throughout the entire domain until the boundary or another unmatched triangle is reached. We also have several optional heuristic strategies that improve quality in practice but degrade the theory guarantees; see Section 3.2.

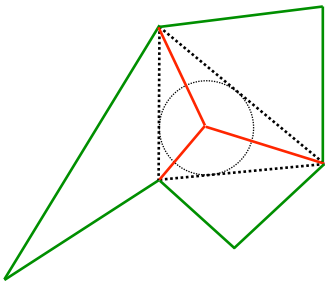
3.1. Guaranteed-quality quads by templates

There are several useful properties because of the two-coloring:

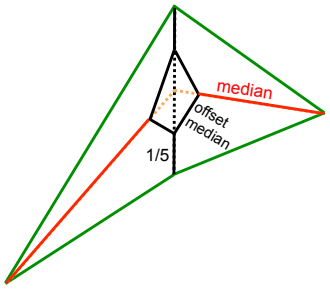
- Triangles have either one or three edges discarded.
- Many-sided cells have an even number of boundary edges.
- Many-sided cells contain a monochromatic triangle, with three discarded edges.

The last property follows from the prior properties, and because any triangulation of a five-or-more sided polygon contains at triangle with at least two internal edges.

Incircle refinement removes monochromatic triangles by adding the incenter—not the circumcenter as in Delaunay refinement—with the opposite color, and deterministically connects it to the three triangle vertices. See Fig. 6a. This produces all quads. If all of the vertices in the mesh are of one color, then this is the same as Q-TRAN refinement [5]. *Median refinement* removes large angles ($> 173°$) of quads by a 1-to-5 quad template; see Fig. 6b. Each of the two



(a) A many-sided region becomes quads by incircle-refinement.



(b) A reflex quad becomes five convex quads by median template refinement.

α		$\square\gamma$	$\square e$
1	min	$10.8°$	$0.1r_s$
	max	$173.3°$	$2r_s$
2	min	$7.7°$	$0.2r_s$
	max	$173.6°$	$4r_s$

(c) Provable quality bounds with no heuristics, final quad angles ($\square\gamma$) and edge lengths ($\square e$).

Fig. 6: Refining monochromatic triangles and reflex quads into convex quads with shape and size guarantees. In 6a, the green many-sided polygon is divided into quads with two green and two red edges. Adjacent monochromatic triangles would form a quad with four red edges. Original monochromatic edges are dashed black, original bichromatic edges are green, introduced bichromatic edges are red or solid black (except in 6b the starting triangles might have come from incircle refinement).

triangles of the quad are refined in the same way, then glued together along the common quad diagonal. The lines of the refinement are parallel to the median lines, but two are offset $1/5$ of the way along the quad diagonal.

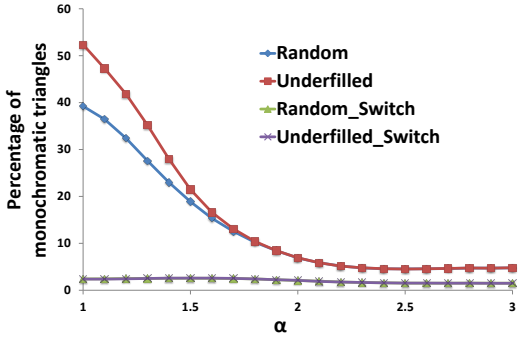
We have provable bounds on the angles and edge lengths in the triangulation and subsequent quadrangulations. These are summarized in the table in Fig. 6c, and their lengthy derivations are in the extended version of this paper, available online as a Sandia tech report [1]. We provide some intuition about why they hold here. The initial triangles from the packing are well-shaped by the standard reasons for (two-radii) maximal Poisson-disk packing [18] and Delaunay refinement [12] algorithms. Incircle refinement halves some triangle angles, but these angles are always combined with another one in a quad. It places three moderately large angles in the center of a triangle, up to $150°$. The refined edges can be as small as about half of an original edge. Median refinement can cut some of the triangle angles by about a third. Because the original triangulation had no extremely large angles (at most $120°$ in the $\alpha = 1$ case), the quad reflex angles are not too large, and median refinement will always produce convex quads. The median template can introduce some large angles where the offset medians intersect the quad diagonal. In the worst case they are only about $6°$ away from being $180°$. It can also introduce some fairly short edges. While these bounds are not great, they do provide a reasonable starting point for smoothing or other mesh improvement techniques.

