Computer-Aided Design 85 (2017) 83–98

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

All-quad meshing without cleanup☆,☆☆

Ahmad A. Rushdi [a,b,*], Scott A. Mitchell [b], Ahmed H. Mahmoud [c], Chandrajit C. Bajaj [a], Mohamed S. Ebeida [b]

[a] Institute for Computational Engineering and Sciences, University of Texas, Austin, TX, USA
[b] Sandia National Laboratories, Albuquerque, NM, USA
[c] University of California, Davis, CA, USA

ARTICLE INFO

Keywords:
All-quadrilateral meshing
Guaranteed quality
Sharp features

ABSTRACT

We present an all-quad meshing algorithm for general domains. We start with a strongly balanced quadtree. In contrast to snapping the quadtree corners onto the geometric domain boundaries, we move them away from the geometry. Then we intersect the moved grid with the geometry. The resulting polygons are converted into quads with midpoint subdivision. Moving away avoids creating any flat angles, either at a quadtree corner or at a geometry–quadtree intersection. We are able to handle two-sided domains, and more complex topologies than prior methods. The algorithm is provably correct and robust in practice. It is cleanup-free, meaning we have angle and edge length bounds without the use of any pillowing, swapping, or smoothing. Thus, our simple algorithm is fast and predictable. This paper has better quality bounds, and the algorithm is demonstrated over more complex domains, than our prior version.

1. Introduction

Generating good quality meshes is a challenging problem with many engineering applications: e.g., Finite Element Analysis (FEA) [1–3] and Computer-Aided Design (CAD) [4,5]. The quality of the mesh plays a significant role in the accuracy and stability of the numerical computation. It is difficult to determine if a mesh possesses even the minimal quality necessary to undertake a computational analysis without concrete metrics. Since the necessary criteria are often application specific, the community has gravitated toward quality metrics based on geometric criteria that are usually sufficient. For example, angle minimum and maximum bounds are commonly used, along with element size, aspect ratio, skew, stretching, and orientation.

Many algorithms for triangular meshing are robust and provide guaranteed quality, and are readily available. In that sense triangular meshing is well-developed, but quad meshing is not. Prior quad methods often have difficulty in achieving angles in $[45^\circ, 135^\circ]$. Indeed, sharp features of the input domain may make this impossible. However, even without sharp input features, many prior methods create flat elements that require post-processing to achieve good angles. Another shortcoming is that many methods only work on restricted classes of input domains, such as meshing only one side of a geometric boundary, or vertices must have four or fewer curves. Ideally, one would like a provably good algorithm (i.e., with quality bounds) guaranteed to work on arbitrary topology, including point sets, two-sided domains, and embedded curves.

1.1. Related work

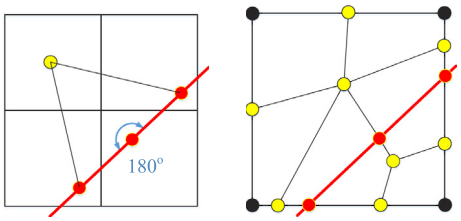
Unstructured all-quad meshing algorithms are usually categorized into two main categories: *indirect* and *direct*. A classical indirect approach starts with a triangular mesh, and then transforms the triangular elements into quadrilateral elements, via optimization [6,7], refinement and coarsening [8], or simplification [9]. A class of indirect methods start with a triangular mesh and applies the mid-point subdivision rule [10,11] to split a triangle into three quad elements. This can be generalized to other local subdivision

☆ This paper has been recommended for acceptance by Franck Ledoux and Katherine.

☆☆ This paper is extended from an earlier version published in the proceedings of the International Meshing Roundtable (IMR) 2015.

* Corresponding author at: Institute for Computational Engineering and Sciences, University of Texas, Austin, TX, USA.
E-mail address: arushdi@utexas.edu (A.A. Rushdi).

http://dx.doi.org/10.1016/j.cad.2016.07.009
0010-4485/



(a) Dual contouring. (b) Our algorithm.

Fig. 1. Starting with a uniform grid (black): comparing mesh element quality of our algorithm to that of dual contouring when an element is intersected by a domain boundary (red). Dual contouring generates a dual quad element whose vertices are preferably on the boundary (red) or the center of a non-intersected element (yellow), which often results in an undesired flat angle degrading the mesh quality. Our algorithm modifies the existing grid by adding points at the boundary intersections (red), splitting the intersected element, adding refinement points (yellow) at the middle of all edges and centers of split elements, and refining the split elements into well-shaped all-quad elements with good angles bounds. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

operations. For example, diamond–kite [12] applies recursive subdivision based on a regular tiling composed of only diamonds and kites, but does not handle domain boundaries. Q-Morph [13] is a popular indirect approach that follows a sequence of systematic triangle transformations to create an all-quadrilateral mesh. However, Q-Morph requires topological cleanup and smoothing to guarantee the quality of the final all-quad mesh. Q-Tran [14] is another indirect algorithm that produces quadrilaterals with provably-good quality without a smoothing post-processing step, and manages to handle domain boundaries. Nevertheless, the class of indirect methods typically suffers from a large number of irregular nodes that are connected to more (or less) than four mesh elements, which is typically undesired in numerical simulations.

Direct approaches construct quadrilaterals without an intermediate triangular mesh. The advancing front algorithms (e.g., paving [15]) generate all quad meshes by placing mesh points on the boundaries of the input domain, then recursively projecting edges on the front towards the interior of the domain to form quads [16,17]. However, these typically suffer from stability problems that require heuristic cleanup operations. Grid based methods construct a uniform Cartesian or quadtree background grid then modify it to conform to the domain boundaries [18,19]. These methods are easy to implement. They can often provide quality guarantees [20–22]. However, many variations produce flat or inverted elements before cleanup and smoothing, by snapping multiple grid points to lie exactly along some curve. Square packing [23] and circle packing [24] methods can generate all quads. Circle packing bounds the maximum angle to 120°, but does not bound the minimum angle. In Atalay et al. [25], the minimum angle is bounded.

Other methods include surface parameterization and orientation field methods [26–28] which find the smoothest orientation field on a surface subject to a target direction or boundary conditions. Similarly, the Quadcover [29] approach computes a global continuous parameterization for an arbitrary given simplicial 2-manifold. However, it is not trivial to find globally optimal orientation solutions.

In this paper, we describe a direct quadtree-based algorithm that produces an all-quad mesh. It is stable, and all operations are simple and local. The output mesh conforms to both the interior and exterior of the domain geometry. It has provably-good quality without post-processing cleanup. The key innovation is to move grid points away from the geometry, rather than snapping onto the geometry. An earlier version of this paper was published in the proceedings of the International Meshing Roundtable (IMR) 2015 [30]. Here we have dramatically improved the provable quality bounds, and demonstrated the method on more complex domains.

1.2. Comparison to dual contouring

Dual contouring is a direct all quad meshing technique that has attracted recent attention [31–33]. It starts with a uniform structured quadrilateral mesh. It is intersected with the domain geometry to produce polygons, which are then dualized to form an all-quad mesh. We highlight a few key advantages for our algorithm over dual contouring:

- **Element Quality**: **Angle Bounds**. Dual contouring adds points exactly on the domain's boundary, creating a large number of flat or close to flat angles which degrades the quality of the resulting mesh. To remove the flat angles, dual contouring requires post-processing "pillowing" [34]: a layer of quads is added between the geometry and the flat angle. Pillowing gives smoothing the freedom to move the node of the flat angle. In contrast, our algorithm creates a grid with points that are far from the boundary. Intersecting the grid with the boundary creates elements with balanced angles and aspect ratio. Our method is clean-up free, and avoids pillowing and smoothing. Fig. 1 contrasts the flat angles from dual contouring with our balanced angles.
- **Employing Quadtrees**. A desired mesh feature is to allocate more points around the boundary and less far from it. Our algorithm can achieve that since it starts with a balanced quadtree refinement, positioning more points "around" the domain's boundary. Dual contouring requires a uniform grid, and using a quadtree would create non-quad elements.
- **Local versus Global Operations**. Modifying an existing dual contouring mesh requires the creation of the dual mesh, and pillowing operations propagate. Our algorithm, on the other hand, can easily adapt to any input domain modification by applying the same steps locally in the neighborhood of the domain modification.

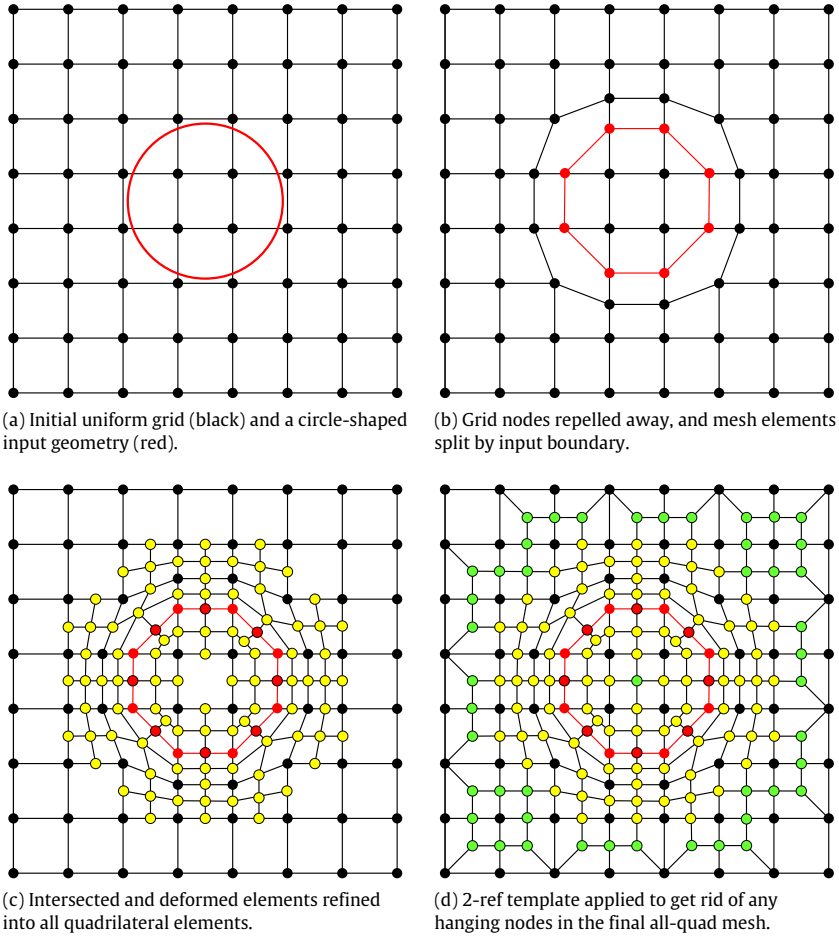
1.3. Terminology

To easily distinguish between the initial grid, the domain geometry, and the final mesh, we use the following terminology. The initial quadtree grid is composed of *squares*, each of which has four *sides* and *corners*. Corners of adjacent smaller squares appear as *hanging nodes*. The side length of a square is denoted s. (A uniform Cartesian grid is a special case of a quadtree.) The geometric domain is composed of *curves* (which are straight line segments, and two-sided) and *vertices*. The intersection of the quadtree and geometry results in *polygons* and *corner* points connected by *segments*. Polygons are meshed with midpoint subdivision; a *midpoint* subdivides each segment, and a *center* is placed interior to the polygon and connected to each midpoint. The final all-quad mesh is composed of *elements*, *edges*, and *nodes*.

