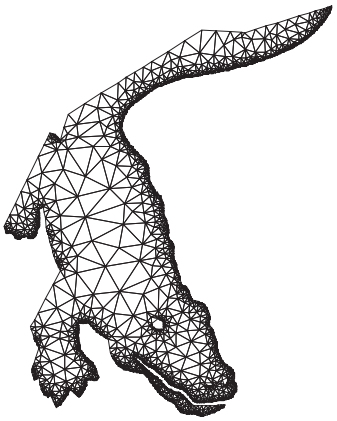
CHAPTER 12

Good Triangulations and Meshing

In this chapter, we show a beautiful application of quadtrees, demonstrating one of their most useful properties: Once we build a quadtree for a point set P, the quadtree captures the "shape" of the point set. We show how, using a quadtree, one can insert points into the given point set so that it looks smooth (i.e., the point density changes slowly).

12.1. Introduction – good triangulations

On simulation and meshing. In a lot of real world applications we are given a model of a physical object, and we would like to simulate some physical process on this object. For example, we are given a model of a solid engine of a rocket, and we would like to simulate how it burns out.



This process involves solving numerically a PDE (partial differential equation) on the given domain, describing the physical process involved. A numerical solver works better if the domain is geometrically simple. As such, we want to break the domain into triangles. The nicer the triangles are (i.e., fat and juicy) the easier it is for the numerical simulation. Naturally, as the simulation progresses, information is being passed between adjacent triangles; as such, the smaller the number of triangles the faster the simulation is.

So, given a geometric domain, we would like to compute a triangulation of the domain breaking it into a small number of fat triangles. This process is usually referred to as ***meshing***. An example of such a triangulation in shown in the figure on the right.[①]

To simplify the presentation, we assume that P has diameter $\geq 1/2$ and it is contained in the square $[1/4, 3/4]^2$. This can be achieved by scaling and translation.

Meshes without borders. Instead of meshing an arbitrary domain (bounded by some polygons), we will mesh a point set that "defines" this domain. The point set marks the critical features of the underlying domain and what features need to be preserved and respected by the resulting triangulation. Intuitively, by placing many points on the boundary of the original domain, we guarantee that the resulting triangulation of the point set would respect this boundary. Handling boundaries in meshing is a major headache but conceptually it is enough to solve the problem when we ignore boundary issues. Such algorithms can then be extended to handle input with boundary.

[①] Sadly, the research field of meshing two-dimensional alligators has only recently received the level of attention it deserves.

12.2. Triangulations and fat triangulations

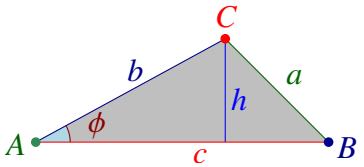
DEFINITION 12.1 (Triangulation). A ***triangulation*** of a domain $\mathcal{D} \subseteq \mathbb{R}^2$ is a partition of the domain into a set of interior disjoint triangles $\mathcal{M}$, such that for any pair of triangles $\triangle, \triangle' \in \mathcal{M}$, we have that either (i) $\triangle$ and $\triangle'$ are disjoint, (ii) $\triangle \cap \triangle'$ is a point which is a common vertex of both triangles or (iii) $\triangle \cap \triangle'$ is a complete edge of both triangles. The ***size*** of the triangulation $\mathcal{M}$, denoted by $|\mathcal{M}|$, is the number of triangles in it.

Observe that a triangulation does not allow a vertex of a triangle in the middle of an edge of an adjacent triangle.

DEFINITION 12.2 (Aspect ratio). The ***aspect ratio*** of a convex body is the ratio between its longest dimension and its shortest dimension. For a triangle $\triangle = \triangle abc$, the aspect ratio $\mathcal{A}_{\text{ratio}}(\triangle)$ is the length of the longest side divided by the height of the triangle when it's on the longest side.

LEMMA 12.3. *Let ϕ be the smallest angle of a triangle $\triangle$. We have that $1/\sin\phi \leq \mathcal{A}_{\text{ratio}}(\triangle) \leq 2/\sin\phi$. In particular, we have $\phi \geq 1/\mathcal{A}_{\text{ratio}}(\triangle)$.*

PROOF. Consider the triangle $\triangle = \triangle ABC$, depicted on the right. We have $\mathcal{A}_{\text{ratio}}(\triangle) = c/h$. However, $h = b\sin\phi$, and since a is the shortest edge in the triangle (since it is facing the smallest angle), it must be that b is the middle length edge. As such, by the triangle inequality, we have $2b \geq a + b \geq c$. Thus, $\mathcal{A}_{\text{ratio}}(\triangle) = c/h \geq b/h = b/(b\sin\phi) = 1/\sin\phi$. Similarly, $\mathcal{A}_{\text{ratio}}(\triangle) = c/h \leq 2b/h = 2b/(b\sin\phi) = 2/\sin\phi$.