3.2. Heuristic improvements

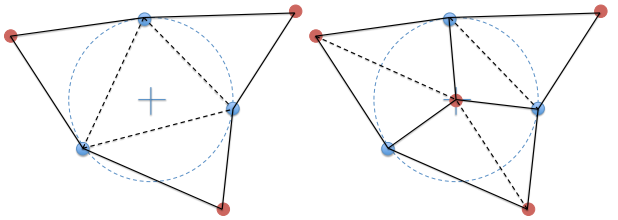
For boundary sampling, we get better mesh quality if we allow the color of disks at domain vertices to be red or blue, in order to get the spacing closer to the ideal $\sqrt{2}r_b$. (We still require alternating disk colors along each domain curve.) Also, curves with vertices of different colors can be advanced farther, so the structured patches are bigger.

We have two heuristics for reducing the frequency of problematic cells, and two for resolving them. For frequency reduction, we increase the parameter α. This is inspired by the diagonal of a square (monochromatic edge) being longer than its side (bichromatic edge). A value of $\sqrt{2}$ is a square diagonal and coincides with the value used in multi-class sampling [17], so it is the first value we tried. Larger values worked better, because they generated fewer monochromatic triangles; see the red and blue lines of Fig. 7a. Any $\alpha > 1$ actually degrades the theoretical quality guarantees; but in practice it helps, and we discovered that $\alpha = 2.5$ works best. A second, very effective strategy is to *switch* colors to reduce the number of monochromatic triangles, down to 2%; see the purple and green lines of Fig. 7a. A nice feature is that color switching does not change the point positions. However, if $\alpha > 1$, it reduces the theoretical quality guarantees farther. One heuristic that did not help was keeping the number of red and blue disks continuously balanced by assigning the initial color to be the underfilled color.

A heuristic to resolve a monochromatic triangle is Delaunay refinement with the opposite color, followed by Delaunay retriangulation; see Fig. 7b. This heuristic has some small chance of creating a new, smaller monochromatic triangle, requiring further refinement. If it terminates, it has the nice property that all edges are Delaunay. A strategy to resolve

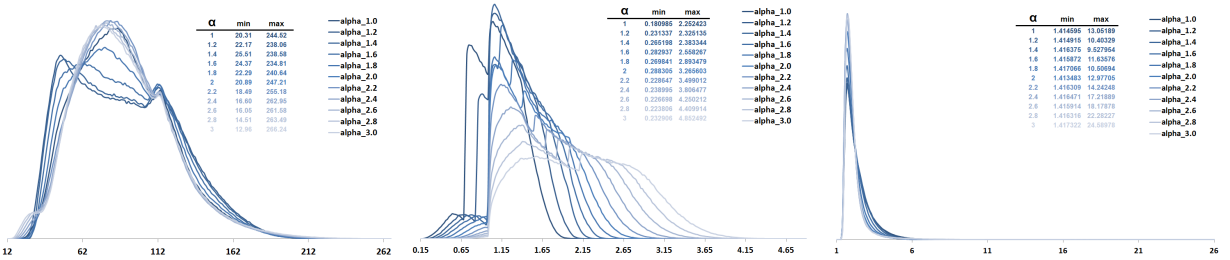


(a) Monochromatic triangle frequency. The number of initial monochromatic triangles decreases with α. *Switch* post-processing switches disk colors, and reduces the number of monochromatic triangles to about 2%.



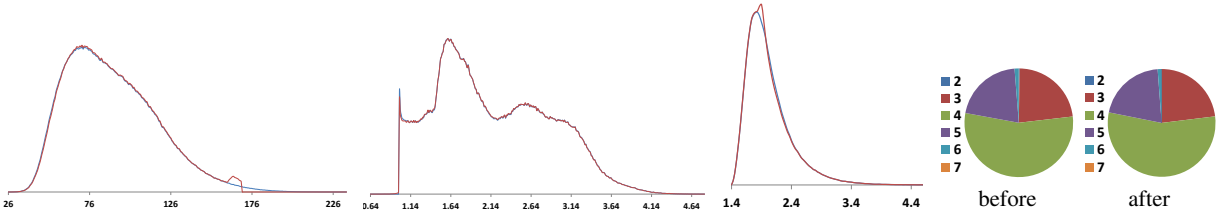
(b) Delaunay circumcenter refinement with retriangulation usually terminates quickly and produces good quads, but not always.