2. Algorithm

Our algorithm is described as a set of repelling, splitting, and refinement steps. These are illustrated in Fig. 2 for the simple case of a geometric circle and uniform grid. We summarize these steps then explain the details in the following subsections.

1. Start with a strongly-balanced quadtree, refined to the level that vertices and disjoint curves do not appear in the same square.
2. Perturb the quadtree by repelling mesh points *away* from the geometry. All squares intersected by a vertex or curve are split into polygons, which conform to the geometry.



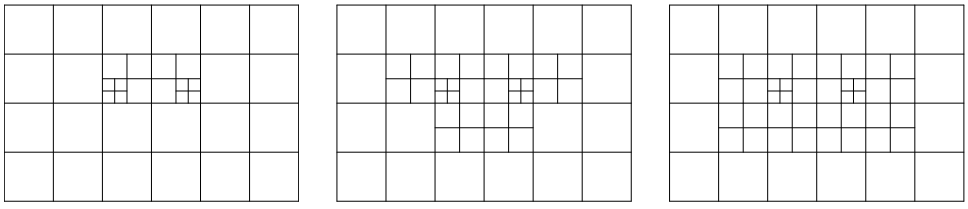
(a) Initial uniform grid (black) and a circle-shaped input geometry (red).

(b) Grid nodes repelled away, and mesh elements split by input boundary.

(c) Intersected and deformed elements refined into all quadrilateral elements.

(d) 2-ref template applied to get rid of any hanging nodes in the final all-quad mesh.

Fig. 2. An overview of the all-quad meshing algorithm, applied to a circle shape. Starting with a uniform Cartesian grid, we repel points that are too close to the boundary away from it with a ratio of the initial grid spacing, and split all elements that are intersected by the boundary. Each split or deformed element is then refined into four or more all-quad elements using the mid-point subdivision rule. Finally, we apply the two-refinement templates to guarantee conforming refinement with no hanging nodes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) An unbalanced quadtree.

(b) A balanced quadtree.

(c) A strongly-balanced quadtree.

Fig. 3. (a) An unbalanced quadtree of 3 levels. In (b), a balanced tree is one where no edge is shared between two squares with more than a difference of 1 in levels. In (c), a strongly-balanced tree is one where no point is connected to two squares of more than a difference of 1 in levels. If either condition is violated, the larger square gets refined. We require a strongly-balanced quadtree to enable the 2-ref templates in the last step.

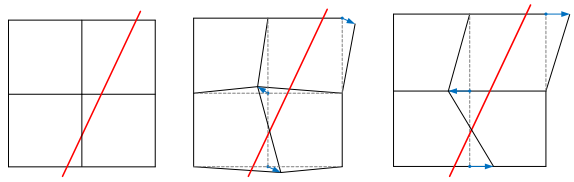
3. Apply the midpoint subdivision rule to split polygons into quad elements. Some other elements will possess hanging nodes.
4. Employ the two-refinement templates to resolve all hanging nodes, generating the final conforming all-quad mesh with good quality.

2.1. Quadtree initialization, refinement, and balancing

We start with a Cartesian grid and refine it to form a quadtree that captures the fine details of the geometric domain boundaries. In principle one could start with another decomposition. Our algorithm requires that the quadtree is strongly balanced, where corner-adjacent squares differ in size by at most one; see Fig. 3 for an illustration. For simplicity, in this version we chose uniform square sizes along the boundaries.

2.2. Perturbing the quadtree

The goal of this step is to prevent small mesh edges and flat and sharp angles. A raw geometry-quadtree intersection might be arbitrarily close to another point, or at a very small angle. We repel corners of squares *away* from nearby geometry, to be at least distance δ from geometry. We choose δ to be $\frac{s}{4}$, where s is the size of the square containing the corner. We have two repelling strategies, as shown in Fig. 4.



(a) Grid points close to boundary. (b) Geometry-normal repelling. (c) Grid-aligned repelling.

Fig. 4. Repelling grid points away from boundary if they are too close to it as in (a). In (b), repelling moves a corner in the direction normal to the boundary until its Euclidean distance is exactly δ. In (c), the curve has slope larger than one and is considered *vertical*. Therefore, repelling moves corners *horizontally* along grid lines until they are δ away in the horizontal direction.

- **Geometry-Normal Repelling**. Corners are moved normal to a curve until they are δ from it.
- **Grid-Aligned Repelling**. Corners are moved horizontally or vertically until they are δ from a curve in the horizontal or vertical direction. A *horizontal* curve has absolute slope less than one; otherwise it is *vertical*. We move a corner vertically away from a horizontal curve, and vice versa.

2.3. Polygon formation

Squares cut by the geometry are split into polygons. Squares cut by a single curve are split into two polygons, either a triangle and a pentagon, or two quadrilaterals. To recover a conforming all-quad mesh, we refine these polygons using the midpoint subdivision rule. Fig. 5 shows example elements before and after refinement. We refine squares cut by two or more curves until only one curve cuts a square. Note that when two curves meet at a common vertex at a sharp angle, the element containing that angle will have poor quality. In this case we do not refine adjacent squares

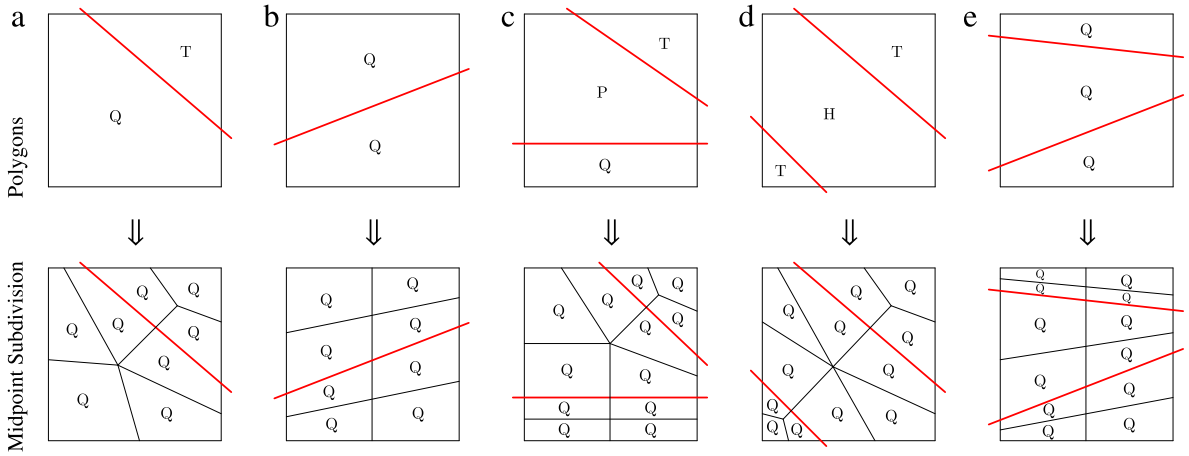


Fig. 5. Example grid-geometry intersection and the resulting midpoint subdivision mesh. Input geometry curves split squares into polygons: Triangles (T), Quads (Q), Pentagons (P), or Hexagons (H). Each polygon is divided into quad elements. The top row shows intersected elements, while the bottom shows the split and divided mesh. Columns show the cases when the boundary splits a quad element into (a) a triangle and a pentagon, (b) two quads, (c) a triangle, a quad, and a pentagon, (d) two triangles and a hexagon, and (e) three quads.

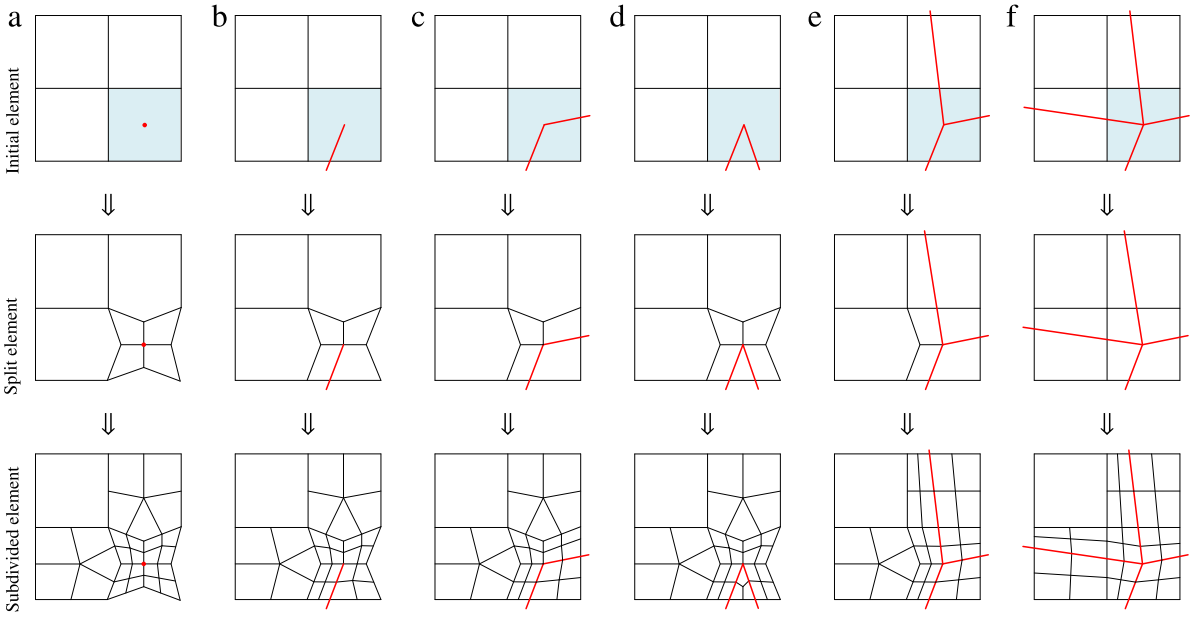
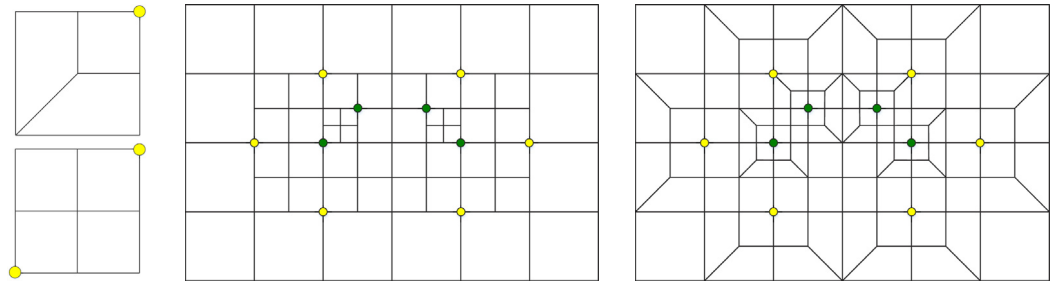
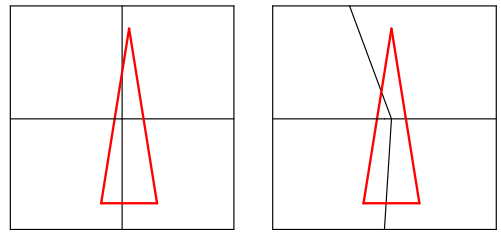