As for the second claim, observe that $\sin\phi \leq \phi$, for all $\phi \geq 0$. As such, $\phi \geq \sin\phi \geq 1/\mathcal{A}_{\text{ratio}}(\triangle)$. ■

Another natural measure of fatness is edge ratio.

DEFINITION 12.4. The ***edge ratio*** of a triangle $\triangle$, denoted by $E_{\text{ratio}}(\triangle)$, is the ratio between a triangle's longest and shortest edges.

Clearly,

$$\mathcal{A}_{\text{ratio}}(\triangle) > E_{\text{ratio}}(\triangle), \tag{12.1}$$

for any triangle $\triangle$. For a triangulation $\mathcal{M}$, we denote by $\mathcal{A}_{\text{ratio}}(\mathcal{M})$ the maximum aspect ratio of a triangle in $\mathcal{M}$. Similarly, $E_{\text{ratio}}(\mathcal{M})$ denotes the maximum edge ratio of a triangle in $\mathcal{M}$.

DEFINITION 12.5. A triangle is ***α-fat*** if its aspect ratio is bounded by α. We will refer to a triangulation $\mathcal{M}$ as ***α-fat*** if all its triangles are α-fat.

12.2.1. Building well-balanced quadtrees.

DEFINITION 12.6. A ***corner*** of a quadtree cell is one of the four vertices of its square. The *corners* of the quadtree are the points that are corners of its cells. We say that the side of a cell is ***split*** if any of the neighboring boxes sharing it is split (i.e., the neighbor has a corner in the interior of this side). A quadtree is ***balanced*** if any side of a leaf cell contains at most one quadtree corner in its interior. Namely, adjacent leaves are either of the same level or of adjacent levels. (Naturally, a balanced quadtree cannot have compressed nodes.)

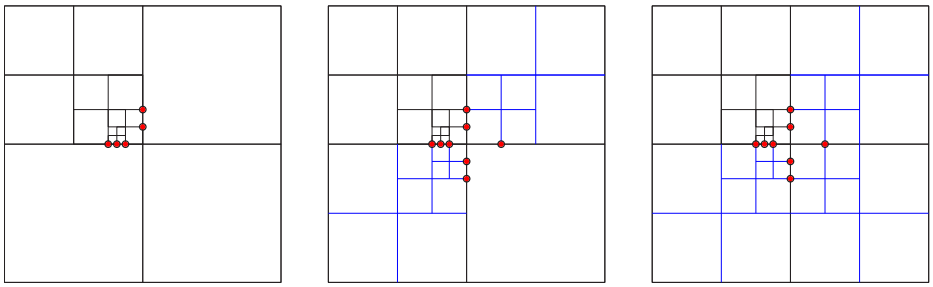


FIGURE 12.1. Rebalancing a quadtree: (i) the marked points are where the quadtree fails to be balanced, (ii) the quadtree resulting from refining nodes till the original unbalancing is fixed (however, new points where the quadtree is not balanced are now created), and (iii) the resulting balanced quadtree after the refining process continues.

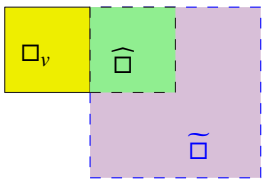
Given a quadtree, we can easily rebalance it by adding required nodes to satisfy the balancing requirement in a greedy fashion – if a leaf has a neighbor which is too large, we split the neighbor. An example of a rebalanced quadtree is depicted in Figure 12.1[②].

DEFINITION 12.7. The ***extended cluster*** of a cell c in a quadtree $\mathcal{T}$ is the set of 5×5 neighboring cells of c in the grid containing c. These are all the cells within a distance $< 2\varkappa(\square)$ from c in this grid, where $\varkappa(\square)$ is the sidelength of c.

A quadtree $\mathcal{T}$ over a point set P is ***well-balanced*** if it is balanced and for every leaf node v that contains a (single) point of P, we have the property that all the nodes of the extended cluster of v are in $\mathcal{T}$ and they do not contain any other point of P.

