Quadrilateral Meshes with Bounded Minimum Angle

F. Betul Atalay[1], Suneeta Ramaswami[2,⋆], and Dianna Xu[3]

[1] St. Joseph's University, Philadelphia, PA
fatalay@sju.edu
[2] Rutgers University, Camden, NJ
rsuneeta@camden.rutgers.edu
[3] Bryn Mawr College, Bryn Mawr, PA
dxu@cs.brynmawr.edu

Summary. This paper presents an algorithm that utilizes a quadtree to construct a strictly convex quadrilateral mesh for a simple polygonal region in which no newly created angle is smaller than $18.43^\circ (= \arctan(\frac{1}{3}))$. This is the first known result, to the best of our knowledge, on quadrilateral mesh generation with a provable guarantee on the minimum angle.

1 Introduction

The generation of quadrilateral meshes with provable guarantees on mesh quality poses several interesting open questions. While theoretical properties of triangle meshes are well understood [4, 8, 9, 7, 11, 12, 15, 14], much less is known about algorithms for *provably good* quadrilateral meshes. Analysts, however, prefer quadrilateral and hexahedral meshes for better solution quality in numerous applications [1, 2, 6, 10, 16]. This is because they have better convergence properties, and hence lower approximation errors, in finite element methods for solutions to systems of partial differential equations. Quadrilateral meshes also offer lower mesh complexity, and better directionality control for anisotropic meshing. For stable analytical results, however, it is critical to construct meshes with certain quality guarantees. Specifically, algorithms that construct well-shaped elements by providing bounds on minimum and maximum angles have much practical value. Techniques such as paving [5] work well in practice, but do not give provable angle guarantees. Circle-packing techniques have been used to construct quadrangulations with no angles larger than 120° for polygon interiors [3], but with no bound on smallest angle. An algorithm to construct linear-sized strictly convex quadrilateral meshes for arbitrary planar straight line graphs is given in [13].

Our contribution. In this paper, we present a new algorithm to generate quadrilateral meshes for simple polygonal regions, possibly with holes, with a provable guarantee on the minimum angle. We use quadtrees to show that no newly created angle in the quadrilateral mesh is smaller than 18.43°. The quadrilaterals are strictly convex, i.e., the maximum angle is strictly less than 180°. This is the first known quadrilateral mesh generation algorithm with a provable bound on the *minimum* angle. (Quadtrees have been used to give triangular meshes without small angles for point sets and polygons

⋆ Research partially supported by NSF Grant 0204293.

in 2D [4], and octrees have been utilized to construct tetrahedral meshes with bounded aspect-ratio elements for polyhedra [12].)

In Section 2, we use quadtrees to construct a quadrilateral mesh for a point set in which the minimum angle is bounded below by $45° - \arctan(\frac{1}{3}) = 26.57°$. We then describe in Section 3 an algorithm that adapts the guaranteed-quality mesh of polygon vertices to polygon edges to construct a quadrilateral mesh for the interior of a simple polygon (possibly with holes) in which new angles (angles other than those determined by the input) are bounded below by $\arctan(\frac{1}{3}) = 18.43°$.

Throughout this paper, we use the shorter terms "quadrangulate" and "quadrangulation" instead of "quadrilateralize" and "quadrilateralization". We also sometimes use the word "quad" for quadrilateral. *Steiner points* are additional points, other than those provided by the input, inserted during the mesh generation process.

2 Point Set Mesh with Bounded Minimum Angle

We first describe an algorithm to construct a quadrilateral mesh with a minimum angle bound of $26.57°$ for a given point set X.

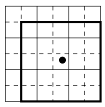
2.1 Construction of the Quadtree

Given a point set X, we construct a quadtree for X with the following separation and balancing conditions. These conditions are similar to those in [4], but adapted to particular requirements for quadrilateral (rather than triangle) meshing.

A. Split a cell C (with side length l) containing at least one point if it is crowded. A cell is crowded if one or more of the following conditions hold:
   1. it contains more than one point from X.
   2. one of the extended neighbors is split (an extended neighbor is a cell of same size sharing either a side or corner of C).
   3. it contains a point with a nearest neighbor less than $2\sqrt{2}l$ units away.

B. When a crowded cell C is split, split those extended neighbors of C that share an edge or corner with a child of C containing an original point in X.

C. The final quadtree is balanced so that the edge lengths of two adjacent cells differ at most by a factor of 2 (neighbors of C with side length l have length $l/2$ or $2l$).



Observe that in a quadtree with the above separation and balancing conditions, a cell containing a point from X is guaranteed to be surrounded by 8 empty cells of the same size. We refine the quadtree decomposition further to do the following: Split each of these eight empty quadtree cells into 2×2 cells and rebalance the quadtree. This converts the original 3×3 grid around every point $p \in X$ into a 6×6 grid. Furthermore, now p lies at the center of a 5×5 equal-sized grid (outlined in bold in figure), and is surrounded by twenty-four empty quadtree cells of the same size. There are two reasons for this refinement step:

1. The final step of our algorithm to construct a quadrilateral mesh for X consists of warping a Steiner point in the mesh to an original point $p \in X$ (Section 2.4). This step is simplified considerably due to the refinement.

2. The algorithm to construct a quadrilateral mesh for non-acute polygons (Section 3.1) uses the 5×5 grid to mesh the region around polygon vertices.

We construct a quadrilateral mesh with bounded minimum angle for X by placing Steiner points in the interior of the quadtree cells. The placement of the Steiner points is determined by identifying and applying templates to the quadtree decomposition. A leaf of the quadtree is an unsplit cell and we refer to these as *1-cells* in our discussion. A template is applied to each internal node of the quadtree.

2.2 The Templates

A template is labeled by the number of children of a quadtree node that are 1-cells. Hence we have 6 template configurations, for nodes with zero ($\mathcal{T}^{(0)}$), one ($\mathcal{T}^{(1)}$), two, three ($\mathcal{T}^{(3)}$) or four ($\mathcal{T}^{(4)}$) 1-cell children. Nodes with two 1-cell children have two layouts, $\mathcal{T}^{(2a)}$ and $\mathcal{T}^{(2b)}$.

Templates at the deepest level of subdivision. The templates at the deepest level of subdivision are shown in Fig. 1. Note that, all other possible configurations are symmetric to the depicted ones. In order to quadrangulate a template, first, a Steiner point is placed at the center of each quadtree cell. These points are denoted with full circles. We then place *extra* Steiner points, which are denoted by empty circles in the figure, for one of two reasons: (i) In $\mathcal{T}^{(1)}$, the top-left extra point and in $\mathcal{T}^{(2b)}$ the middle extra point are added to be able to quadrangulate *properly* within the template. (ii) The remaining extra points are added in the 1-cells, halfway on the diagonal between the center Steiner point and the outer cell corner. The reason for adding the second type of Steiner points is that after an internal node is quadrangulated, it will provide a polygonal chain with an even number of points (we will call them *even-connector chains*) to which its neighbors can connect.