Fig. 6. Refining a square containing a geometric vertex into all-quad elements: (a) an isolated vertex, (b) a hanging edge, (c) and (d) vertex with two line segments, (e) vertex with three curves, (f) vertex with four curves. Other intersection scenarios can be decomposed into superpositions of these cases. The square is first split into four quad elements, with pulling the new corners closer to the vertex to avoid the creation of a flat angle. Then, these quads and the squares sharing an edge with the initial element get split into quad elements by midpoint subdivision. This generates nearby squares with hanging nodes, which are resolved in the last step of the algorithm.



(a) Templates. (b) A quadtree with hanging nodes. (c) A balanced quadtree with templates applied.

Fig. 7. Applying the two-refinement templates to the strongly balanced quadtree with hanging nodes in Fig. 3. The two-refinement templates are shown in (a), and applied to the marked mesh points in (b) to eliminate the hanging nodes as in (c).



(a) An extreme sharp corner case of an isosceles triangle. (b) Trapped point is centered half way between geometry curves.

(c) Splitting elements. (d) Subdividing elements.

Fig. 8. Applying our all-quad algorithm to a vertical isosceles triangle. The trapped mid-point is centered between the two vertical geometry curves. Our algorithm is flexible enough to handle this sharp corner, but does not provide a quality bound in this case.

to separate these curves. The mesh between the two curves may have stretched elements of poor quality, and the δ separation may not be achievable.

2.3.1. Handling geometric vertices

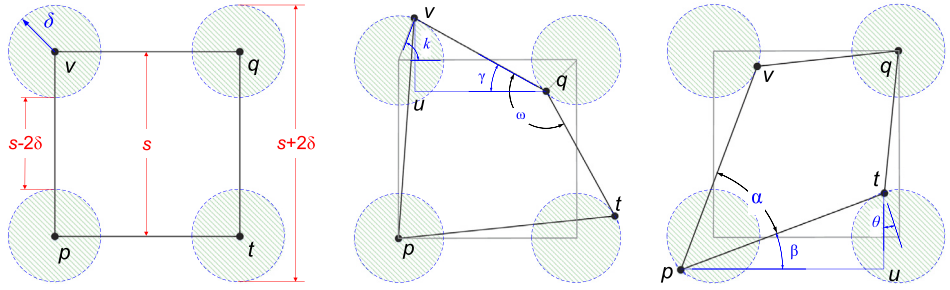
We seek to preserve sharp features of the input boundary domain without introducing new ones. We require that a square contains at most one vertex. We insert a new mesh point coincident with a domain vertex, and split any element that contains a vertex before we apply midpoint subdivision.

Fig. 6 shows examples of initial, split, and refined elements for a vertex. We consider isolated vertices, as well as vertices with one or more curves. In principle a vertex may have an arbitrary number of curves. For each square side that does not intersect a curve, we introduce its midpoint as a mesh point, and pull it closer to the sharp corner (a distance of $s/8$, where s is the square side length) to avoid creating flat angles in the adjacent elements. Squares are split into multiple regions by the curves, and vertex–midpoint edges. Each region is a polygon that is meshed with midpoint subdivision independently. Adjacent squares are also meshed with midpoint subdivision, introducing hanging nodes on some of their neighbors. These, along with hanging nodes from adjacent squares being refined to a different level, are resolved in the last step of the algorithm.

Note that extreme sharp corner case might result in trapped grid points with no enough space for repelling. In this case, we position the grid point half-way between the trapping geometry curves. See Fig. 8 for an extreme example of a vertical isosceles triangle. Note that while our algorithm is flexible enough to handle sharp corners, we only analyze quality bounds near single curves in Section 3.

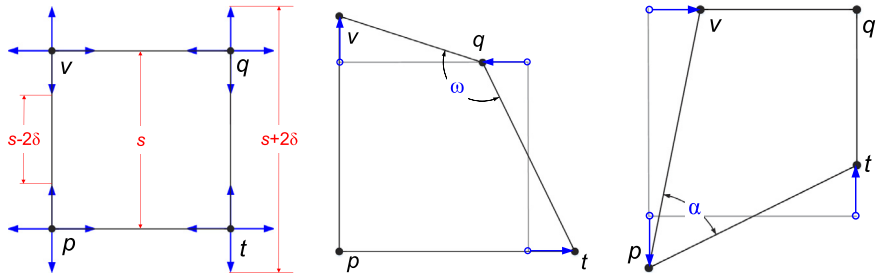
2.4. Handling hanging nodes

Squares with a geometric curve or vertex produced polygons and were meshed with midpoint subdivision. We now consider the remaining empty squares. Squares with no hanging nodes are immediately mesh quads. Squares with hanging nodes are meshed using the two-refinement (2-ref) templates with corner marking [35]. The two-refinement templates shown in Fig. 7(a) split an element with a hanging node into either three or four quads. Using a checker board pattern, we mark square corners for template application. In essence, the marking provides a local and globally consistent way to pair adjacent squares with one or three



(a) Possible boundary-normal corner movements. (b) Corner movements for the largest possible angle. (c) Corner movements for the smallest possible angle.

Fig. 9. Angles and edge lengths for boundary-normal repelling corners of a square.



(a) Possible grid-aligned corner movements.
(b) Corner movements for the largest possible angle.
(c) Corner movements for the smallest possible angle.

Fig. 10. Angles and edge lengths for grid-aligned repelling corners of a square.

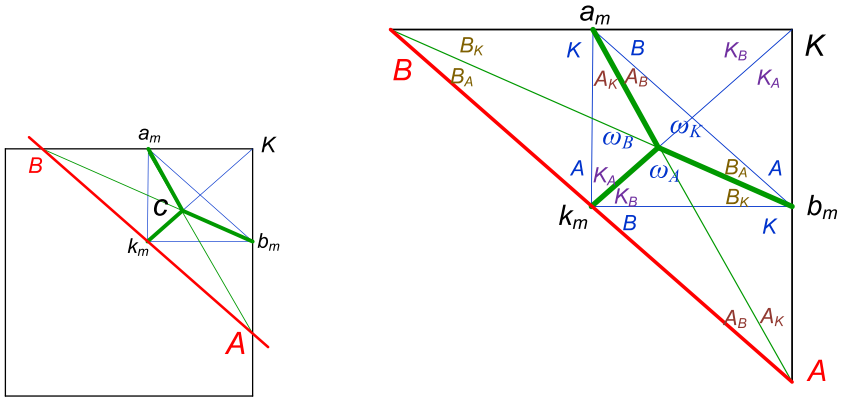
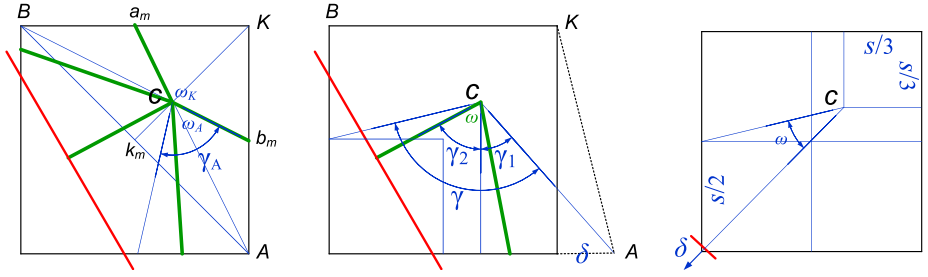


Fig. 11. Angles in a triangular polygon split into quads along segments from corners to midpoints.



(a) Midpoint subdivision and ω_A.
(b) Max ω opposite a cut corner.
(c) Min ω opposite a cut corner.

Fig. 12. Pentagon midpoint subdivision, and worst-case center angles ω.

hanging nodes together, providing an all-quad mesh that allows size transitions. For example, the strongly-balanced quadtree with hanging nodes in Fig. 3 is marked for 2-ref template application in Fig. 7(b) and consequently refined in Fig. 7(c).

3. Analysis

In this section, we analyze the theoretical bounds on the element quality in terms of angle bounds and edge lengths. We denote the maximum angle: ω, the minimum angle: α, the maximum edge length: $e_{\max}$, and the minimum edge length: $e_{\min}$. We take several cases into consideration, proceeding from simple to hard cases, to analytically study the impact of repelling and mid-point subdivision. We do not analyze the cases of a sharp input angle at a vertex: the achieved angles and edge lengths depend on this angle.

3.1. Squares without geometry or moved corners

Quadtree squares that do not interact with the geometry are meshed into quads using templates, depending on the hanging nodes. It is trivial to see that these result in edge lengths between s and $s/2$ (for a side with a hanging node) and angles between 45° and 135°, as can be seen in Fig. 7(a).

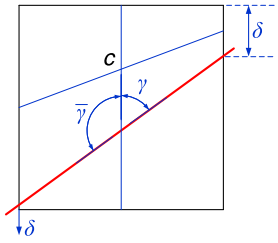
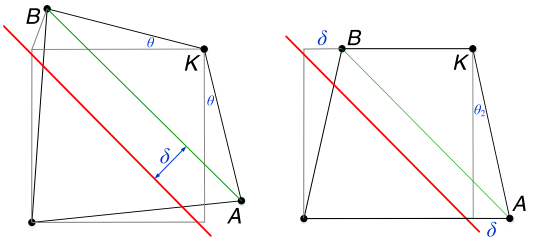


Fig. 13. The worst-case angles in a quadrilateral polygon.