LEMMA 12.8. *Let P be a set of points in the plane, such that* $\operatorname{diam}(\mathsf{P}) = \Omega(1)$. *Then, one can compute a well-balanced quadtree $\mathcal{T}$ of P, in $O(n \log n + m)$ time, where m is the size of the output quadtree.*

PROOF. Compute a compressed quadtree $\mathcal{T}$ of P, in $O(n \log n)$ time, using the algorithm of Theorem 2.23_{p24}. Next, we traverse $\mathcal{T}$ and replace every compressed node of $\mathcal{T}$ by the sequence of quadtree nodes that define it. To guarantee the balance condition, we create a queue of the nodes of $\mathcal{T}$ and store the nodes of $\mathcal{T}$ in a hash table, with their IDs.



We handle the nodes in the queue one by one. For a node v, we check whether the current adjacent nodes to $\square_v$ are balanced. Specifically, let $\widehat{\square}$ be one of $\square_v$'s neighboring cells in the grid of $\square_v$ (there are four such neighbor cells), and let $\widetilde{\square}$ be the square containing $\widehat{\square}$ in the grid one level up; see the figure on the right. We compute $\operatorname{id}(\widetilde{\square})$ and check if there is a node in $\mathcal{T}$ with the ID $\operatorname{id}(\widetilde{\square})$. If not, then $\widehat{\square}$ and $\widetilde{\square}$ are both missing in the quadtree, and we need to create at least the node $\widetilde{\square}$ to fix the balancing property locally for the node $\square_v$. In this case, we create a node w with region $\widetilde{\square}$ and $\operatorname{id}(\widetilde{\square})$ and recursively retrieve its parent (i.e., if it exists, we retrieve it; otherwise, we create it) and hang w from the parent node. We add all the new nodes to the queue. We repeat the process till the queue is empty.

We charge the work involved in creating new nodes to the output size.

[②]It would be nice if people could be rebalanced as easily as quadtrees.

Now, for every leaf node v of $\mathcal{T}$ which contains a point of P, we check if its extended cluster contains any other point of P, and if so, we split it. Furthermore, for each child u of v we insert all the nodes in the extended cluster of u into the quadtree (if they do not already exist there). We repeat this process till all non-empty leaves have the required separation property. Of course, during this process, we keep the balanced property of the quadtree by adding necessary nodes.

The resulting quadtree is well-balanced, as required. Furthermore, the work can be charged to newly created nodes and as such takes linear time in the output size once the balanced compressed quadtree is computed.

Let t be the length of the closest pair of P, and observe that the algorithm will never create a node with side length smaller than t/c_1, where c_1 is some sufficiently large constant. As such, this algorithm terminates. Overall, the running time of the algorithm is $O(n \log n + m)$, since the work associated with any newly created quadtree node is a constant and can be charged to the output. ■

REMARK 12.9. For the sake of simplicity of exposition, the algorithm of Lemma 12.8 generates some nodes that are redundant. Indeed, the algorithm inserts all cluster nodes of a node if its parent cluster contains two points. In particular, it does not output the minimal well-balanced quadtree of the input point set. One can directly argue that the size of the tree generated is a constant factor larger than the minimal well-balanced tree or the given point set. Alternatively, our analysis below implies the same.

REMARK 12.10. A technicality we glossed over in describing the algorithm of Lemma 12.8 is how to implement the empty extended cluster queries efficiently. Specifically, given a cell $\square$ and a point p in it, we need to check if there is any other point in its extended cluster. Naturally, if the twenty five squares of the extended cluster exist in the quadtree, we can answer this query using hashing in constant time (in particular, when constructing a node v, a 'contain-points' flag is created that indicates if any point is stored in the subtree of v). However, it might be that such a cell query $\widehat{\square}$ (in the extended cluster of $\square$) does not exist in the tree $\mathcal{T}$.

To this end, we mark every node of the initial balanced quadtree if it contains points in its subtree. This information can be computed bottom-up in the balanced quadtree in linear time.

Next, we compute this flag whenever we create a new node u. We remind the reader that we store the points of P in the leaves of the quadtree. As such, when u is being created, it is formed by splitting a leaf node $\overline{\mathsf{p}}(u)$ of the quadtree, and this leaf node can contain only a single point of P. If $\overline{\mathsf{p}}(u)$ contains a point, we store it in the correct child of $\overline{\mathsf{p}}(u)$ and compute the 'contain-points' flag for all the newly minted children (which will be true only for one child[③]).