Fig. 7: Some heuristics for removing monochromatic triangles.



(a) Angles distribution (b) Edge length distribution (c) Aspect ratio distribution

Fig. 8: Quad quality before refining non-convex quads, as α varies.



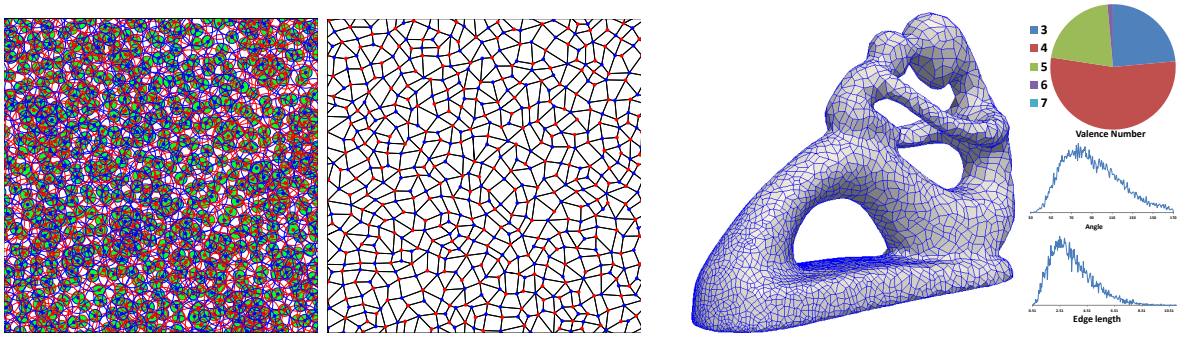
(a) Quad angles (b) Edge length (c) Aspect ratio (d) Edge valence

Fig. 9: Quad quality for our random algorithm on a periodic unit square, about one million points, and $\alpha = 2.5$; before (blue) and after (red) heuristic improvements. Resampling yields a maximum angle of 170°.

non-convex quads is resampling: remove the points in a neighborhood of the large angle, then fill it in with a different random packing. This works well down to angles of about 170°. Fig. 9 shows the typical quality after these heuristics. Of course, standard smoothing and edge flipping can also be performed on the quad mesh. None of our examples do that, in order to illustrate that our new techniques already produce a fairly good mesh. The best ways to improve quad quality, and retain provable angle bounds, remain topics for further investigation.

4. Results

Unit square with periodic boundary conditions. We generated random quad meshes of a periodic square. Fig. 9 shows the quad quality and Fig. 10a shows the disks and mesh. Fig. 11 shows that our output has blue noise, and studies the effect of varying α on the spectra.



(a) Random periodic unit square with $r_s = 0.02$ and $\alpha = 2.5$.

(b) Varying sizing function on Fertility, 5k points, $\alpha = 2.5$.

Fig. 10: Delaunay quadrangulations of periodic or curved domains.