3.2. Squares with moved corners, without geometry

Corners of a square with no hanging nodes would be moved in our algorithm using either boundary-normal repelling, or grid-aligned repelling. Each approach results in slightly different angle

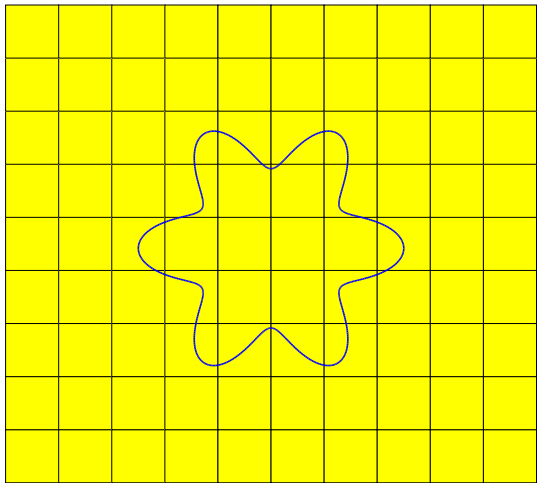


(a) Boundary-normal repelling. (b) Grid-aligned repelling.

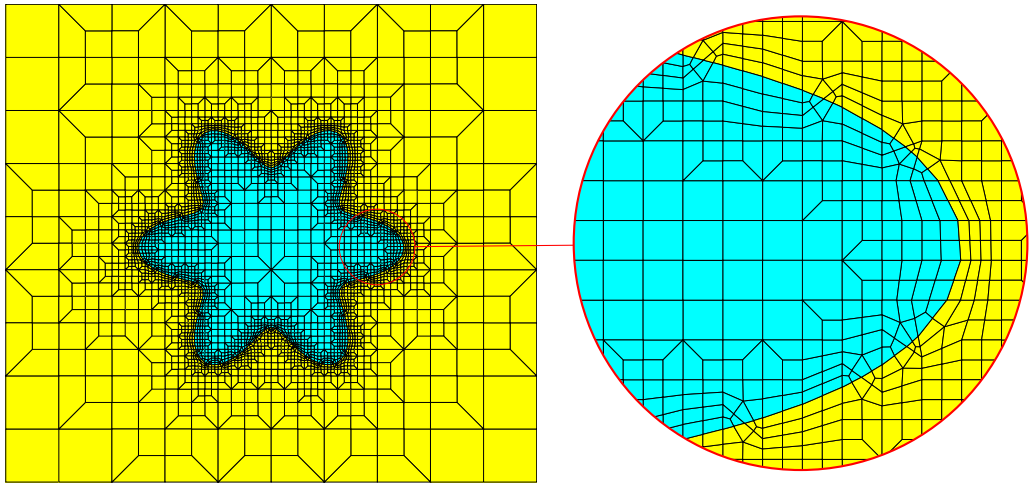
Fig. 14. Worst case corner angles for a pentagon.

bounds. Figs. 9 and 10 show the geometry of the largest and smallest possible angles at a corner after repelling by δ. The edge length $|e|$ varies in both repelling cases between $s - 2\delta$ and $s + 2\delta$. Next, we derive formulas for the expected minimum and maximum angles, and quantify their values for the experimental choice of $\delta = s/4$:

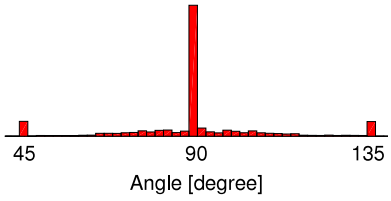
Boundary-Normal Repelling. Starting at Fig. 9(a), the largest angle would be formed when a corner, say q, moves towards the square's center a distance δ in the direction $\vec{qp}$, while its neighbor corners v and t move a distance δ outwards in directions such that qv and qt are orthogonal to the displacements of v and t, respectively. In this case, as shown in Fig. 9(b), the horizontal distance between q and v is $|uq| = s - \frac{\delta}{\sqrt{2}} - \delta\cos\kappa$, while the vertical distance $|uv| = \frac{\delta}{\sqrt{2}} + \delta\sin\kappa$. This yields an angle γ between vq and qu, where $\tan\gamma = |uv|/|uq|$. The maximum angle ω at corner q is therefore equal to $90^\circ + 2\gamma$. When $\delta = s/4$, the maximum angle is $\omega = 148.8^\circ$.



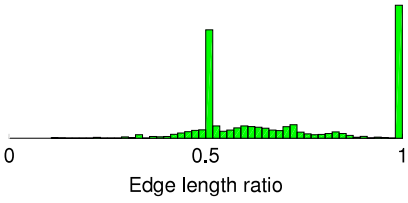
(a) Initial grid and domain.



(b) Final all-quad mesh and a magnified region.

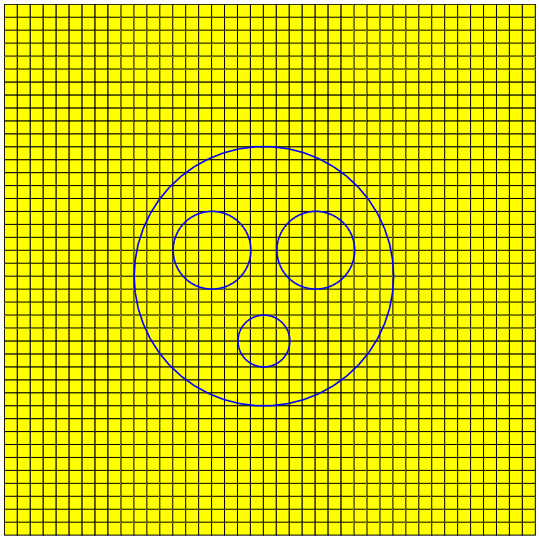


(c) Angle histogram.

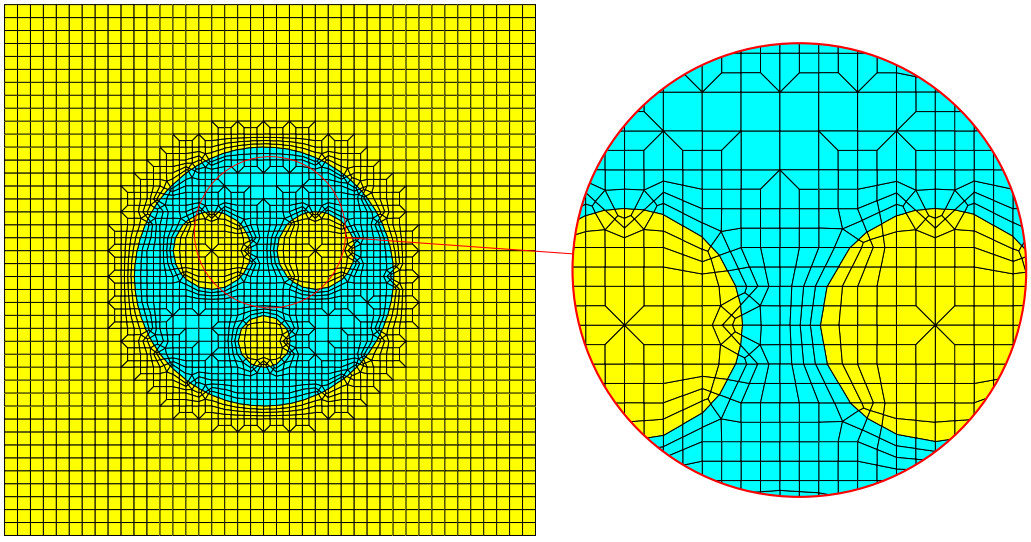


(d) Edge length histogram.

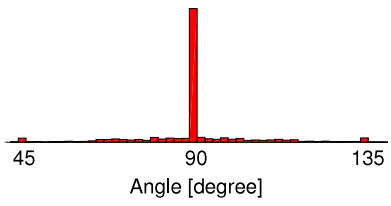
Fig. 15. Application of the all-quad meshing algorithm to a valentine flower shaped domain.



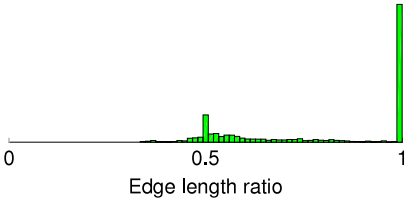
(a) Initial grid and domain.



(b) Final all-quad mesh and a magnified region.



(c) Angle histogram.

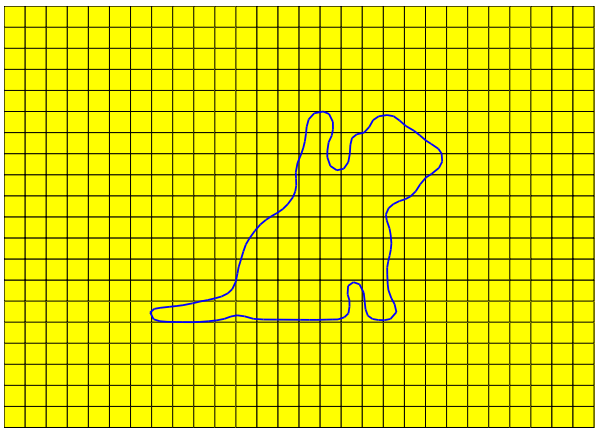


(d) Edge length histogram.

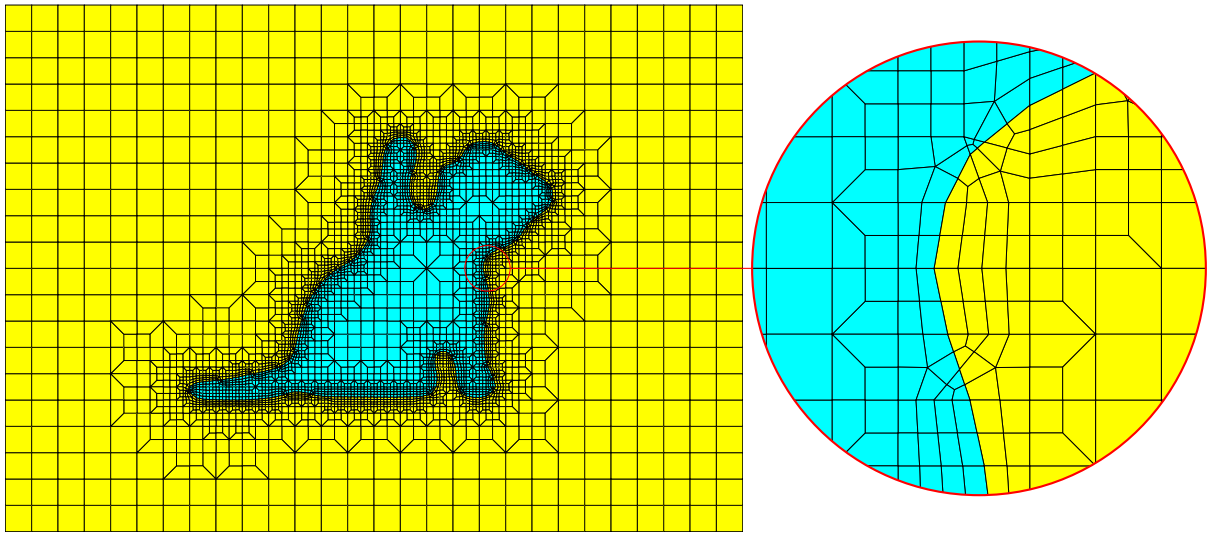
Fig. 16. Application of the all-quad meshing algorithm to a face-shaped domain.