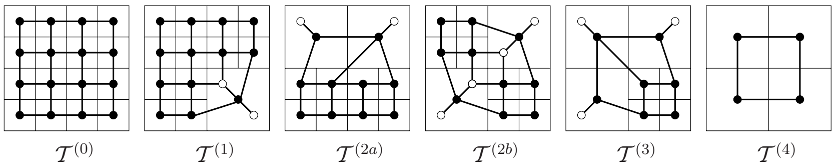


Fig. 1. Templates at the deepest level of the subdivision

General Templates. Our recursive algorithm applies templates to all internal nodes starting with the deepest ones. We generalize the templates to apply to an arbitrarily deep internal node as shown in Fig. 2. In general, when a template is applied to an internal node, its children which are not 1-cells have already had templates applied to them. Each such child has been quadrangulated internally and provides even-connector chains on all four sides. The corresponding endpoints of two neighboring chains are then connected to construct a polygon with guaranteed even number of vertices which can therefore be quadrangulated. We name this process "stitching", illustrated by the cross-hatched regions in Fig. 2. The processed internal nodes are

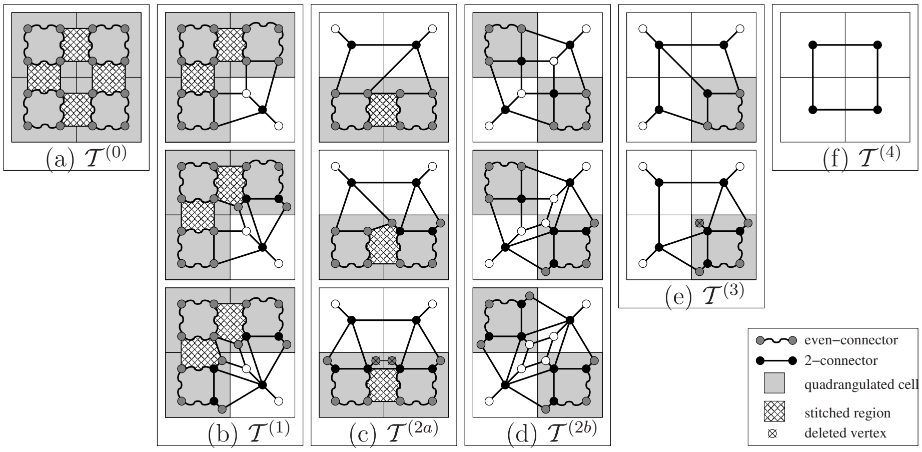
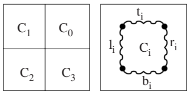


Fig. 2. General templates at arbitrary level of subdivision

depicted as blackboxes with even-connector chains at each side. Note that the placement of a chain's endpoint does not necessarily correspond to the exact location of the endpoint within the actual cell, due to the existence of type (ii) Steiner points.



Labeling the chains. Children quadrants of a cell are labeled C_0, C_1, C_2, and C_3 in counterclockwise order starting from the northeast quadrant. The four chains surrounding a processed quadrant C_i are labeled l_i, r_i, t_i and b_i (see figure).

2.3 The Algorithm

The recursive procedure *applyTemplate* that applies a template to an internal node is presented in the code block given in Fig. 3. It is initially called with the root node of the quadtree. Note that the algorithm is presented only with respect to the depicted configurations of the templates. Symmetric configurations are handled similarly.

Stitching Chains. Procedure *stitchChains* connects the four endpoints of two neighboring even-connector chains and quadrangulates the resulting polygon. Note that such a polygon is guaranteed to have an even number of vertices on the boundary. The algorithm is illustrated in Fig 4. Procedure *stitchChains* is only called if the current template is of type $\mathcal{T}^{(0)}$, $\mathcal{T}^{(1)}$ or $\mathcal{T}^{(2a)}$. The action of this procedure is also illustrated by the crosshatched areas in Fig. 2(a), (b) and (c).

The quadrangulation process divides the chains into half chains, each of which spans the corresponding edge of a child quadrant. These half chains are then recursively

```
applyTemplate($QuadtreeNode\ N$)
  templateType $\leftarrow$ whichTemplate($N$)
  for $C_i \in$ children($N$)
    if $C_i$ is not a 1-cell
      $(l_i, r_i, t_i, b_i) \leftarrow$ applyTemplate($C_i$)
    else
      construct $(l_i, r_i, t_i, b_i)$ for $C_i$.
  switch (templateType)
    case $\mathcal{T}^{(0)}$:
      stitchChains($l_0, r_1$), stitchChains($b_1, t_2$)
      stitchChains($r_2, l_3$), stitchChains($t_3, b_0$)
    case $\mathcal{T}^{(1)}$:
      stitchChains($l_0, r_1$), stitchChains($b_1, t_2$)
      Place Steiner points and quadrangulate per Fig. 2(b).
    case $\mathcal{T}^{(2a)}$:
      stitchChains($r_2, l_3$)
      Place Steiner points and quadrangulate per Fig. 2(c).
    case $\mathcal{T}^{(2b)}$:
      Place Steiner points and quadrangulate per Fig. 2(d).
    case $\mathcal{T}^{(3)}$:
      Place Steiner points and quadrangulate per Fig. 2(e).
    case $\mathcal{T}^{(4)}$:
      Place Steiner points and quadrangulate per Fig. 2(f).
  return $(l_1 + l_2, r_0 + r_3, t_1 + t_0, b_2 + b_3)$
```

Fig. 3. *applyTemplate* quadrangulates node N

```
stitchChains($Chain\ ch_1, Chain\ ch_2$)
  switch (length($ch_1$), length($ch_2$))
    case $(2-2), (2-4), (4-2)$:
    Apply appropriate base case from Fig. 5.
    case $(2-6), (2-8)$):
    Apply appropriate base case from Fig. 6.
    default:
      $(f_1, s_1) \leftarrow$ getHalfChains($ch_1$)
      $(f_2, s_2) \leftarrow$ getHalfChains($ch_2$)
      stitchChains($f_1, f_2$)
      stitchChains($s_1, s_2$)
```

Fig. 4. *stitchChains* stitches two even-connector chains, one from each of the two neighbor cells sharing an edge

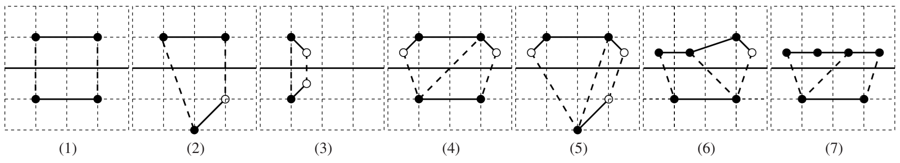


Fig. 5. Stitching 2-2 and 2-4 or 4-2 connector chains

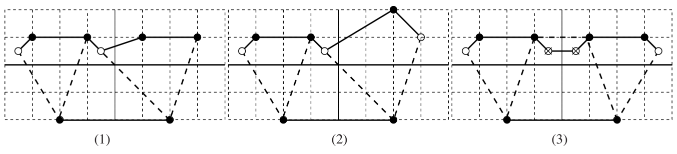


Fig. 6. Stitching 2-6 and 2-8 connector chains

stitched. Although the even-connector chains can be arbitrarily long, at the base case there are only four types of chains: chains with 2, 4, 6 or 8 connectors. Figs. 5-6 illustrate how the base-case chains are stitched (the stitching edges are dotted). Symmetric cases are not listed in the illustrations.

2.4 Angle Bounds

Minimum Angle. We analyze the minimum angle bound in each of *applyTemplate*, the base case of *stitchChains*, and the recursive step of *stitchChains*.

General templates: By construction, the minimum angle appears in templates $\mathcal{T}^{(1)}$ and $\mathcal{T}^{(2b)}$ and equals $45° - \arctan(\frac{1}{3}) = 26.57°$ (illustrated in Fig. 7).

Stitching base case: The base cases of stitching generate the same minimum angle of $45° - \arctan(\frac{1}{3}) = 26.57°$ which can be found in in Fig. 5-(5).