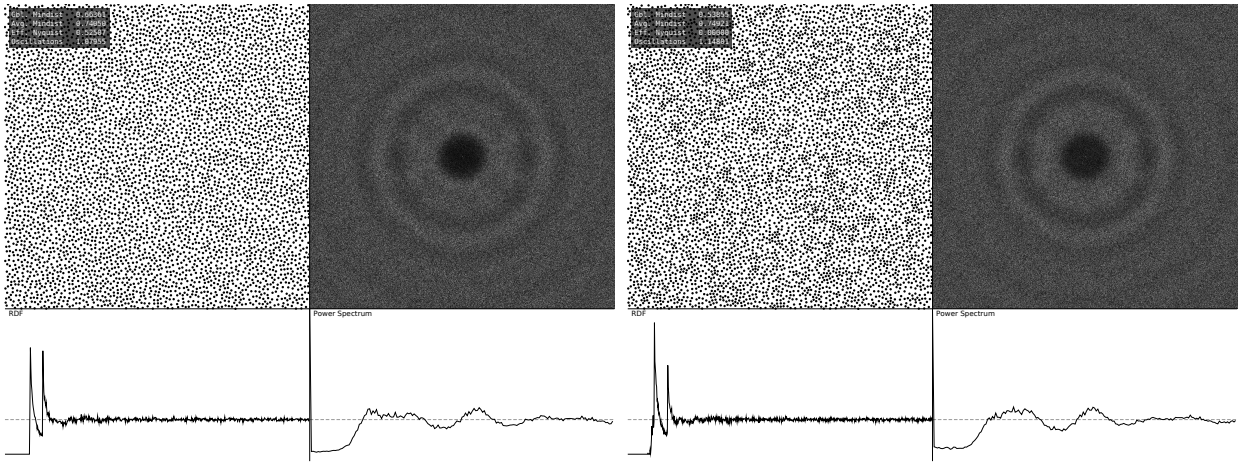
Non-convex domain with uniform sizing function. In Fig. 12 we compare our advancing front algorithm with the paving algorithm from Cubit. Cubit's output is superior in some respects, but this illustrates that our quality is reasonable enough that a production capability could be built on it. See Fig. 13 for more planar examples.

Curved surface with uniform sizing function. Our algorithms extend to curved surfaces. For the random algorithm, instead of a background grid of squares, we start with a (graphics) triangulation of the curved surface. These triangles are the initial active pool. To generate the candidate sample, a triangle is chosen uniformly by area, then we generate a uniform-random point inside it. Some large triangles might be able to contain more than one sample, so we must check before discarding it. We use the Euclidean distance for simplicity, which is an approximation. When advancing the iteration and refining the active cells, we split each triangular cell into four triangles. We also have an advancing front variation for curved surfaces. The results of the random algorithm are shown in Fig. 14, and the advancing-front algorithm in Fig. 15.

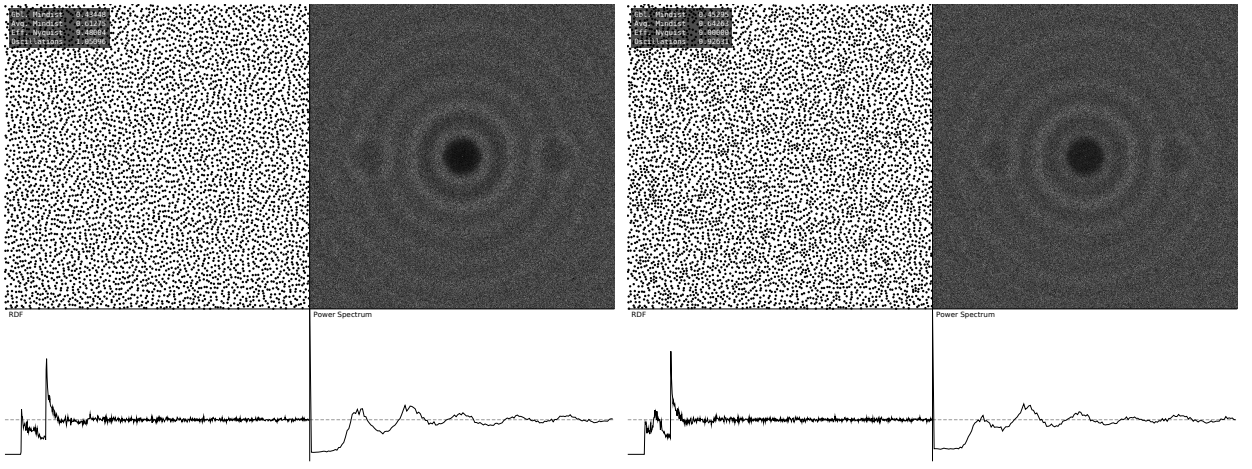
Curved surface with varying sizing function. Our random algorithm can use a spatially-varying sizing function over a curved surface. We require that we can obtain the sizing function value for any random point on the surface, for example by interpolation in its containing triangle. As before, we pick a random triangle and chose a random point from it. We define r_b to be the sizing function value at the point, and $r_s = r_b/\alpha$. These are used to define conflict between same-colored and opposite-colored points as before. Since the sizing function varies the radius for the point and a neighboring point may be different; we define disk conflict by the **Smaller Disks** criterion [18]. That is, the disks are in conflict if the center of the larger disk lies inside the smaller disk. When checking if refined child cells are covered the only change is that the r_b and r_s of each nearby sample is different. The mesh for the Fertility model is shown in Fig. 10b.