On the other hand, the smallest angle would be formed when a corner, say p, moves outwards away from the square's center a distance δ in the direction $\vec{qp}$, while its neighbor corners t and v move a distance δ towards the square's center in directions such that pt and pv are orthogonal to the displacements of t and v, respectively. In this case, as shown in Fig. 9(c), the horizontal distance between p and t is $|up| = s + \frac{\delta}{\sqrt{2}} - \delta\cos\theta$, while the vertical distance $|ut| = \frac{\delta}{\sqrt{2}} + \delta\sin\theta$. This yields an angle β between pu and pt, where $\tan\beta = |ut|/|up|$. The minimum angle α at corner p is therefore equal to $90^\circ - 2\beta$. When $\delta = s/4$, the minimum angle is $\alpha = 48.67^\circ$.

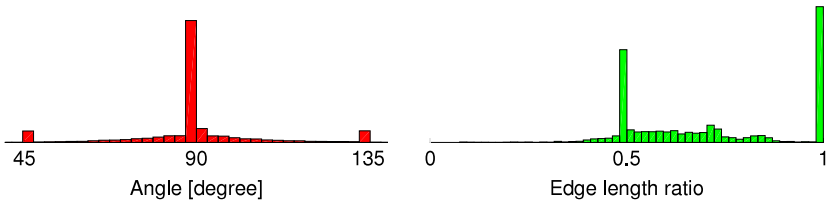
Grid-Aligned Repelling. This repelling approach can be perceived as a special case of the boundary normal repelling that limits the points' movements to the grid lines. Similar to the analysis above, we start with an initial square as shown in Fig. 10(a). The largest angle would be formed when two opposite corners, say v and t move a distance δ outwards from the square on the vertical



(a) Initial grid and domain.



(b) Final all-quad mesh and a magnified region.



(c) Angle histogram. (d) Edge length histogram.

Fig. 17. Application of the all-quad meshing algorithm to a cat-shaped domain.

and horizontal grid lines, respectively, and the corner q moves a distance δ on either grid line. In this case, as shown in Fig. 10(b), the largest angle can be found to be $\omega = 90^\circ + \tan^{-1}(\frac{\delta}{s-\delta}) + \tan^{-1}(2\delta/s)$. When $\delta = s/4$, $\omega = 135^\circ$.

On the other hand, the smallest angle would be formed when two opposite corners, say v and t move a distance δ inwards, on the horizontal and vertical grid lines, respectively, and corner p moves a distance δ on either grid line. In this case, as shown Fig. 10(c), the smallest angle can be found to be $\alpha = 90^\circ - \tan^{-1}(\frac{\delta}{s+\delta}) - \tan^{-1}(2\delta/s)$. When $\delta = s/4$, $\alpha = 52.125^\circ$.

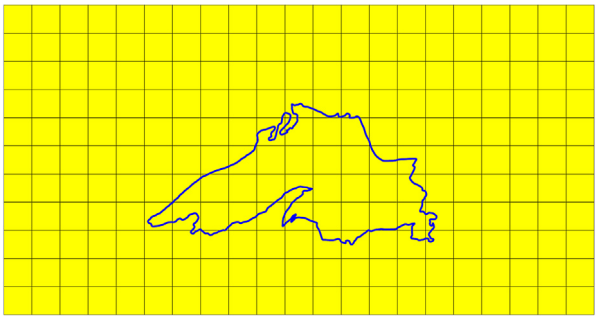
The bounds on the angles at the center node of the templates are better than the above corner-angle bounds. The template for two hanging nodes may split a corner angle. The angle bounds resulting from either repelling approach are then $(48.67^\circ, 148.8^\circ)$ and the edge length bounds are $(0.5, 1.5)s$.

3.3. Squares with geometry

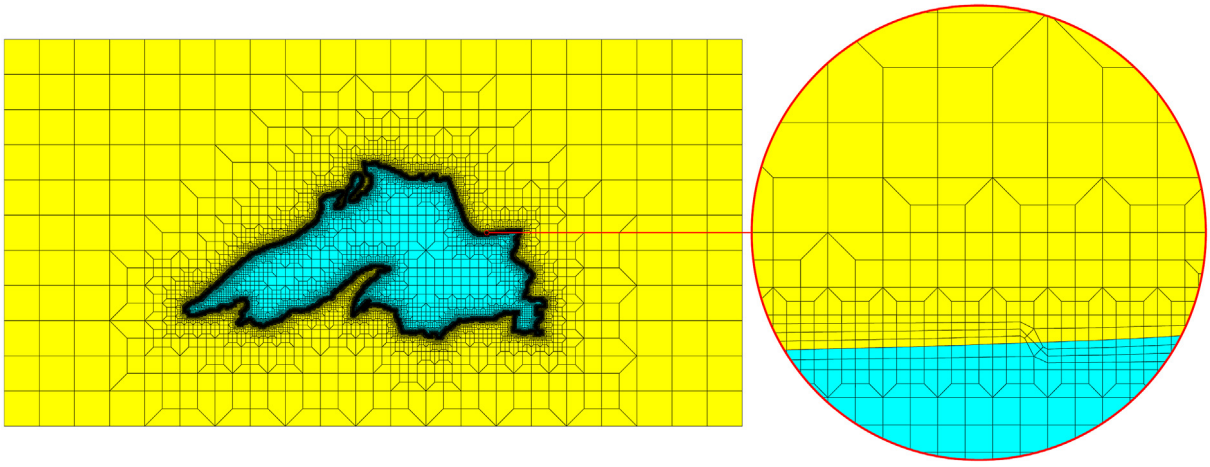
Squares with geometry are divided into 3–6 sided polygons by the curves of the geometry. Some corners may be moved away from the geometry to ensure a δ clearance.

3.3.1. Edge lengths bounds

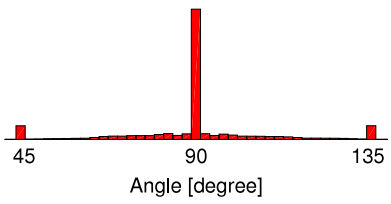
For a square with a vertex, a moved side length is at most $s+2\delta$, bounded by the case of a square corner–corner edge with both



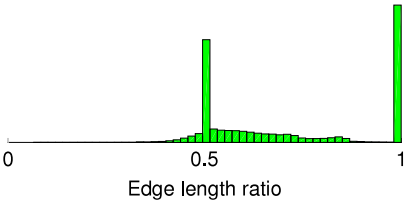
(a) Initial grid and domain.



(b) Final all-quad mesh and a magnified region.



(c) Angle histogram.



(d) Edge length histogram.

Fig. 18. Application of the all-quad meshing algorithm to a lake shaped domain.

nodes moved away from each other by δ. For a square cut only by curves, we consider moving nodes normal to the boundary (which provides upper-bounds for grid-aligned movements as well). Since curves are straight, two adjacent square corners are moved in the same direction. If a single corner is moved, then we have $|e|^2 < s^2 + \delta^2$. An edge along a curve may not be longer than square diagonal, $|e| < \sqrt{2}s$. Any corner–corner distance is at least δ, and hence corner–midpoint edge is at least $\delta/2$. Any vertex center to midpoint edge is at least δ, by construction. We now consider centers that are not vertices. Any polygon center is at least $\delta/3$ from a non-center edge by the following. The worst case is a triangle. Let one edge be the horizontal axis. The other corner must be at least δ from the edge. Since the center is placed at the geometric center, in particular its vertical coordinate is one-third of each of the corners' vertical coordinate, its height above the edge is at least $\delta/3$. To summarize, for $\delta = s/4$,

- corner–midpoint curve edges $|e| \in s[0.125, 0.71]$
- and side edges $|e| \in s[0.125, 0.52]$;
- center–midpoint edges $|e| \geq s/12$.

3.3.2. Square corner angles

The square corner angles are bounded similar to the uncut squares case, where $\alpha = 48.6°$, and $\omega = 148.8°$.

- square corner angles $\in (48.6°, 148.8°)$.

3.3.3. Cut corner angles

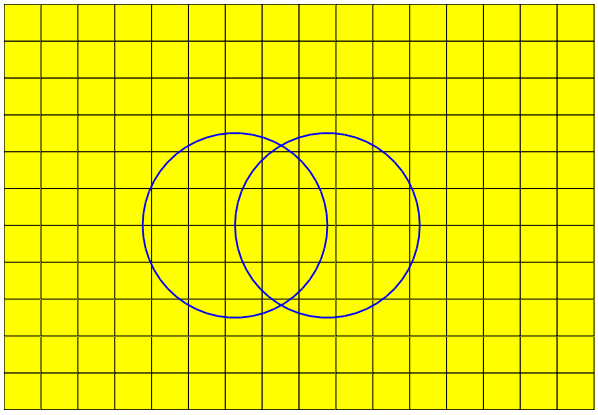
The limiting case for angles at a node where a curve crosses a square side occurs when each corner moves δ away from the curve, and the pre-cut side is at most $e_{\max}$ for sides, $1.03s$. We see that $\sin\alpha = 2\delta/e_{\max}$. Hence for $\delta = \frac{s}{4}$,

- cut corner angles are $\in (29°, 151°)$.

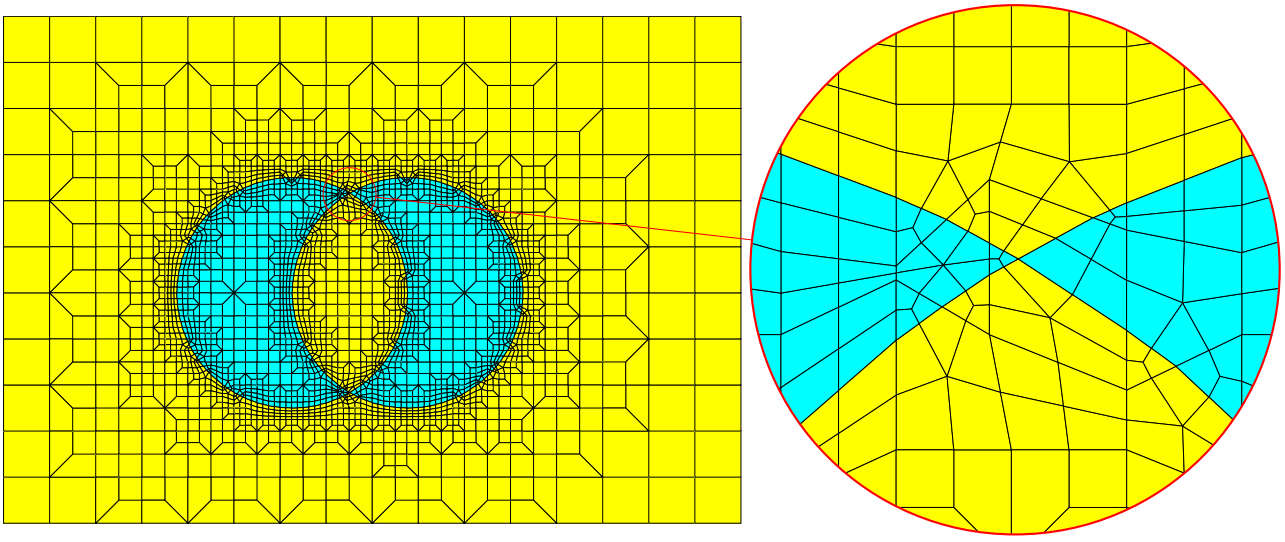
3.3.4. Center and midpoint angles

We consider each case of the number of sides of a polygon.