Now, during the well-balancing stage, we maintain the balancing property of $\mathcal{T}$ on the fly, and as such, the parent of such a query cell $\widehat{\square}$ must exist in the quadtree, and if not, the algorithm creates it since it needs to be created (so that the balancing condition is maintained). Let u' be the node of this bigger cell. If the flag indicates that u' does not contain points, then we are done. Otherwise, u' must be a leaf (otherwise, the cell of $\widehat{\square}$ would exist in the quadtree), and it contains a single point of P. We can now check in constant time if this point is inside $\widehat{\square}$.

[③]This is, the parents' favorite child, who is disliked by all the other children.

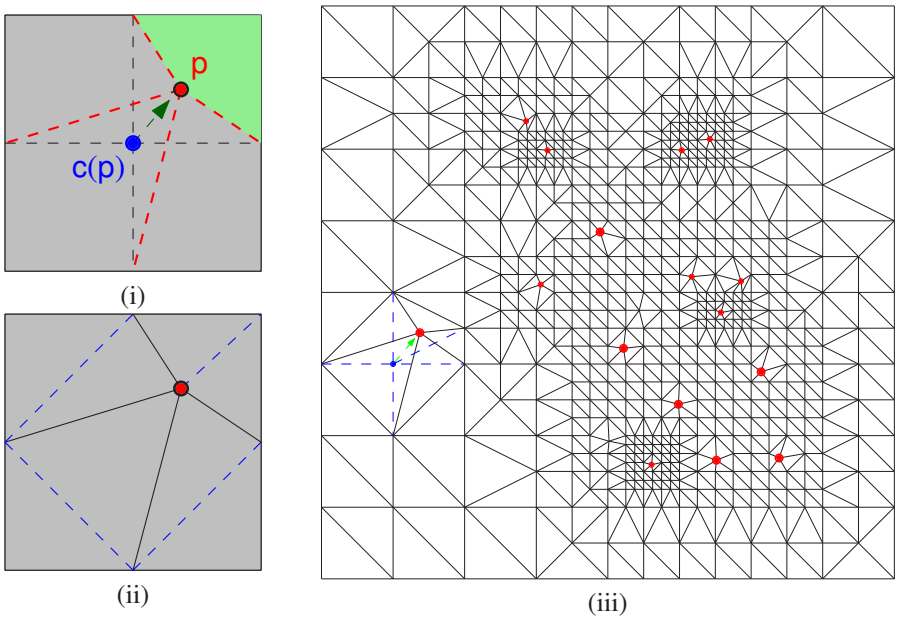


FIGURE 12.2. (i) We warp the square corner into the input point, turning the four squares into quadrangles. (ii) The triangulation of these quadrangles. (iii) A fat triangulation extracted from a quadtree. Note that this figure was generated by a variant of the algorithm we describe that does less refining (i.e., the quadtree used is not quite well-balanced).

12.2.2. The algorithm – extracting a good triangulation. The intuition behind the triangulation algorithm is that the well-balanced quadtree has leaves that have cells of the right size – they are roughly of the size that any triangle in a (small) fat triangulation of P has to be to cover the same region. Thus, we just need to decompose these cells into triangles in such a way that the points of P are vertices of the resulting triangulation.

Equipped with this vision, let us now plunge into the murky details. A well-balanced quadtree $\mathcal{T}$ of P provides for every point a region (i.e., its extended cluster) where it is well protected from other points. It is now possible to turn the partition of the plane induced by the leaves of $\mathcal{T}$ into a triangulation of P.

We "warp" the quadtree framework as follows. For a point $p \in P$, let $c(p)$ be the corner nearest p of the leaf of $\mathcal{T}$ containing p; we replace $c(p)$ by p. Thus, a 2×2 group of square leaves turns into a square divided into four (fat) quadrangles; see Figure 12.2(i).

Next, we triangulate the resulting planar subdivision. Regular squares can be triangulated by just introducing the diagonal, creating two isosceles right triangles. For cells containing points of P, we warped them into quadrangles, and we triangulate them by choosing the diagonal that gives a better aspect ratio (i.e., we choose the diagonal such that the maximum aspect ratio of the resulting two triangles is minimized); see Figure 12.2(ii).