5. Conclusions

We have introduced a novel concept, a bichromatic Delaunay quadrangulation, based on assigning points one of two colors, and only retaining the Delaunay edges between opposite-colored points. In spatially-isolated cases, constrained incircle refinement is needed to produce all quads, without any six-or-more sided cells. Template refinement or re-sampling is needed in spatially-isolated cases to remove large angles. For the best quality, we define two radii around each point, so that same-colored points must be farther apart than opposite-colored points. We have demonstrated

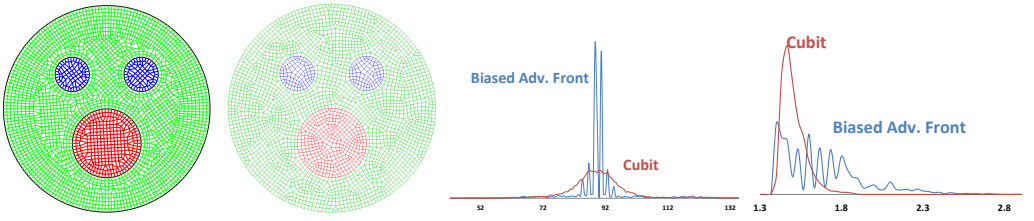


(a) $\alpha = 1.5$ (b) $\alpha = 1.5$



(c) $\alpha = 2.5$ (d) $\alpha = 2.5$

Fig. 11: Spectral analysis using PSA [23], a standard tool of the graphics community. Left column, (a) (c), before refining monochromatic triangles; right column, (b) (d), after. Top row is $\alpha = 1.5$ and bottom row is $\alpha = 2.5$. In each quad-chart, upper left is the points, and upper right is the Fourier transform of all point-to-point distance vectors. Lower left shows the 1-d histogram of the magnitude of these distances, scaled by area so uniform-random points would generate a horizontal line; for MPS, this resembles a step function with an 3× tall spike at the beginning of the step. Lower right is the Fourier transform of the 1-d histogram; for MPS, this is close to the sinc function.



(a) Our mesh (b) Cubit paving mesh (c) Quad angles (d) Quad aspect ratio

Fig. 12: Advancing front vs. Cubit's paver. Our lack of smoothing and quad cleanup shows in the jaggedness of (c) and (d). We create larger structured patches.

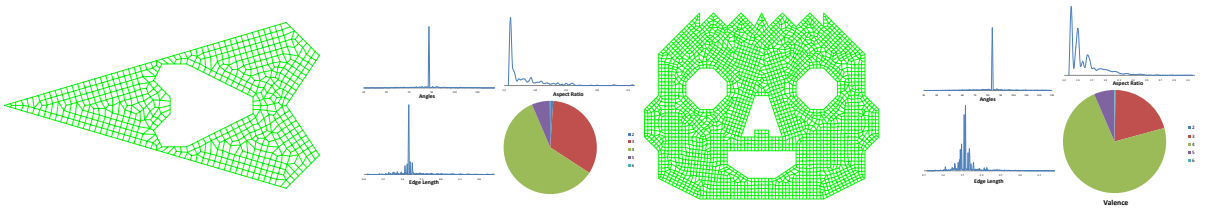


Fig. 13: Advancing-front Delaunay quadrangulation of non-convex planar surfaces.