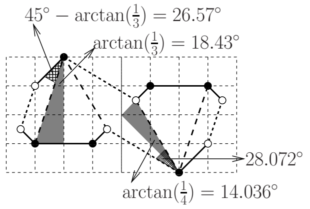

$45° - \arctan(\frac{1}{3}) = 26.57°$
$\arctan(\frac{1}{3}) = 18.43°$
28.072°
$\arctan(\frac{1}{4}) = 14.036°$

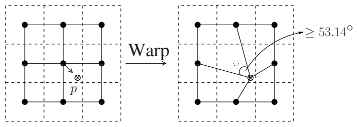
Fig. 7. Minimum angle bounds

Stitching merging step: After the corresponding half-chains are stitched in the recursive step of *stitchChains*, a middle quad is formed by the four end points of the stitched half-chains. This middle quad gives a minimum angle of $2 \times \arctan(\frac{1}{4}) = 28.07°$. See Fig. 7. Recall that these four points are by construction on the two diagonals that cross at the center of four quadtree quadrants. Furthermore, they are either at the center of the quadtree quadrant, or halfway down the diagonal from the center. The worst-case configuration is illustrated in Fig. 7. This results from connecting any Fig. 5-(5) connector chain with an inverted version of itself.

Degenerate quads. In the stitching cases illustrated by Fig. 5-(7), Fig. 6-(1), Fig. 6-(2) as well as template $\mathcal{T}^{(2a)}$ (Fig. 1), there are degenerate quads with two edges on a straight line. In all cases, the $180°$ vertex is connected to a third vertex on the other side of the degenerate quad, by construction. This allows perturbation of the degenerate vertex along the third edge, which reduces the $180°$ angle and increases the other two, thus eliminating the degenerate quad.

Maximum Angle. The quadrilaterals generated by our algorithms (including the one in Section 3) are *strictly* convex; i.e., the maximum angle is bounded away from $180°$. The perturbation used to handle degenerate quads (above) implies the maximum angle is less than $(180 - \epsilon)$ for some $\epsilon > 0$. We conjecture the value of ϵ can be bounded from below, but do not explicitly address that question in this paper.

Warping to Original Points. After the construction of the quadrilateral mesh using quadtree cell centers and extra points as Steiner points, we warp certain mesh vertices to the original points from the input point set X. Recall that the quadtree splitting rules of Section 2.1 ensure that the quadtree cell containing an original point $p \in X$ is surrounded by twenty-four empty quadtree cells of the same size. Moreover, the eight empty cells immediately surrounding p do not contain any extra points. Therefore, the warping step simply consists of translating the Steiner point in p's cell to p, along with all the incident edges.



The worst-case minimum angle arising from these nine cells after the warping step is $2 \times 26.57 = 53.14°$. In summary, we have shown the following result:

Theorem 1. *Given a quadtree decomposition with N quadtree cells satisfying the point set separation conditions for a point set X, applyTemplate constructs a mesh for X with at most $3N$ quadrilaterals in which every angle is at least 26.57 degrees.*

Observe that the value of N in the above theorem depends on the geometry of the point set as well as the size of the point set. Due to the point set separation conditions, which are derived from [4] and as was shown there, the size of the quadtree decomposition increases as the distance between the closest pair of points decreases.

3 Polygon Mesh with Bounded Minimum Angle

Given a simple polygon P, possibly with holes, with vertex set X, we give an algorithm to construct a quadrilateral mesh for P and its interior in which no new angle is larger than $18.43°$. The basic idea behind the algorithm is to first construct a guaranteed-quality mesh for X as described in the previous section, and then adapt this mesh to incorporate the edges of P. We use δP to refer to the polygon boundary, and P to refer to the union of the boundary as well as interior.

We describe in Section 3.1 a provably good algorithm to construct a quadrilateral mesh with bounded minimum angle for a simple polygon P in which all interior angles are non-acute (i.e., greater than or equal to $90°$). In Section 3.2, we describe how to handle acute angles.

3.1 Non-acute Simple Polygons

Let P be a non-acute polygon with vertex set X and edges oriented counter-clockwise about the boundary. Let $\mathcal{QT}$ be a quadtree decomposition of X satisfying the point set separation conditions of Section 2.1. Let $\mathcal{Q}$ be a quadrilateral mesh for X with minimum angle $26.57°$, as guaranteed by Theorem 1. In this section, we describe a method to adapt $\mathcal{Q}$ to δP to create a *constrained* quadrilateral mesh for P. In a constrained quadrilateral mesh, we allow Steiner points to be inserted on δP as well, so that the union of the finite elements of the mesh is equal to P.

We start by describing an algorithm to adapt $\mathcal{Q}$ to include a single edge of P. In order to use this algorithm on all edges of P, $\mathcal{QT}$ must satisfy certain polygon edge separation conditions, which are discussed towards the end of the section. We conclude the section by describing how to construct the final constrained mesh for P by adapting to the regions around the vertices.

Inserting an edge into $\mathcal{Q}$

Consider an edge $\vec{e} = (a, b)$ of P oriented from a to b, where $a, b \in X$. Assume that $\vec{e}$ makes an angle between $-45°$ and $45°$ with the positive x axis (if not, orient the x axis so that this is the case). We say that a point lies "above" $\vec{e}$ if it lies in the open halfspace to the left of the oriented line through $\vec{e}$. We use $\vec{e}$ to define two chains of edges from $\mathcal{Q}$ and $\mathcal{QT}$, as described below:

(i) $\vec{e}$ intersects quadrilaterals of $\mathcal{Q}$. Edges of these quadrilaterals are used to define a chain of edges called the *quadrangulation chain* α associated with $\vec{e}$.
(ii) $\vec{e}$ intersects quadtree cells of $\mathcal{QT}$. The centers of these cells are used to define a chain of edges called the *quadtree chain* β associated with $\vec{e}$.

Quadrangulation Chain. Let $q_1, q_2, \ldots, q_k$ be the quadrilaterals of $\mathcal{Q}$ intersected by $\vec{e}$ in left to right order as traversed from a to b (since the quadrilaterals are convex, each q_i is unique). Let E_i be the edges of q_i that lie entirely above $\vec{e}$. E_i may have 0, 1, or 2 edges. If E_i has two edges, they are listed in clockwise order about q_i. Then the *quadrangulation chain* α is defined as $\alpha = E_1 \cdot E_2 \ldots \cdot E_k$, where $\cdot$ represents edge concatenation. See Fig. 8 for an example of a quadrangulation chain, in which E_1 has 1 edge, E_2 has 2 edges, and E_3 has 0 edges. Note that the same edge may repeat twice in α (the repetitions always appear consecutively) and such an edge is incident to q_i that has $|E_i| = 0$. For example, the quadrangulation chain in Fig. 8 has three repeating edges, which are incident to the quadrilaterals q_3, q_5 and q_{10}. If we drop the repetitions from α, then α is a weakly simple polygonal chain.

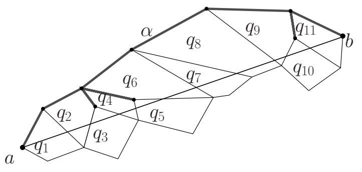


Fig. 8. Quadrangulation chain α

A vertex belongs to an edge if it is one of the endpoints of the edge. We say that $v \in \alpha$ if v is a vertex of $\mathcal{Q}$ and belongs to one of the edges of α. If we quadrangulate the region bounded by α and $\vec{e}$ by adding Steiner points either in the interior of the region or on $\vec{e}$ itself, the resulting quadrangulation is compatible with $\mathcal{Q}$ (since edges of α are edges in $\mathcal{Q}$). However, in order to quadrangulate the region with the desired angle bounds, we need to know more about the geometry of α. The quadtree chain, described below, allows us to establish the required geometric properties for α.

Quadtree Chain. In the remainder of the paper, we use the same symbol to refer to a quadtree cell as well as its center whenever the meaning is clear from the context. Given a cell c, $N(c), W(c)$, and $E(c)$ denote, respectively, the set of north, west, and east neighbor cells of c (note that each set has at most two elements in it because of the balancing conditions for $\mathcal{QT}$).