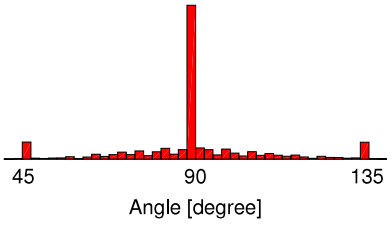
Triangle. We first consider a triangle, that is a curve cutting off a corner of a square. See Fig. 11. The worst case is when K is already δ away from the curve before movement, and the curve is as close to



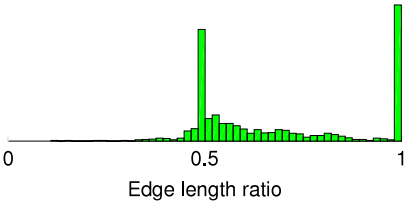
(a) Initial grid and domain.



(b) Final all-quad mesh and a magnified region.



(c) Angle histogram.



(d) Edge length histogram.

Fig. 19. Application of the all-quad meshing algorithm to an intersecting-circles shaped domain.

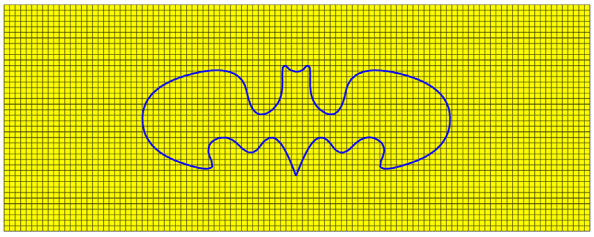
K as possible on one square side, and as far away as possible on the other. We place center c at 1/3 of the way from k_m to K, which is 2/3 of the way to D. (This is the triangle centroid, the average of its corners, and where all three of the corner-to-midpoint lines meet.) From similar triangles we see $\angle a_m b_m k = \angle ABc = B_A$, etc., as labeled in the figure. Hence $\omega_K = 180^\circ - A_B - B_A$, $\omega_A = 180^\circ - C_B - B_C$ and $\omega_B = 180^\circ - A_K - K_A$. Using the cotangent angle relationships for midpoint subdivided triangles (e.g. Section A.3.2 in [36,37]), we have $\cot K_B = \cot B + 2 \cot K$, etc. Given $\{A, B, C\}$, these allow us to derive A_B, etc., and hence the ω's. We consider two extreme cases. The first is isosceles with $|\overline{KB}| = |\overline{KA}|$. Then $\omega_B = \omega_A \approx 108^\circ$, and $\omega_K \approx 143.2^\circ$. The second case is extreme disparity in edge lengths, $|\overline{KB}| = s$ and $|\overline{KA}| = \delta$. (If K was repelled, then $\overline{KA}$ may be longer, but the resultant angles are less extreme.) Then $\omega_A \approx 158.8^\circ$, and $\omega_B \approx 40.6^\circ$ and $\omega_K \approx 160.6^\circ$.

We bound midpoint angles in the same way. We see that the midpoint angles are $\{A+B_A, K+B_K, B+A_B, K+A_K, B+K_b, A+K_A\}$. Again considering the extreme edge lengths, we have $\{\angle a_m, \angle b_m\} \in (26.5^\circ, 153.5^\circ)$. Also $\angle k_m \in (28^\circ, 152^\circ)$.

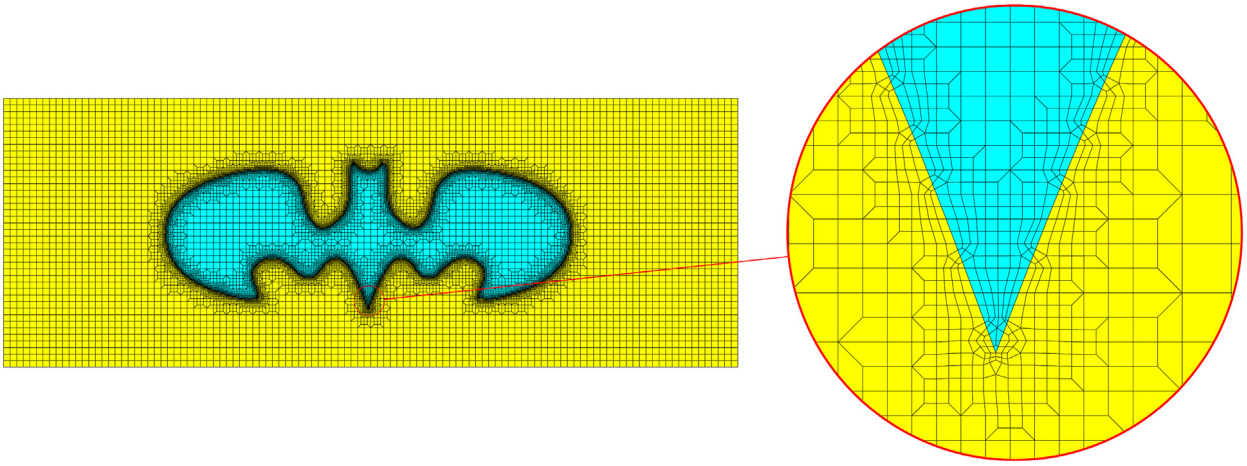
For triangles,

- center angles are in $(40.6^\circ, 160.6^\circ)$.
- midpoint angles are in $(26.5^\circ, 153.5^\circ)$.

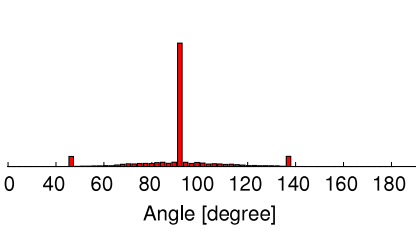
Quadrilateral. This is the case of a square cut by a curve from one side to its opposite. Observe that the center is placed where the two lines connecting opposite midpoints meet. (This is not the



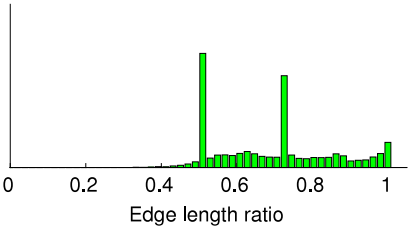
(a) Initial grid and domain.



(b) Final all-quad mesh and a magnified region.



(c) Angle histogram.



(d) Edge length histogram.

Fig. 20. Application of the all-quad meshing algorithm to batman-shaped domain with sharp corners.

center of mass in general, but is the average of the four corners.) The worst angles occur at the midpoint of the curve and the cut corners; see Fig. 13. The smallest angle is γ, and the largest angle is its supplement $\overline{\gamma}$. The worst case is when neither quadrilateral corner was moved, but the curve is δ away from one of the corners, and almost passes through the corner of square on the other side of the curve. In this case the curve has maximum slope 3/4, so $\gamma \geq \arctan(4/3)$, and

- center and midpoint angles $\in (53.1^\circ, 126.9^\circ)$.

Pentagon. Here we consider a square cut by a single curve. One of the square diagonals does not intersect the curve. We place the center c at the centroid of the triangle ΔAKB on the far side of that diagonal from the curve. See Fig. 12.

Consider the corner and midpoint angles. The worst case is when A and B have been repelled δ from the curve, so we must first bound angles A, B, and K. For geometry-normal repelling, Fig. 14(a), we have $K \leq 90 + 2\theta$ where $\theta \leq 12.2^\circ$, or $K < 114.3^\circ$ and $\{\angle BAK, \angle ABK\} > 32.8^\circ$. (Note $\angle BAK$ and $\angle ABK$ are not mesh angles.) Using the cotangent relations, $\omega_K \leq 155.7$, and the midpoint angles along $\overline{KA}$ and $\overline{KB}$ are in $(44.9^\circ, 135.1^\circ)$. (For axis-aligned repelling, we have $K \leq 90^\circ + \theta_2 \leq 104.1^\circ$, and the bounds are subsumed. See Fig. 14(b).)

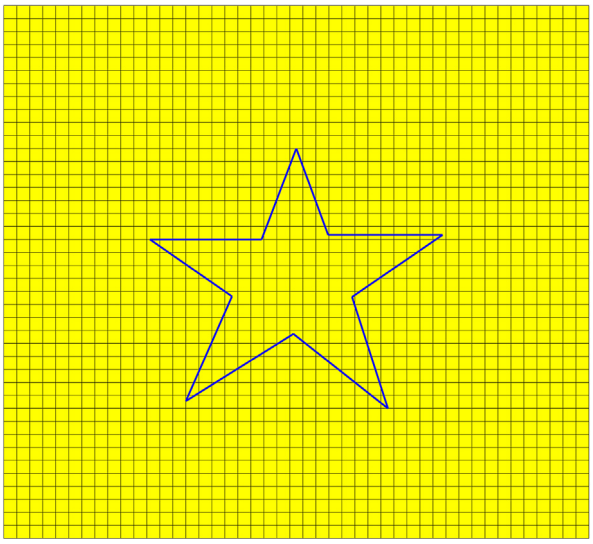
Consider the center angles. For ω_A, the worst case is when there is no repelling. See Fig. 12(a), where $\omega_A < \angle a_m c k_m < 108.5^\circ$. The same holds for ω_B. For the other two central angles, we note that the midpoint of the curve edge must be inside the lower-left quadrant in Fig. 12(b). Hence $\omega < \gamma = \gamma_1 + \gamma_2 < 76^\circ + 41.2^\circ = 117.2^\circ$.

For the minimum center angles, we have $\omega_A \geq A_K > 48.3^\circ$. The same holds for ω_B. For the central angles involving the curve, the worst case is when the curve barely cuts the box corner, see Fig. 12(c). The angle subtends nearly half of a box length, and $\omega > 30.9^\circ$. For the curve and the sides cut by the curve, the midpoint angles are in $45^\circ, 135^\circ$.