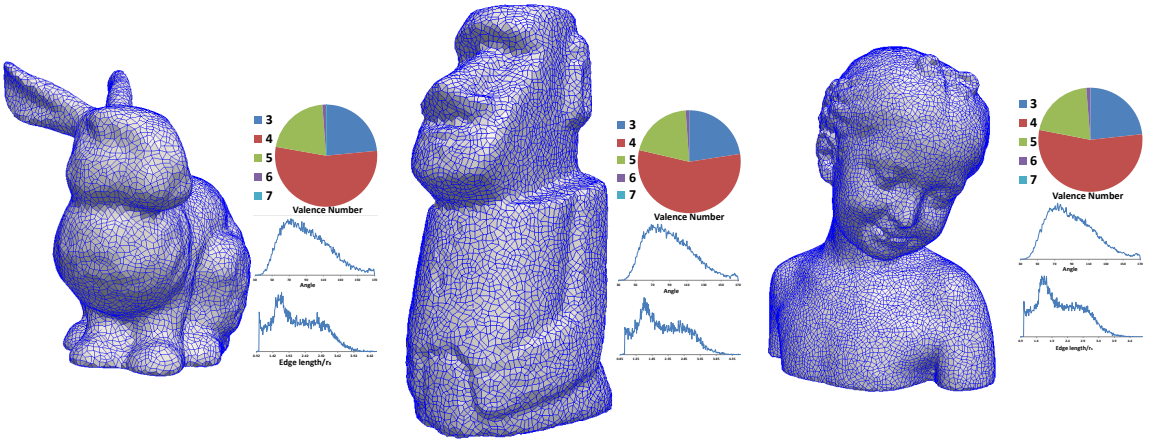


Fig. 14: Random Delaunay quadrangulations of curved surfaces: Bunny (10k points), Moai (10k points), Bimba (20k points).

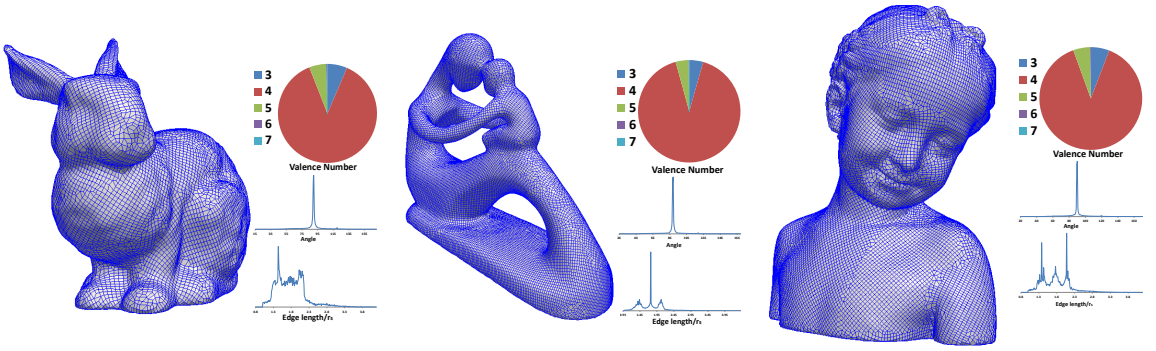


Fig. 15: Delaunay Quadrangulation of curved surfaces using advancing front: Bunny (10k points), Fertility (20k points), Bimba (20k points).

both a random algorithm that produces good spectral properties, and an advancing front algorithm that produces a structured mesh over large patches. We can also quadrangulate the points of well-spaced sphere-packings from other algorithms, such as a triangulation produced by classical Delaunay refinement.

We have demonstrated these algorithms on a variety of models, and studied the effects of tuning the ratio of the two radii. We have an initial demonstration of variable sizing functions on curved surfaces; we would like to understand the limits of how fast the sizing function can vary and still get good quality and robustness. We would like a size-varying version of our advancing front algorithm. In some applications it may be desirable to control the extent of the

structured patches and irregular vertices, switching between the structured and random algorithms in a more nuanced way.

While our output has provable quality, the bounds are not that good. We would like to discover ways to achieve better quality in theory and practice. This may involve some combination of different point placement algorithms, and different monochromatic triangle and large-angled quad resolution algorithms, as well as standard quad smoothing and cleanup.

Acknowledgements

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] S. A. Mitchell, M. A. Mohammed, A. H. Mahmoud, M. S. Ebeida, Delaunay quadrangulation by two-coloring vertices — extended version with quad-quality proofs appendix, Technical Report, Sandia National Laboratories, 2014. Available at `http://www.cs.sandia.gov/~samitch/bibliography_2007.html`.