Let C be the set of cell centers of quadtree cells in $\mathcal{QT}$ that are intersected by $\vec{e}$. C does not include the starting or ending cells (i.e., the cells containing a and b, respectively). Let θ be the angle (in degrees) that $\vec{e}$ makes with the positive x axis. The *quadtree chain* β is defined as follows:

1. If $c \in C$ and c lies above $\vec{e}$, then c belongs to β.
2. If $c \in C$ and c lies below $\vec{e}$, then $N(c) \subset \beta$. Note that the centers of cells in $N(c)$ must lie above $\vec{e}$ under our assumption that $-45 \leq \theta \leq 45$.
3. If $c \in C$, c lies below $\vec{e}$, and $0 \leq \theta \leq 45$ (resp., $-45 \leq \theta < 0$), then a cell center in $W(c)$ (resp., $E(c)$) belongs to β if it lies above $\vec{e}$.

Let $\{c_1, c_2, \ldots, c_m\}$ be the cell centers in β in lexicographically sorted (by x, then y) order. Recall that $\mathcal{Q}$ is constructed from the quadtree decomposition $\mathcal{QT}$ of X. The overall approach to incorporating edge $\vec{e}$ into $\mathcal{Q}$ is summarized below:

(A) We first show that quadtree chain β is a subset of quadrangulation chain α.
(B) This fact allows us, in turn, to exploit the structure provided by $\mathcal{QT}$ and our algorithm from Section 2 to identify a small number of possible ways in which two consecutive points c_i and c_{i+1} of β can be connected along the chain α. We use α_i, $1 \leq i \leq m-1$, to refer to the subchain of α starting at c_i and ending at c_{i+1}. α_i may lie under $\overrightarrow{c_i c_{i+1}}$. In this case, we choose instead a chain of edges in $\mathcal{Q}$ lying above $\overrightarrow{c_i c_{i+1}}$ in order to simplify the final quadrangulation step in part (C) below.
(C) Finally, we quadrangulate the region bounded by $\vec{e}$ and α by breaking it into smaller sub-regions defined by perpendicular projections from c_i and c_{i+1} onto $\vec{e}$. The case analysis form part (B) is then used to prove a minimum angle guarantee of $18.43°$ for the quadrangulation of each subregion.

Lemmas 1-3 are required for steps (A)-(C) and stated here without proof.

Lemma 1. *For $1 \leq i \leq m, c_i \in \alpha$. That is, every cell center in the quadtree chain belongs to the quadrangulation chain.*

Lemma 2. *For $1 \leq i \leq m-1$, c_i and c_{i+1} are edge or corner neighbors in $\mathcal{QT}$.*

Lemma 3. *Let v_i be the vertical projection of c_i on $\vec{e}$. For $1 \leq i \leq m$, the segment $\overline{c_i v_i}$ does not intersect α.*

Lemmas 1 and 3 imply that the edge sequence $(v_i, c_i) \cdot \alpha_i \cdot (c_{i+1}, v_{i+1}) \cdot (v_{i+1}, v_i)$ defines a simple polygon for all $1 \leq i \leq m-1$. Call this polygon A_i. We now use Lemma 2 to prove that α_i is composed of at most four edges. This is done via a case analysis on the ways in which c_i and c_{i+1} are connected in $\mathcal{Q}$.

Lemma 4. *The number of edges in α_i is at most four.*

Proof: We know from Lemma 2 that c_i and c_{i+1} are either edge or corner neighbors in $\mathcal{QT}$. We consider each case separately. Our case analysis only depicts α_i with two or more edges (i.e., when c_i and c_{i+1} are not directly connected). Let $s_i, 1 \leq i \leq m$ refer to the size of c_i's cell (by "size", we mean "side length").

Case 1: c_i and c_{i+1} are edge neighbors. In this case, the connectivity between c_i and c_{i+1} in $\mathcal{Q}$ may come from either the application of a template (*applyTemplate*) at some level of recursion, or the application of the stitching step (*stitchChains*) at some level of recursion. We consider different possibilities based on the ratio $s_i : s_{i+1}$, which may be $1:1, 1:2$, or $2:1$. Configurations for these cases are shown in Figs. 9, 10, and 11, respectively. Each of these figures indicates the minimum internal angle in A_i along α_i. Note that each of them is well above $18.43°$. We depict only distinct α_i that differ in either the number of edges, or the angles at the vertices (that is, we do not show other, symmetric configurations that lead to the same α_i).

Case 2: c_i and c_{i+1} are corner neighbors. In this case, the connectivity between c_i and c_{i+1} in $\mathcal{Q}$ may come from the application of *applyTemplate*, the application of *stitchChains*, or through a *center quad*. The center quad is the quadrilateral formed at the center, i.e. the meeting point of the four quadrants, after a general template (ref. Fig. 3) is applied during the recursive step. Since c_i and c_{i+1} are corner neighbors, the

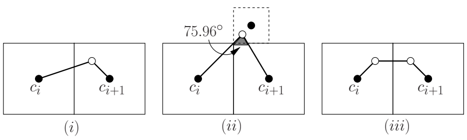


Fig. 9. Configurations for α_i when $s_i : s_{i+1}$ is $1 : 1$. (i) and (ii) come from stitching base cases, and (iii) from stitching merge steps.

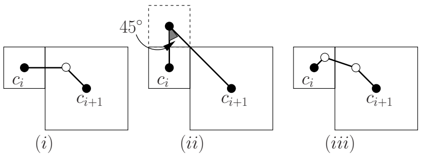


Fig. 10. Configs. for α_i when $s_i : s_{i+1}$ is $1 : 2$. (i) and (ii) from *applyTemplate* and stitching base cases. (iii) from stitching merge steps.

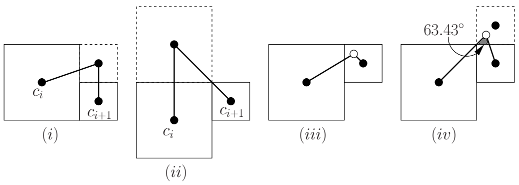


Fig. 11. Configurations for α_i when $s_i : s_{i+1}$ is $2 : 1$. (i) and (ii) come from *applyTemplate* and stitching base cases. (iii) and (iv) occur only in the stitching base cases.

ratio $s_i : s_{i+1}$ can be $1 : 1, 1 : 2, 2 : 1, 1 : 4$, or $4 : 1$. We consider the case of center quads first, and then consider templates and stitchings.

Case 2.1: α_i contains center quad edges. Let s be the size of the cell adjacent to c_i as well as c_{i+1} and lying above $\overrightarrow{c_ic_{i+1}}$. Possible configurations for α_i when $s_i : s : s_{i+1} \equiv 1 : 1 : 1$ are shown in Fig. 12. In 12(i), 12(iv), and 12(vi), the point in the cell adjacent to c_i and c_{i+1} may be either a cell center or an extra point of a larger cell. When $s_i : s : s_{i+1} \equiv 1 : 2 : 1$ or $s_i : s : s_{i+1} \equiv 1 : \frac{1}{2} : 1$, possible configurations of α_i are shown in Figs. 13 and 14, respectively.