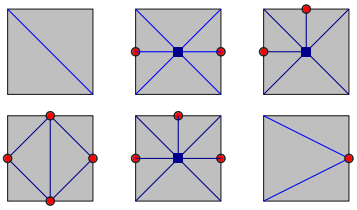


FIGURE 12.3

We also have to triangulate squares that have one or more split sides. Note that by our well-balanced property (see Definition 12.7), we have that *no* such node can be warped – so it is indeed a square. If there is a single split side, we break the square into three triangles by introducing the two long diagonals. If there are more split sides, we introduce the middle of the square and we split the square into right triangles in the natural way; see Figure 12.3. If all four sides are split, then we do not need to introduce a middle point.

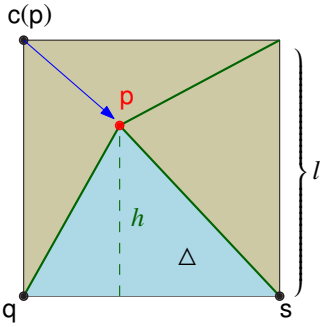
Let $\mathcal{QT}(\mathsf{P})$ denote the resulting ***triangulation***. Figure 12.2(iii) shows a triangulation resulting from a variant of this method.

12.3. Analysis

LEMMA 12.11. *The method above gives a triangulation $\mathcal{QT}(\mathsf{P})$ with $\mathcal{A}_{\text{ratio}}(\mathcal{QT}(\mathsf{P})) \leq 4$. The time to compute this triangulation is $O(n \log n + m)$, where $n = |\mathsf{P}|$ and m is the number of triangles in the triangulation.*

PROOF. The right triangles used for the unwarped cells have aspect ratio 2. If an original cell $\square$ with sidelength l is warped, we have two cases.

In the first case, the input point of P is inside the square of the original cell. To bound the aspect ratio, we assume that the diagonal touching the warped point is chosen; otherwise, the aspect ratio can only be smaller than what we prove (since we choose the diagonal that minimizes the aspect ratio). Consider one of the two triangles, say $\triangle = \triangle\mathsf{pqs}$, formed by the input point p (which is a warped corner) and the two other cell corners q and s. The maximum length diagonal is formed when the warped point p is at the original location $\mathsf{c(p)}$ and has length $x = \sqrt{2}l$. The minimum area of the triangle $\triangle$ is formed when the point is in the center of the square and has area $\alpha = l^2/4$. We have that $\text{area}(\triangle) = \|\mathsf{q} - \mathsf{s}\| \cdot h/2$, where h is the height of $\triangle$ when on the edge qs. Thus, we have



$$\text{height of the triangle } \triangle \geq 2\frac{\text{min area } \triangle}{\text{max length edge of } \triangle} = \frac{2\alpha}{x}$$

and

$$\mathcal{A}_{\text{ratio}}(\triangle) = \frac{\text{length of longest edge of } \triangle}{\text{height of } \triangle} \leq \frac{x}{2\alpha/x} = \frac{x^2}{2\alpha} = \frac{2l^2}{2l^2/4} = 4.$$

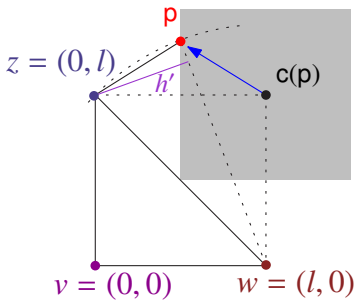


FIGURE 12.4

In the second case, the input point is outside the original square. Since the quadtree is well balanced, the new point p is somewhere inside a square of sidelength l centered at $\mathsf{c}(\mathsf{p})$ (since we always move the closest leaf corner to the new point). In this case, we assume that the diagonal not touching the warped point is chosen. This divides the cell into an isosceles right triangle $\triangle zvw$ and the other triangle $\triangle = \triangle z\mathsf{p}w$. If the chosen diagonal is the longest edge of $\triangle$, then one can argue as before, and the aspect ratio is bounded by 4. Otherwise, the longest edge of $\triangle$ is adjacent to the input point p. The altitude h' is minimized when the triangle is isosceles with as sharp an angle as possible; see Figure 12.4. Using the notation of the figure, we have $\mathsf{p} = (l/2, \sqrt{7}l/2)$. Thus, computing the area of a triangle, using Lemma 28.3,

$$\mu = \operatorname{area}(\triangle z\mathsf{p}w) = \frac{1}{2}\begin{vmatrix} 1 & 0 & l \\ 1 & l & 0 \\ 1 & l/2 & (\sqrt{7}/2)l \end{vmatrix} = \frac{1}{2}\begin{vmatrix} l & -l \\ l/2 & (\sqrt{7}/2-1)l \end{vmatrix} = \frac{\sqrt{7}-1}{4}l^2.$$