[2] D. Bommes, B. Lévy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, D. Zorin, Quad-mesh generation and processing: A survey, Computer Graphics Forum 32 (2013) 51–76.

[3] S. Owen, A survey of unstructured mesh generation technology, in: International Meshing Roundtable, volume 7, Sandia National Laboratories, 1998, pp. 26–28.

[4] T. Itoh, K. Shimada, Automatic conversion of triangular meshes into quadrilateral meshes with directionality, Int. J. CAD/CAM 1 (2002) 11–21.

[5] M. S. Ebeida, K. Karamete, E. Mestreau, S. Dey, Q-TRAN: a new approach to transform triangular meshes into quadrilateral meshes locally, in: International Meshing Roundtable, volume 19, Sandia National Laboratories, 2010, pp. 23–34.

[6] T. Schiffer, F. Aurenhammer, M. Demuth, Computing convex quadrangulations, Discrete Appl. Math. 160 (2012) 648–656.

[7] J. Z. Zhu, O. C. Zienkiewicz, E. Hinton, J. Wu, A new approach to the development of automatic quadrilateral mesh generation , Int. J. Numer. Meth. Eng. 32 (1991) 849–866.

[8] T. D. Blacker, M. B. Stephenson, Paving: A new approach to automated quadrilateral mesh generation, Int. J. Numer. Meth. Eng. 32 (1991) 811–847.

[9] D. R. White, P. Kinney, Redesign of the paving algorithm: Robustness enhancements through element by element meshing, in: International Meshing Roundtable, volume 6, Sandia National Laboratories, 1997, pp. 323–335.

[10] M. Bern, D. Eppstein, Quadrilateral meshing by circle packing, Int. J. Comp. Geom. & Appl. 10 (2000) 347–360.

[11] D. Talmor, Well-spaced points for numerical methods, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, 1997. CMU CS Tech Report CMU-CS-97-164.

[12] L. P. Chew, Guaranteed-quality triangular meshes, Technical Report 89-983, Department of Computer Science, Cornell University, 1989.

[13] M. S. Ebeida, S. A. Mitchell, A. A. Davidson, A. Patney, P. M. Knupp, J. D. Owens, Efficient and good Delaunay meshes from random points, Computer-Aided Design 43 (2011) 1506–1515.

[14] M. S. Ebeida, A. Patney, S. A. Mitchell, A. Davidson, P. M. Knupp, J. D. Owens, Efficient maximal Poisson-disk sampling, ACM Transactions on Graphics 30 (2011).

[15] M. N. Gamito, S. C. Maddock, Accurate multidimensional Poisson-disk sampling, ACM Transactions on Graphics (TOG) 29 (2009) 8.

[16] M. S. Ebeida, S. A. Mitchell, A. Patney, A. A. Davidson, J. D. Owens, A simple algorithm for maximal Poisson-disk sampling in high dimensions, Computer Graphics Forum 31 (2012) 785–794.

[17] L.-Y. Wei, Multi-class blue noise sampling, ACM Trans. Graph. 29 (2010) 79:1–79:8.

[18] S. A. Mitchell, A. Rand, M. S. Ebeida, C. Bajaj, Variable radii Poisson-disk sampling, in: Canadian Conference on Computational Geometry, volume 24, 2012, pp. 185–190.

[19] A. Soifer, The Mathematical Coloring Book, Springer–Verlag, 2008.

[20] S. A. Mitchell, A characterization of the quadrilateral meshes of a surface which admit a compatible hexahedral mesh of the enclosed volume, in: STACS 96, Springer Berlin Heidelberg, 1996, pp. 465–476.

[21] J.-F. Remacle, J. Lambrechts, B. Seny, E. Marchandise, A. Johnen, C. Geuzainet, Blossom-quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm, International Journal for Numerical Methods in Engineering 89 (2012) 1102–1119.

[22] L. Velho, D. Zorin, 4-8 subdivision, Comput. Aided Geom. Des. 18 (2001) 397–427. Special issue on Subdivision Algorithms.

[23] T. Schlömer, PSA point set analysis, `http://code.google.com/p/psa/`, 2011.