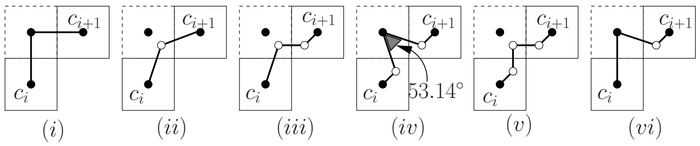


Fig. 12. $s_i : s : s_{i+1} \equiv 1 : 1 : 1$

When $s_i : s : s_{i+1} \equiv 1 : 1 : 2$, possible configurations of α_i are shown in Fig. 15. For the case when $s_i : s : s_{i+1} \equiv 2 : 1 : 1$, the α_i are obtained by reflections about the line $y = x$ of those in Case 2.1.4. Hence the minimum internal angle shown in Fig. 15 holds here as well. Similarly, Fig. 16 depicts α_i when $s_i : s : s_{i+1} \equiv 1 : 2 : 2$ and the chains in this figure are reflections about the line $y = x$ of possible α_i when $s_i : s : s_{i+1} \equiv 2 : 2 : 1$

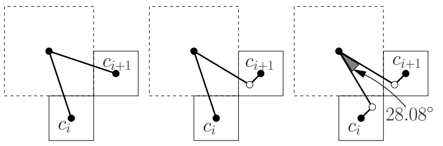


Fig. 13. $s_i : s : s_{i+1} \equiv 1 : 2 : 1$

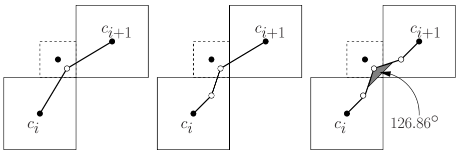


Fig. 14. $s_i : s : s_{i+1} \equiv 1 : \frac{1}{2} : 1$

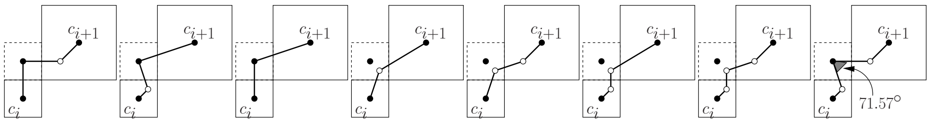


Fig. 15. $s_i : s : s_{i+1} \equiv 1 : 1 : 2$

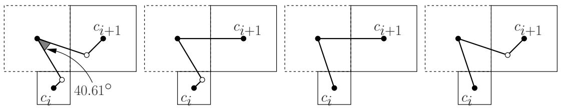


Fig. 16. $s_i : s : s_{i+1} \equiv 1 : 2 : 2$

Finally, Fig. 17 shows possible configurations of α_i when $s_i : s : s_{i+1} \equiv 1 : 2 : 4$. For the case when $s_i : s : s_{i+1} \equiv 4 : 2 : 1$, the α_i are obtained by 180° rotations of those in Fig. 17.

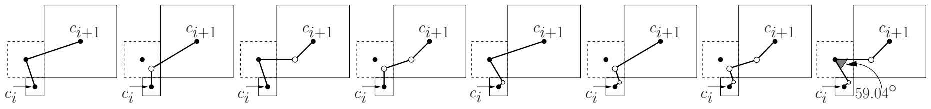


Fig. 17. $s_i : s : s_{i+1} \equiv 1 : 2 : 4$

Case 2.2: α_i constructed by application of applyTemplate or stitchChains. All new configurations of α_i that occur by a template application, or a stitching step at some level of recursion are listed. By "new", we mean configurations that do not appear in Figs. 12-17. Note that when c_i and c_{i+1} are connected via templates or stitchings, $s_i : s_{i+1}$ is $1 : 1$ (see Fig. 18), $1 : 2$, (see Fig. 19) or $2 : 1$ (reflections about the line of $y = x$ of the α_i in Fig. 19), but not $1 : 4$ or $4 : 1$.

While Figs. 12-19 all depict c_i and c_{i+1} in the southwest and northeast quadrants respectively, note that each of the α_i in these figures has a 90° rotational symmetry corresponding to c_i and c_{i+1} in the northwest and southeast quadrants, which does not change the minimum internal angles indicated in those figures.

It follows from the above case analysis that α_i has at most four edges. $\square$

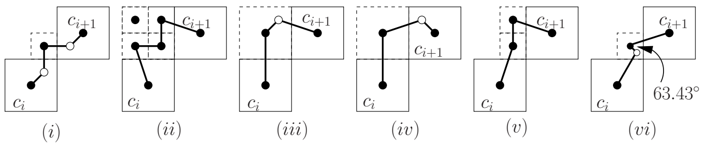


Fig. 18. $s_i : s_{i+1} \equiv 1 : 1$

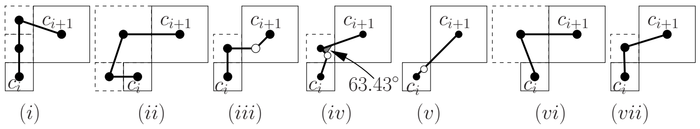


Fig. 19. $s_i : s_{i+1} \equiv 1 : 2$

We now describe how to quadrangulate each polygonal region $A_i = (v_i, c_i) \cdot \alpha_i \cdot (c_{i+1}, v_{i+1}) \cdot (v_{i+1}, v_i)$ independently for $1 \leq i \leq m$. Before doing this, we first show that rather than using the vertical projections v_i and v_{i+1}, we may instead use perpendicular projections of c_i and c_{i+1} onto edge $\vec{e}$ (Lemma 6). This allows us to prove angle bounds for quadrangulating A_i that are independent of the angle that $\vec{e}$ makes with the horizontal (recall that this is between $-45°$ and $45°$).

Lemma 5. *Let δ_i be the signed angle (in degrees) between $\overrightarrow{c_i c_{i+1}}$ and the positive x-axis. Then abs$(\delta_i) \in \{0, 18.43, 45, 71.57, 90\}$.*

Proof: Since c_i and c_{i+1} are both cell centers in $\mathcal{QT}$, and we know from Lemma 2 that they are edge or corner neighbors, it follows that there are a constant number of possibilities for δ_i: If c_i and c_{i+1} are edge neighbors with $s_i = s_{i+1}$, then δ_i is either 0 or 90°. If c_i and c_{i+1} are edge neighbors with $s_i \neq s_{i+1}$, then $\tan(\delta_i) = \frac{1}{3}$, i.e., abs$(\delta_i) = 18.43°$, or $\tan(\delta_i) = 3$, i.e., abs$(\delta_i) = 71.57°$. If c_i and c_{i+1} are corner neighbors, then abs$(\delta_i) = 45°$. □

Let θ be the signed angle made by $\vec{e}$ with the positive x-axis. The value of θ determines the range of possibilities for δ_i. This is because of our definition of the quadtree chain, which specifies that either cell c_i (resp. c_{i+1}) has center above $\vec{e}$ and is intersected by $\vec{e}$, or it is the north/west neighbor of a cell intersected by $\vec{e}$ whose center lies below $\vec{e}$. Table 1 summarizes the possible values of δ_i for given ranges of θ. This relationship also allows us to prove Lemma 6, stated here without proof.

Lemma 6. *Let p_i be the perpendicular projection of c_i on $\vec{e}$, and v_i the vertical projection of c_i on $\vec{e}$. Assume $\vec{e}$ makes an angle between $-45°$ and $45°$ with the positive x axis. Then for all $1 \leq i < m$, c_{i+1} lies outside the triangle $\Delta(p_i c_i v_i)$.*

We know from Lemma 3 that for all $1 \leq i \leq m$, α_i does not intersect $\overline{c_i v_i}$ or $\overline{c_{i+1} v_{i+1}}$. Furthermore, we know from the proof of Lemma 4 that α_i lies above $\overrightarrow{c_i c_{i+1}}$ (that is, it does not intersect the region bounded by $c_i v_i v_{i+1} c_{i+1}$). Therefore, Lemma 6 implies

Table 1. Range of values for θ and δ_i

Range of θ	**Values of δ_i**
$18.43° \leq \theta < 45°$	$-18.43° \leq \delta_i \leq 90°$
$0 < \theta < 18.43°$	$-45° \leq \delta_i < 90°$
$-18.43° < \theta \leq 0$	$-71.57° \leq \delta_i \leq 45°$
$-45° < \theta \leq -18.43°$	$-71.57° \leq \delta_i \leq 18.43°$