We have $h'\sqrt{2}l/2 = \mu$, and thus $h' = \sqrt{2}\mu/l = \frac{\sqrt{7}-1}{2\sqrt{2}}l$. The longest distance p can be from w is $\beta = \sqrt{(1/2)^2 + (3/2)^2}l = (\sqrt{10}/2)l$. Thus, the aspect ratio of the new triangle is bounded by $\beta/h' = \left(\sqrt{10}/2\right) / \frac{\sqrt{7}-1}{2\sqrt{2}} \approx 2.717 \leq 4$. ∎

12.3.1. On local feature size.

DEFINITION 12.12. The ***local feature size*** of a point p in the plane, with relation to a point set P, denoted by $\mathrm{lfs}_{\mathsf{P}}(\mathsf{p})$, is the radius of the smallest closed disk centered at p containing at least two points of P.

The following lemma testifies that the local feature size function changes slowly.

LEMMA 12.13. *The local feature is* 1*-Lipschitz; that is, for any* $\mathsf{p}, \mathsf{q} \in \mathbb{R}^2$*, we have that*

$$\mathrm{lfs}_{\mathsf{P}}(\mathsf{p}) - \|\mathsf{p} - \mathsf{q}\| \leq \mathrm{lfs}_{\mathsf{P}}(\mathsf{q}) \leq \mathrm{lfs}_{\mathsf{P}}(\mathsf{p}) + \|\mathsf{p} - \mathsf{q}\| .$$

PROOF. Let s and s' be the two nearest points to p in P. As such, we have $\mathrm{lfs}_{\mathsf{P}}(\mathsf{p}) = \max(\|\mathsf{p} - \mathsf{s}\|, \|\mathsf{p} - \mathsf{s}'\|)$. Also, by the triangle inequality, we have that $\|\mathsf{q} - \mathsf{s}\| \leq \|\mathsf{p} - \mathsf{q}\| + \|\mathsf{p} - \mathsf{s}\| \leq \|\mathsf{p} - \mathsf{q}\| + \mathrm{lfs}_{\mathsf{P}}(\mathsf{p})$ and $\|\mathsf{q} - \mathsf{s}'\| \leq \|\mathsf{p} - \mathsf{q}\| + \mathrm{lfs}_{\mathsf{P}}(\mathsf{p})$. As such,

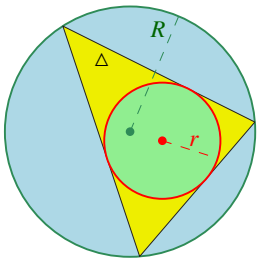
$$\mathrm{lfs}_{\mathsf{P}}(\mathsf{q}) \leq \max\left(\|\mathsf{q} - \mathsf{s}\|, \left\|\mathsf{q} - \mathsf{s}'\right\|\right) \leq \|\mathsf{p} - \mathsf{q}\| + \mathrm{lfs}_{\mathsf{P}}(\mathsf{p}) .$$

The other direction follows by a symmetric argument. ∎

12.3.2. From triangulations to covering by disks and back. In this section, we provide some intuition about how the local feature size relates to the meshing problem, by investigating a somewhat tangential problem of how to cover the domain with a few disks such that no disk covers too many points of P. In particular, the material here, except for building the intuition for our later proofs, is not strictly needed for our analysis, and the reader looking for a quick enlightenment[④] might skip directly to Section 12.3.3.

[④] And who isn't, in this world, on the lookout for a quick enlightenment on the cheap? Naturally, if you are looking for enlightenment in the middle of a section about meshing, you are looking for a very strange kind of enlightenment. The author can only hope that the text would provide it.

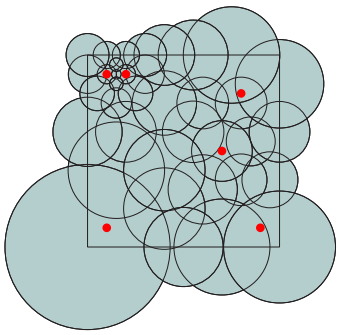
Intuitively, the local feature size captures the local size of fat triangles that one has to use, such that a fat triangulation P_{map} would contain P (as vertices). Here, the triangulation would have a set of vertices Q that contains P but might potentially contain many additional points inserted to make such a triangulation possible. Indeed, not all point sets have a fat triangulation if we do not allow inserting additional points.