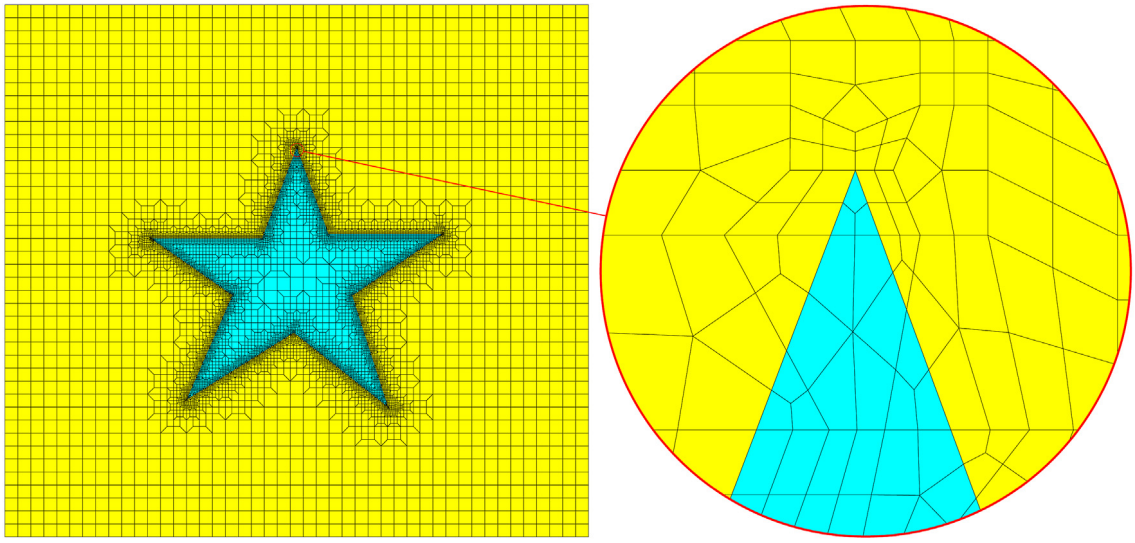
For pentagons,

- center angles are in $(30.9^\circ, 155.7^\circ)$.
- midpoint angles are in $(44.9^\circ, 135.1^\circ)$.

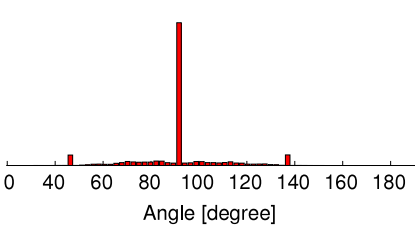
Two or more curves in a square. The regions bounded by one curve and the square have meshes and angle bounds as described in the triangle and square cases. It remains to consider the region between two curves: a square, pentagon, or hexagon. See Fig. 5. These cases only occur near sharp angles, and the obtained bounds depend on the input angles. We do not analyze these explicitly, but observe reasonable quality compared to the element containing the sharp vertex.



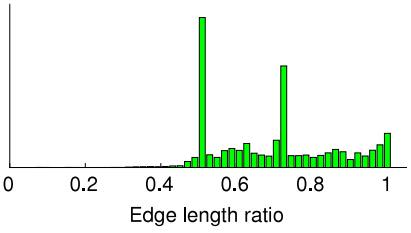
(a) Initial grid and domain.



(b) Final all-quad mesh and a magnified region.



(c) Angle histogram.



(d) Edge length histogram.

Fig. 21. Application of the all-quad meshing algorithm to star-shaped domains with sharp corners.

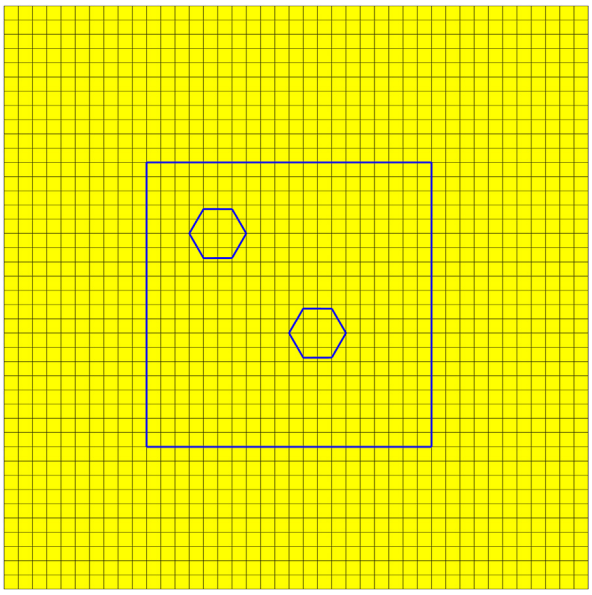
3.4. Quality bound summary

Angles are provably bounded away from 0° and 180°, and edge lengths are within a reasonable constant factor of square side lengths, except perhaps in the presence of input curves that meet at a sharp angle.

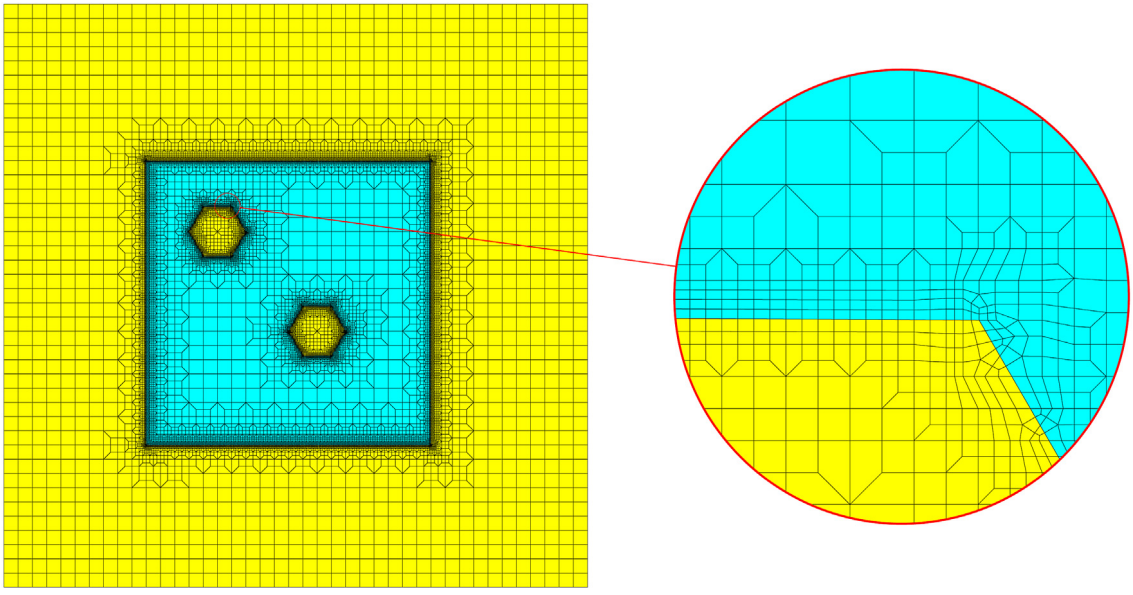
- Mesh angles are in (24.3°, 160.6°).
- Mesh edge lengths are in $s(0.08, 1.5)$.

4. Experimental results

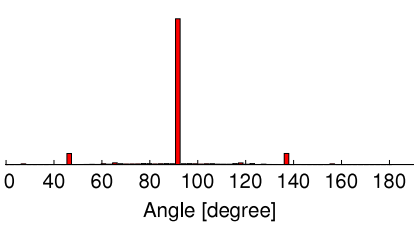
To illustrate the capabilities of our algorithm, we apply it to some domains with smooth and sharp features. Figures 15, 16, 17, 18, and 19 show initial uniform meshes and final all-quad meshes applied to a few smooth domains (flower-, face-, cat-, lake-, and two circles-shaped domains). Our algorithm generates angles that are mostly 90° and are well-bounded between 45° and 135°. On the



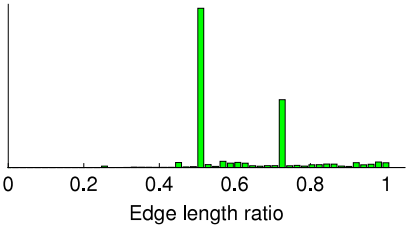
(a) Initial grid and domain.



(b) Final all-quad mesh and a magnified region.



(c) Angle histogram.

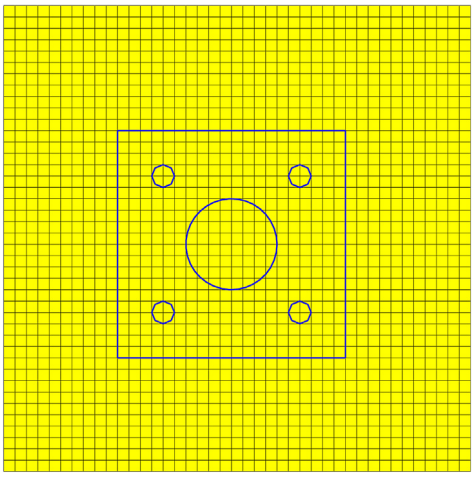


(d) Edge length histogram.

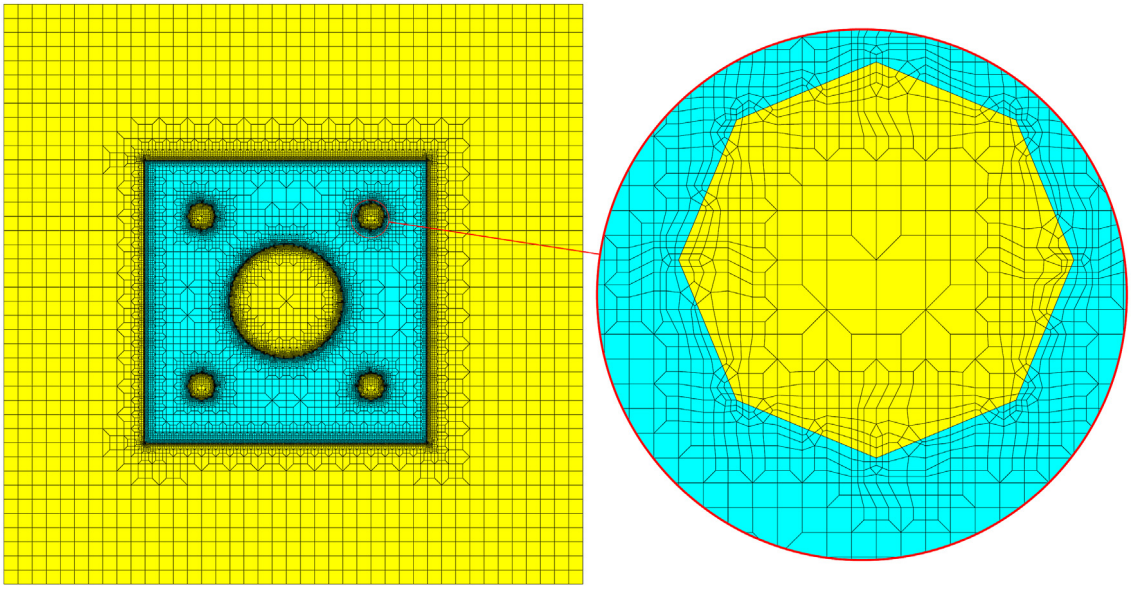
Fig. 22. Application of the all-quad meshing algorithm to a double-hex domain with sharp corners.

other hand, although our analysis focused on smooth geometries, we show some experimental results applying our algorithm to geometries with sharp corners. In specific, Figures 20, 21, 22, and 23 show initial uniform meshes and final all-quad meshes applied to a few domains with connected regions and sharp features (batman-, star-, double hex-, and five holes-shaped domains). Most but not all angle are bounded between 45° and 135° in these cases as shown in the histograms.