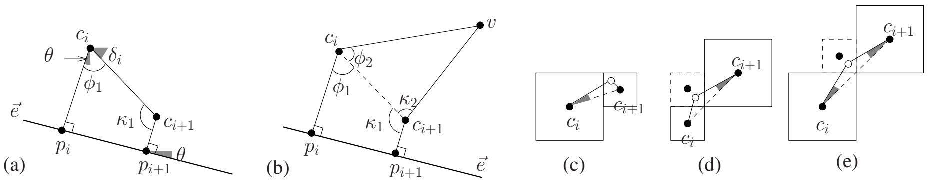


Fig. 20. (a) $\min\{\phi_1, \kappa_1\} \geq 18.43°$. (b) $\min\{\phi_1 + \phi_2, \kappa_1 + \kappa_2\} \geq 2 \times 18.43°$. (c) Shaded angle is $12.54°$. (d)-(e) Shaded angle is $14.04°$.

that α_i does not intersect $\overline{c_i p_i}$ or $\overline{c_{i+1} p_{i+1}}$ either. We redefine polygon A_i to be $(p_i, c_i) \cdot \alpha_i \cdot (c_{i+1}, p_{i+1}) \cdot (p_{i+1}, p_i)$ (that is, it is defined by the perpendicular projections rather than the vertical ones). Lemmas 7 and 8 establish bounds on some angles in A_i when α_i has one and two edges, respectively.

Lemma 7. *Let* $\phi_1 = \angle p_i c_i c_{i+1}$ *and* $\kappa_1 = \angle c_i c_{i+1} p_{i+1}$. *Then* $\min\{\phi_1, \kappa_1\} \geq 18.43°$.

Proof: Refer to Fig. 20(a). Since $\phi_1 = 90 - \theta + \delta_i$ and $\kappa_1 = 90 + \theta - \delta_i$ (recall θ and δ_i are signed angles), and the fact $-71.57° \leq (\theta - \delta_i) \leq 71.57°$ (refer to Table 1), it follows that $\phi_1 \geq 18.43°$ and $\kappa_1 \geq 18.43°$. □

Lemma 8. *Suppose* α_i *has two edges,* $c_i v$ *and* $v c_{i+1}$. *Let* $\phi_1 = \angle p_i c_i c_{i+1}$, $\kappa_1 = \angle c_i c_{i+1} p_{i+1}$, $\phi_2 = \angle c_{i+1} c_i v$, *and* $\kappa_2 = \angle c_i c_{i+1} v$. *Then (i)* $\min\{\phi_1, \kappa_1\} > 18.43°$ *and (ii)* $\min\{\phi_1 + \phi_2, \kappa_1 + \kappa_2\} \geq 2 \times 18.43°$.

Proof: (i) From Lemma 7 we know that $\min\{\phi_1, \kappa_1\} \geq 18.43°$. To see that it must be *strictly* greater, note that if $\min\{\phi_1, \kappa_1\} = 18.43°$ then abs$(\theta - \delta_i) = 71.57°$. From Lemma 5 and Table 1, it can be seen that abs$(\theta - \delta_i) = 71.57°$ when *(a)* $\theta = 18.43°$ and $\delta_i = 90°$, which is impossible because c_i and c_{i+1} are directly connected whenever $\delta_i = 90°$, or *(b)* $\theta = 0$ and $\delta_i = 71.57°$, which is also impossible because θ must be strictly greater than 0 whenever $\delta_i = 71.57°$.

(ii) Refer to Fig. 20. First observe that if $\min\{\phi_2, \kappa_2\} \geq 18.43°$, then part (i) implies the claim. Hence assume that $\min\{\phi_2, \kappa_2\} < 18.43°$. The only configurations of α_i for which $\min\{\phi_2, \kappa_2\} < 18.43°$ are shown in Fig. 20(c)-(e). We use the angle dependency in Table 1 to prove that these configurations imply $\min\{\phi_1 + \phi_2, \kappa_1 + \kappa_2\} \geq 2 \times 18.43°$. Further details are omitted here. □

Lemma 9. *For* $1 \leq i \leq m - 1$, *the simple polygon* $A_i = (p_i, c_i) \cdot \alpha_i \cdot (c_{i+1}, p_{i+1}) \cdot (p_{i+1}, p_i)$ *can be quadrangulated with at most five quadrilaterals with a minimum angle of* $18.43°$.

Proof: Since α_i has at most four edges (Lemma 4), we have four cases.

Case 1: α_i has one edge. In this case, A_i is already a quadrilateral. The fact that all angles of A_i are at least $18.43°$ follows from Lemma 7.

Case 2: α_i has two edges. Let $c_i v$ and $v c_{i+1}$ be the two edges of α_i. Let ϕ_1, κ_1, ϕ_2, and κ_2 be as in Fig. 20(b). Let $\gamma = \angle c_i v c_{i+1}$. Observe that $\gamma \geq 26.57°$ because the edges of α_i come from $\mathcal{Q}$. We quadrangulate A_i according to the angles ϕ_2 and κ_2.

- If $\min\{\phi_2, \kappa_2\} \geq 18.43°$, place a Steiner point s on $\overline{c_i c_{i+1}}$ and at the perpendicular projection p of s onto $\vec{e}$. Connect s to c_i, c_{i+1}, and p to obtain a quadrangulation of A_i. Perturb s towards p to obtain strictly convex quadrilaterals. We know from Lemma 8(i) that there is always a small perturbation of s that maintains all angles in the resulting quadrangulation at or above $18.43°$. See Fig. 21(a).

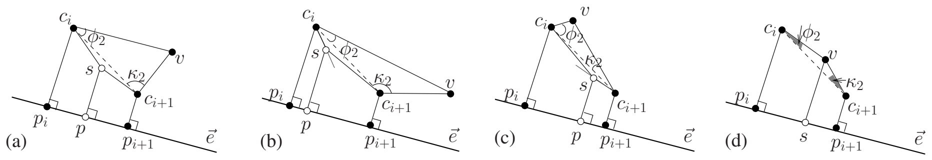


Fig. 21. α_i has two edges. (a) $\min\{\phi_2, \kappa_2\} \geq 18.43°$. (b) $\phi_2 < 18.43°$ and $\kappa_2 \geq 18.43°$. (c) $\phi_2 \geq 18.43°$ and $\kappa_2 < 18.43°$. (d) $\phi_2 < 18.43°$ and $\kappa_2 < 18.43°$.

- If $\min\{\phi_2, \kappa_2\} < 18.43°$, the placement of Steiner points depends on which of ϕ_2 and κ_2 are smaller than $18.43°$ (assume wlog that $\phi_1 + \phi_2 \leq \kappa_1 + \kappa_2$). If exactly one of ϕ_2 or κ_2 is less than $18.43°$, then Lemma 8 guarantees a quadrangulation of A_i with the required angle bounds. We omit details and refer to Fig. 21(b)-(c). If both ϕ_2 and κ_2 are less than $18.43°$, the only possible configuration for α_i is shown in Fig. 20(e). Observe that in this case, v can see $\vec{e}$. Let s be the perpendicular projection of v onto $\vec{e}$, unless $0 < \theta < -18.43°$, in which case let s be the vertical projection of v onto $\vec{e}$. Connect v to s to obtain a quadrangulation of A_i in which all angles satisfy the lower bound .