The leap of faith here[5] is to think about a fat triangle in the triangulation P_{map} that we want to construct as being roughly a disk. Indeed, consider an α-fat triangle $\triangle$ and two disks associated with it, that is, the largest disk inscribed inside it with radius r and the smallest disk containing $\triangle$ of radius R; see the figure on the right. It is not hard to verify that $R/r = O(\alpha)$. Now, if we take the inscribed disk and move it a bit locally, it would not contain more than one point of P (since the adjacent triangles to $\triangle$ in the triangulation P_{map} have roughly the same size as $\triangle$).

So, for the time being, forget about the triangles in the triangulation, and imagine considering only their inscribed disks. Slightly changing the problem, we would like to cover the domain $\mathcal{D} = [0,1]^2$ with such disks. Conceptually, the question becomes how many disks we need to cover the domain. The name of the game is that we do not allow a disk to contain more than one point of the original point set P in its interior. Also, we will require that any two disks that intersect have roughly (up to a constant factor) the same radius.

Specifically, we are trying to find a minimum number of disks that cover the domain with these properties. Clearly, a disk in this collection centered at a point p can have radius at most $\mathrm{lfs}_{\mathsf{P}}(\mathsf{p})$.



Let us construct such a covering in a greedy fashion. Let $\mathcal{F}$ be the current set of disks chosen so far (initially, it is empty). As long as there is a point $\mathsf{p} \in \mathcal{D}$ that is not covered by any disk of $\mathcal{F}$, let us add the disk centered at p of radius $\mathrm{lfs}_{\mathsf{P}}(\mathsf{p})/2$ to $\mathcal{F}$. We stop when the domain is fully covered by the disks of $\mathcal{F}$. Since the points of P are distinct and the domain $\mathcal{D} = [0,1]^2$ is finite, this algorithm indeed stops. See the figure on the right for an example of what the output of this algorithm looks like. We refer to this algorithm as **algCoverDisksGreedy**.

Let us start by proving that the resulting set of disks has indeed the required properties.

LEMMA 12.14. *If* $\mathbf{b}(\mathsf{p}, r) \in \mathcal{F}$ *and* $\mathbf{b}(\mathsf{q}, r') \in \mathcal{F}$ *intersect and* $r' \leq r$*, then* $r/3 \leq r' \leq r$ *and* $\|\mathsf{p} - \mathsf{q}\| \geq r'$*, where* $\mathbf{b}(\mathsf{p}, r)$ *denotes the disk of radius* r *centered at* p*.*

PROOF. By construction, $r = \mathrm{lfs}_{\mathsf{P}}(\mathsf{p})/2$ and $r' = \mathrm{lfs}_{\mathsf{P}}(\mathsf{q})/2$. Since the two disks intersect, it follows that $\|\mathsf{p} - \mathsf{q}\| \leq r + r'$. Observe that, by Lemma 12.13, we have that

$$2r' = \mathrm{lfs}_{\mathsf{P}}(\mathsf{q}) \geq \mathrm{lfs}_{\mathsf{P}}(\mathsf{p}) - \|\mathsf{p} - \mathsf{q}\| \geq 2r - r - r' \quad \Longrightarrow \quad r' \geq r/3.$$

As for the second claim, if $\mathbf{b}(\mathsf{q}, r')$ was added to $\mathcal{F}$ after $\mathbf{b}(\mathsf{p}, r)$, then $\mathsf{q} \notin \mathbf{b}(\mathsf{p}, r)$ (otherwise q would be covered and the ball would have never been created). Namely,

[5] Naturally, if you do not like the resulting faith, you can always leap back.

$\|\mathsf{p}-\mathsf{q}\| \geq r \geq r'$. Arguing similarly, if $\mathbf{b}(\mathsf{p}, r)$ was added to $\mathcal{F}$ after $\mathbf{b}(\mathsf{q}, r')$, then $\|\mathsf{p}-\mathsf{q}\| \geq r'$. ■

Claim 12.15. *No point in the plane is covered by more than $c = O(1)$ disks of $\mathcal{F}$.*

Proof. The proof follows by a relatively easy packing argument. Indeed, let p be an arbitrary point in the plane that is covered by the disks of $\mathcal{F}$. Now, let X be the set of all disks of $\mathcal{F}$ covering p, and let $r_{\min}$ and $r_{\max}$ be the minimum and maximum radii of the disks of X. All the pairs of disks of X intersect since they contain p. As such, by Lemma 12.14, we have that $r_{\min} \leq r_{\max} \leq 3r_{\min}$, and the distance between any pair of centers of disks of X is at least $r_{\min}$. In particular, the centers of the disks of X are contained inside a disk of radius $r_{\max}$ centered at p.