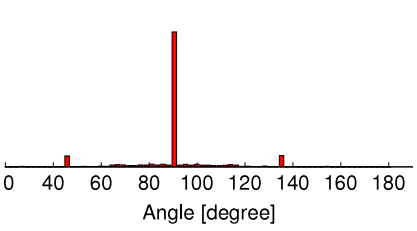
Quality metrics of the resulting meshes are summarized in Table 1, including minimum and maximum angles, and the minimum and average ratios of shortest to longest edge lengths in a quad.



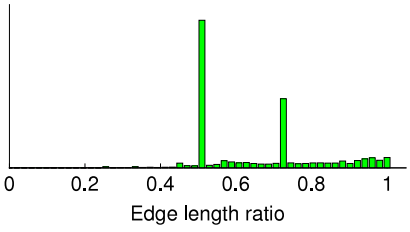
(a) Initial grid and domain.



(b) Final all-quad mesh and a magnified region.



(c) Angle histogram.



(d) Edge length histogram.

Fig. 23. Application of the all-quad meshing algorithm to a domain with sharp corners: square with five eight-sided holes.

5. Conclusions

We described an algorithm for all quad meshing of non-convex domains, with connected regions, conforming to both the interior and exterior of the domain. Our algorithm is robust and provably-correct. It does not require post-processing cleanup operations, such as pillowing, to control the angle bounds of the final mesh, and can easily handle smooth shapes as well as domains with sharp features and corners. Our next steps are to adaptively vary the sizing function along the boundary. We also seek to demonstrate the method on domains with many curves meeting at a single vertex at sharp angles. A natural extension of the presented algorithm is to apply the same steps to *3d all-hex meshing*. This is simply doable because the steps of our algorithm (repelling, splitting, mid-point subdivision, 2-ref template employment) can be employed in 3d in a similar fashion, especially for smooth surfaces. Some cases of sharp features would certainly be more challenging. The extent of the 3d analysis and implementation is outside the scope of this paper and will therefore be addressed in an upcoming publication.

Table 1
Mesh quality examples: v is the number of mesh vertices, E is the number of mesh elements, $\theta_{\min}$ and $\theta_{\max}$ are the minimum and maximum angles, $\alpha_{\min}$ is the minimum ratio of shortest to longest edge length in a quad, and α_{avg} is the average ratio of shortest to longest edges in a quad.

Input domain	v	E	$\theta_{\min}$	$\theta_{\max}$	$\alpha_{\min}$	α_{avg}
Smooth geometries						
flower	4 150	4 130	45°	135°	0.107	0.705
face	3 580	3 496	45°	135°	0.332	0.714
cat	5 466	5 417	45°	135°	0.132	0.709
lake	153 997	153 964	45°	135°	0.064	0.718
2 circles	3 220	3 190	45°	135°	0.108	0.690
Sharp features geometries						
batman	26 908	26 747	31°	145°	0.095	0.866
star	11 511	11 410	27.2°	160.6°	0.083	0.864
double hex	23 389	23 298	22.1°	157.2°	0.127	0.847
5 holes	31 773	31 679	22.5°	169.7°	0.012	0.847

Acknowledgments

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

References

[1] Ho-Le K. Finite element mesh generation methods: a review and classification. Comput Aided Des 1988;20(1):27–38.
[2] Liseikin VD. Grid generation methods. Springer Science & Business Media; 2009.
[3] Hackbusch W. Multi-grid methods and applications, vol. 4. Berlin: Springer-Verlag; 1985.
[4] Lee K. Principles of CAD/CAM/CAE systems. Addison-Wesley Longman Publishing Co., Inc.; 1999.
[5] Thakur A, Banerjee AG, Gupta SK. A survey of CAD model simplification techniques for physics-based simulation applications. Comput-Aided Des 2009;41(2):65–80.
[6] Brewer ML, Diachin LF, Knupp PM, Leurent T, Melander DJ. The mesquite mesh quality improvement toolkit. In: IMR, 2003.
[7] Garimella RV, Shashkov MJ, Knupp PM. Triangular and quadrilateral surface mesh quality optimization using local parametrization. Comput Methods Appl Mech Engrg 2004;193(9):913–28.
[8] Anderson BD, Benzley SE, Owen SJ. Automatic all quadrilateral mesh adaption through refinement and coarsening. In: Proceedings of the 18th international meshing roundtable. Springer; 2009. p. 557–74.
[9] Daniels J, Silva CT, Shepherd J, Cohen E. Quadrilateral mesh simplification. In: ACM transactions on graphics (TOG), vol. 27. ACM; 2008. p. 148.
[10] Peters J, Reif U. The simplest subdivision scheme for smoothing polyhedra. ACM Trans Graph 1997;16(4):420–31.
[11] Prautzsch H, Chen Q. Analyzing midpoint subdivision. Comput Aided Geom Design 2011;28(7):407–19.
[12] Eppstein D. Diamond-kite adaptive quadrilateral meshing. Eng Comput 2014;30(2):223–35.
[13] Owen SJ, Staten ML, Canann SA, Saigal S. Q-morph: an indirect approach to advancing front quad meshing. Internat J Numer Methods Engrg 1999;44(9):1317–40.
[14] Ebeida MS, Karamete K, Mestreau E, Dey S. Q-tran: a new approach to transform triangular meshes into quadrilateral meshes locally. In: Proceedings of the 19th international meshing roundtable. Springer; 2010. p. 23–34.
[15] Blacker TD, Stephenson MB. Paving: A new approach to automated quadrilateral mesh generation. Internat J Numer Methods Engrg 1991;32(4):811–47.
[16] Zhu J, Zienkiewicz O, Hinton E, Wu J. A new approach to the development of automatic quadrilateral mesh generation. Internat J Numer Methods Engrg 1991;32(4):849–66.
[17] White DR, Kinney P. Redesign of the paving algorithm: Robustness enhancements through element by element meshing. In: 6th international meshing roundtable. Citeseer; 1997. p. 323–35.
[18] Baehmann PL, Wittchen SL, Shephard MS, Grice KR, Yerry MA. Robust, geometrically based, automatic two-dimensional mesh generation. Internat J Numer Methods Engrg 1987;24(6):1043–78.
[19] Schneiders R, Schindler R, Weiler F. Octree-based generation of hexahedral element meshes. In: Proceedings of the 5th international meshing roundtable. Citeseer; 1996.
[20] Liang X, Ebeida MS, Zhang Y. Guaranteed-quality all-quadrilateral mesh generation with feature preservation. Comput Methods Appl Mech Engrg 2010;199(29):2072–83.
[21] Liang X, Zhang Y. Hexagon-based all-quadrilateral mesh generation with guaranteed angle bounds. Comput Methods Appl Mech Engrg 2011;200(23):2005–20.
[22] Liang X, Zhang Y. Matching interior and exterior all-quadrilateral meshes with guaranteed angle bounds. Eng Comput 2012;28(4):375–89.
[23] Shimada K, Liao J-H, Itoh T. et al. Quadrilateral meshing with directionality control through the packing of square cells. In: IMR, 1998, p. 61–75.
[24] Bern M, Eppstein D. Quadrilateral meshing by circle packing. Int J Comput Geom Appl 2000;10(04):347–60.
[25] Atalay FB, Ramaswami S, Xu D. Quadrilateral meshes with bounded minimum angle. In: Proceedings of the 17th international meshing roundtable. Springer; 2008. p. 73–91.
[26] Fisher M, Schröder P, Desbrun M, Hoppe H. Design of tangent vector fields. In: ACM transactions on graphics (TOG), vol. 26. ACM; 2007. p. 56.
[27] Cohen-Steiner D, Morvan J-M. Restricted delaunay triangulations and normal cycle. In: Proceedings of the nineteenth annual symposium on computational geometry. ACM; 2003. p. 312–21.
[28] Cazals F, Pouget M. Estimating differential quantities using polynomial fitting of osculating jets. Comput Aided Geom Design 2005;22(2):121–46.
[29] Kälberer F, Nieser M, Polthier K. Quadcover-surface parameterization using branched coverings. In: Computer graphics forum, vol. 26. Wiley Online Library; 2007. p. 375–84.
[30] Rushdi AA, Mitchell SA, Bajaj CL, Ebeida MS. Robust all-quad meshing of domains with connected regions. Procedia Eng 2015;124:96–108. (Proceedings of the 24th International Meshing Roundtable).
[31] Ju T, Losasso F, Schaefer S, Warren J. Dual contouring of hermite data. ACM Trans Graph 2002;21(3):339–46.
[32] Zhang Y, Qian J. Dual contouring for domains with topology ambiguity. Comput Methods Appl Mech Engrg 2012;217:34–45.
[33] Zhang Y, Bajaj C. Finite element meshing for cardiac analysis, Univ. of Texas at Austin: ICES Technical Report.
[34] Mitchell SA, Tautges TJ. Pillowing doublets: Refining a mesh to ensure that faces share at most one edge. In: 4th international meshing roundtable, 1995, p. 231–40.
[35] Ebeida MS, Patney A, Owens JD, Mestreau E. Isotropic conforming refinement of quadrilateral and hexahedral meshes using two-refinement templates. Internat J Numer Methods Engrg 2011;88(10):974–85.
[36] Mitchell SA, Mohammed MA, Mahmoud AH, Ebeida MS. Delaunay quadrangulation by two-coloring vertices. Procedia Eng 2014;8:364–76. proceedings of the 23rd International Meshing Roundtable. Freely available online at Science Direct.
[37] Mitchell SA, Mohammed MA, Mahmoud AH, Ebeida MS. Delaunay quadrangulation by two-coloring vertices — extended version with quad-quality proofs appendix, Tech. Rep. SAND2014-16625C. Sandia National Laboratories; 2014.