Case 3: α_i has three edges. The method used to quadrangulate A_i depends on the number of reflex internal vertices of α_i, which is zero or one (note that since α_i lies above $\overrightarrow{c_i c_{i+1}}$, it is not possible for both internal angles to be reflex):

- If the two internal angles along α_i are both convex, draw an edge between c_i and c_{i+1}, which quadrangulates A_i with two quadrilaterals. Some examples of such α_i can be seen in Fig. 10(iii) and 19(ii). In all such cases, Lemma 7 guarantees that all angles in the quad below $\overrightarrow{c_i c_{i+1}}$ is at least $18.43°$. The quad above $\overrightarrow{c_i c_{i+1}}$ has a minimum angle of $26.57°$. See Fig. 22(a).
- If one of the internal angles along α_i is reflex, c_i and c_{i+1} must be corner neighbors. Let r be the reflex vertex. r either lies on the segment $\overline{c_i c_{i+1}}$ (Fig. 22(b)), or belongs to the quadtree cell $N(c_i)$ adjacent to c_i and c_{i+1} and lying above $\overrightarrow{c_i c_{i+1}}$ (Fig. 22(c)). Several examples of the former appear in Figs. 12-17. For the latter, see Fig. 18(v)-(vi) and Fig. 19(iv). We can show that in both cases,

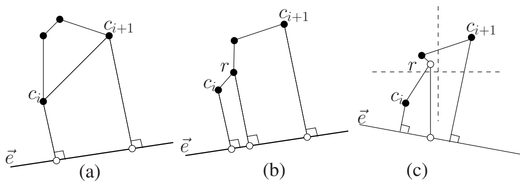


Fig. 22. α_i has three edges

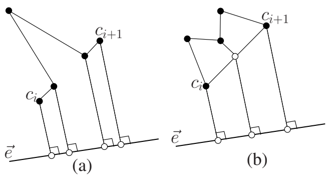


Fig. 23. α_i has four edges

A_i can be decomposed into a quadrilateral and a pentagon that is further decomposed into three quads with the required minimum angle bounds. Details are omitted here.

Case 4: α_i has four edges. α_i is classified according to the three internal vertices:

- If the three internal vertices consist of two reflex vertices separated by a convex vertex (e.g., Fig. 18(i)), the reflex vertices always lie on $\overline{c_i c_{i+1}}$. Insert edges from each reflex vertex to its perpendicular projection onto $\vec{e}$. This decomposes A_i into two quads and a pentagon. Lemmas 7 and 8 provide the required minimum angle bounds. See Fig. 23(a).
- If the three internal vertices consist of two convex vertices separated by a reflex vertex (Fig. 18(ii)), decompose A_i into four quads as shown in Fig. 23(b). Note that this decomposition has the required minimum angle bounds regardless of the value of θ.

This completes the proof that A_i can be quadrangulated with at most five quadrilaterals with a minimum angle of $18.43°$. □

Edge separation conditions for quadtree

Every edge $\vec{e}$ of the polygon P defines a chain of edges given by $\cup_{1 \leq i \leq m} \alpha_i$. From this chain, we obtain the polygons A_i, each of which is then quadrangulated as described above. In order to conduct this process independently for every edge of the polygon, we impose an *edge separation condition* on $\mathcal{QT}$. The edge separation condition requires that all quadrangulation chains $\cup_{1 \leq i \leq m} \alpha_i$ defined by the edges of the polygon be disjoint from each other. Recall that these chains do not start in the cell containing the segment endpoint, but rather in one adjacent to it. This allows quadrangulation chains to be separated completely, except in the 5×5 grid of cells around each polygon vertex. In the worst case, the edge separation condition requires that every cell intersected by a polygon edge be surrounded by a 3×3 grid of empty cells, but in practice, this requirement does not apply uniformly across the entire segment.

Connecting quadtree chains around polygon vertices

For every edge of the polygon P, the quadtree chain starts and ends at a cell center within the 3×3 grid of quadtree cells that is guaranteed to exist around each of its endpoints. Let v be a vertex of P and let e and f be the two oriented edges incident on v (the interior of P lies to their left). Let u be the last quadtree chain vertex for edge e and let w be the first quadtree chain vertex for edge f. Note that u and w are both cell centers in the 3×3 grid around v. Let $\bar{u}$ and $\bar{w}$ be the perpendicular projections of u and w onto e and f, respectively. Let E be a sequence of edges connecting u to w in the 3×3 grid. The region around vertex v is meshed by quadrangulating the polygon P_v defined by the edges $v\bar{u}, \bar{u}u, E, w\bar{w}, \bar{w}v$. The method used to quadrangulate P_v depends on the number of edges in E, which is between one and seven (inclusive). Refer to Fig. 24 for an illustration of some cases. The underlying 3×3 grid is used to prove the following lemma, stated here without proof.

Lemma 10. *P_v can be decomposed into at most seven quadrilaterals with a minimum angle of* $18.43°$.

Figure 25 shows quadrangulation chains $\cup_{1 \leq i \leq m} \alpha_i$ for some edges of a polygon (the entire polygon is shown in Fig. 28). Quadtree chain vertices are highlighted.

Summary of algorithm. We summarize in Figure 26 the algorithm to quadrangulate the interior of a non-acute simple polygon P of n edges $e_1, e_2, \ldots, e_n$ and vertices $v_0, v_1, \ldots, v_{n-1}$, where $e_i = (v_{i-1}, v_i)$ (where $v_n = v_0$). The resulting quadrilaterals have a minimum angle bound of $18.43°$.

Theorem 2. *Given a quadtree decomposition with N quadtree cells satisfying the edge separation condition for a simple polygon P, Quadrangulate(P, n) constructs a mesh for P with at most $5N$ quadrilaterals in which every angle is at least* $18.43°$.

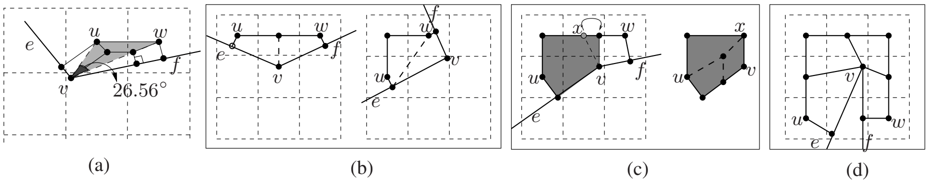


Fig. 24. Connecting at corners. Number of edges in E is (a) one, (b) two, (c)-(d) three or more.

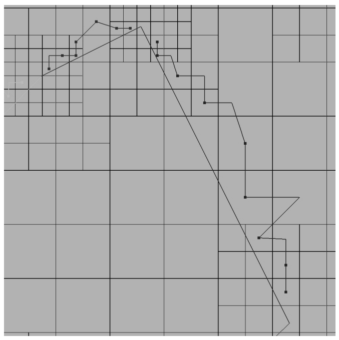


Fig. 25. Quadrangulation chains

```
Quadrangulate$(P, n)$:
    $\mathcal{QT}$ ← Quadtree decomposition satisfying edge separation conditions
            for vertices of $P$
    $\mathcal{Q}$ ← Quadrangulation resulting from applyTemplate on $\mathcal{QT}$.
    for $e_i \in \{e_1, e_2, \ldots, e_n\}$
        $\alpha(e_i)$ ← Quadrangulation chain for $e_i$
        $\mathcal{Q}(e_i)$ ← Quadrangulation of region bounded by $e_i$ and $\alpha(e_i)$,
                as given by Lemma 9.
        $\mathcal{Q}(v_i)$ ← Quadrangulation of corner polygon $P_{v_i}$,
                as given by Lemma 10.
    $\mathcal{Q}'$ ←$\bigcup_{1 \leq i \leq n} \mathcal{Q}(e_i) \cup \bigcup_{1 \leq i \leq n} \mathcal{Q}(v_i)$

    return $\mathcal{Q}' \cup (\mathcal{Q} \cap (P - \mathcal{Q}'))$
```

Fig. 26. Summary of algorithm

3.2 General Simple Polygons

Let P be a general simple polygon containing acute angles. We first convert P into a polygon that contains only obtuse angles by "cutting off" the acute angle vertices. Let a be an acute angle vertex of P. Let $\theta, 0 \leq \theta < 90$, be the angle at that vertex. Let v be a point on the angle bisector of a, and let p and q be the perpendicular projections of v onto the two edges incident at a. v is chosen so that the quadrangular region $apvq$ does not contain any other vertices of the polygon P. Cut all such regions $apvq$ from P. Let B be the polygon resulting from this procedure. Construct a quadrilateral mesh for B using the algorithm in Section 3.1. Observe that now there might be Steiner points on pv (e.g., v_1, v_2, v_3 in Fig. 27) and vq. These are used to quadrangulate the region $apvq$ with the required angle bounds. We omit details and refer to Fig. 27 for an illustration of the quadrangulation.

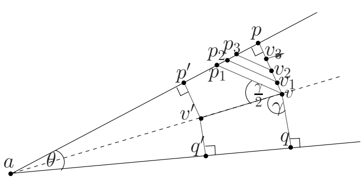


Fig. 27. Handling acute angles in P

4 Conclusion

Sample meshes generated by our algorithm are shown in Figures 28 and 29. Observe that in these examples the ratio of the number of quadrilaterals to the number of quadtree cells is less than one. The image on the right shows a zoomed-in portion of the mesh on the left.

This paper presents the first known result on the generation of a quadrilateral mesh for the interior of a simple polygon (possibly with holes) in which every new angle in

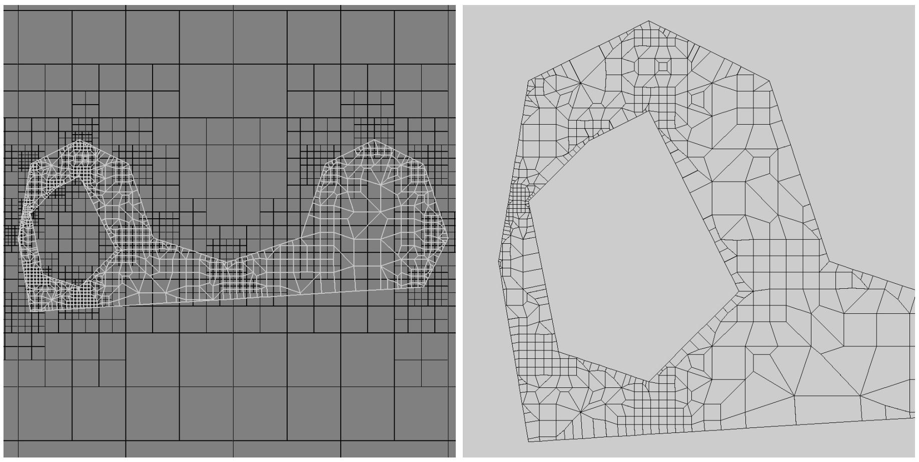


Fig. 28. Number of polygon edges: 19. Minimum mesh angle: 24.26°. Number of quadtree cells: 1399. Number of mesh vertices: 989. Number of mesh faces (quads): 841.

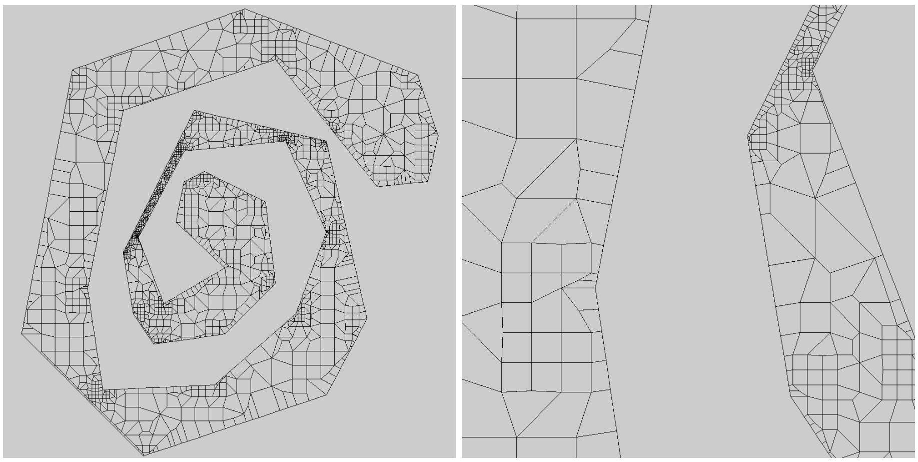


Fig. 29. Number of polygon edges: 33. Minimum mesh angle: 20.67°. Number of quadtree cells: 2623. Number of mesh vertices: 2213. Number of mesh faces (quads): 1859.

the mesh is bounded from below. The main open question resulting from this work is its extension to polygon interior as well as exterior. While our algorithm itself is applicable to the interior or the exterior of the polygon, the difficulty of adapting it to both lies in resolving mesh compatibility at the boundary without propagating the changes throughout the mesh. We are currently investigating alternative strategies to mesh the region bounded by quadtree chains on both sides of each polygon edge.

Acknowledgments. The authors would like to thank anonymous reviewers for helpful comments that served to improve the presentation in the paper.

References

1. Allman, D.J.: A quadrilateral finite element including vertex rotations for plane elasticity analysis. Int. J. Numer. Meth. in Engineering 26, 717–730 (1988)
2. Benzley, S., Perry, E., Merkley, K., Clark, B., Sjaardema, G.: A Comparison of All Hexahedral and All Tetrahedral Finite Element Meshes for Elastic and Elastic-Plastic Analysis. In: Proc. of the 4th International Meshing Roundtable, pp. 179–192 (1995)
3. Bern, M., Eppstein, D.: Quadrilateral meshing by circle packing. In: 6th IMR, pp. 7–19 (1997)
4. Bern, M., Eppstein, D., Gilbert, J.: Provably good mesh generation. J. Comp. Sys. Sci. 48, 384–409 (1994)
5. Blacker, T., Stephenson, M.: Paving: A new approach to automated quadrilateral mesh generation. Int. J. Numer. Meth. in Engineering 32(4), 811–847 (1991)
6. Brauer, J.R. (ed.): What every engineer should know about finite element analysis, 2nd edn. Marcel-Dekker, New York (1993)
7. Cheng, S.-W., Dey, T., Ramos, E., Ray, T.: Quality meshing for polyhedra with small angles. In: Proc. 20th Annual Symposium on Computational Geometry (2004)

8. Chew, L.P.: Guaranteed-quality mesh generation for curved surfaces. In: Proc. of the 9th ACM Symposium on Computational Geometry, pp. 274–280 (1993)
9. Chew, L.P.: Guaranteed-quality delaunay meshing in 3d. In: Proc. of the 13th ACM Symposium on Computational Geometry, pp. 391–393 (1997)
10. Johnston, B.P., Sullivan, J.M., Kwasnik, A.: Automatic conversion of triangular finite meshes to quadrilateral elements. Int. J. Numer. Meth. in Engineering 31(1), 67–84 (1991)
11. Miller, G.L., Talmor, D., Teng, S.-H., Walkington, N.: A delaunay based numerical method for three dimensions: Generation, formulation, and partition. In: Proc. of the 27th ACM Symposium on the Theory of Computing, pp. 683–692 (1995)
12. Mitchell, S., Vavasis, S.: Quality Mesh generation in Three Dimensions. In: Proceedings of the 8th Annual Symposium on Computational Geometry, pp. 212–221 (1992)
13. Ramaswami, S., Siqueira, M., Sundaram, T., Gallier, J., Gee, J.: Constrained quadrilateral meshes of bounded size. Int. J. Comp. Geom. and Appl. 15(1), 55–98 (2005)
14. Shewchuk, J.: Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery. In: Proc. of the 11th International Meshing Roundtable, pp. 193–204 (2002)
15. Shewchuk, J.R.: Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In: Lin, M.C., Manocha, D. (eds.) FCRC-WS 1996 and WACG 1996. LNCS, vol. 1148. Springer, Heidelberg (1996)
16. Zienkiewicz, O.C., Taylor, R.L.: The finite element method. McGraw-Hill, New York (1989)