So, place a disk of radius $r_{\min}/2$ centered at each one of these centers and let $\mathcal{G}$ be the resulting set of disks. The disks of $\mathcal{G}$ are interior disjoint disks and are all contained in a disk of radius $r_{\max} + r_{\min}/2 \leq 2r_{\max}$ centered at p. As such, by charging these disks to the area they occupy in $\mathbf{b}(\mathsf{p}, 2r_{\max})$, we have that

$$|X| = |\mathcal{G}| \leq \frac{\operatorname{area}(\mathbf{b}(\mathsf{p}, 2r_{\max}))}{\operatorname{area}(\mathbf{b}(r_{\min}/2))} = \frac{4\pi r_{\max}^2}{\pi r_{\min}^2/4} \leq \frac{16 r_{\max}^2}{(r_{\max}/3)^2} = 144.$$

■

So, how many disks do we expect to have in such a cover? Well, if a point p has local feature size α, then the disk of $\mathcal{F}$ centered at p has radius $\alpha/2$, and it has area $\Theta(\alpha^2)$. If the local feature size was the same everywhere, then all the disks we would use would be of area $\pi\alpha^2/4$ and the number of disks needed would be proportional to $\operatorname{area}(\mathcal{D})/\alpha^2$. Since the local feature size changes over the domain, we have to adapt to it in bounding the number of disks used. One possible way to do so is to use an integral. Thus, we claim that the number of disks we have to use is roughly

$$\#_{\triangle}(\mathsf{P}, \mathcal{D}) = \int_{\mathsf{p}\in\mathcal{D}} \frac{1}{\left(\operatorname{lfs}_{\mathsf{P}}(\mathsf{p})\right)^2}\, d\mathsf{p}, \tag{12.2}$$

where $\mathcal{D} = [0,1]^2$ is the domain being covered.

Lemma 12.16. *For a set of points $\mathsf{P} \subseteq [1/4, 3/4]^2$, let $\mathcal{F}$ be the cover of $\mathcal{D}$ computed by* **algCoverDisksGreedy**$(\mathsf{P}, \mathcal{D})$. *Then $|\mathcal{F}| = \Theta(\#_{\triangle}(\mathsf{P}, \mathcal{D}))$.*

Proof. For a ball $\mathbf{b}(\mathsf{p}, r) \in \mathcal{F}$, we have that $r = \operatorname{lfs}_{\mathsf{P}}(\mathsf{p})/2$. As such, by Lemma 12.13, we have, for any point $\mathsf{t} \in \mathbf{b}(\mathsf{p}, r)$, that $\|\mathsf{p}-\mathsf{t}\| \leq r$ and

$$r = \operatorname{lfs}_{\mathsf{P}}(\mathsf{p}) - r \leq \operatorname{lfs}_{\mathsf{P}}(\mathsf{p}) - \|\mathsf{p}-\mathsf{t}\| \leq \operatorname{lfs}_{\mathsf{P}}(\mathsf{t}) \leq \operatorname{lfs}_{\mathsf{P}}(\mathsf{p}) + \|\mathsf{p}-\mathsf{t}\| \leq \operatorname{lfs}_{\mathsf{P}}(\mathsf{p}) + r = 3r.$$

As such,

$$\begin{aligned}\#_{\triangle}(\mathsf{P}, \mathcal{D}) &= \int_{\mathsf{t}\in\mathcal{D}} \frac{1}{\left(\operatorname{lfs}_{\mathsf{P}}(\mathsf{t})\right)^2}\, d\mathsf{t} \leq \sum_{\mathbf{b}(\mathsf{p},r)\in\mathcal{F}} \int_{\mathsf{t}\in\mathbf{b}(\mathsf{p},r)} \frac{1}{\left(\operatorname{lfs}_{\mathsf{P}}(\mathsf{t})\right)^2}\, d\mathsf{t} \leq \sum_{\mathbf{b}(\mathsf{p},r)\in\mathcal{F}} \int_{\mathsf{t}\in\mathbf{b}(\mathsf{p},r)} \frac{1}{r^2}\, d\mathsf{t} \\ &\leq \sum_{\mathbf{b}(\mathsf{p},r)\in\mathcal{F}} \frac{\pi r^2}{r^2} \leq \pi\,|\mathcal{F}|\,.\end{aligned}$$

As for the other direction, observe that, by construction, every disk of $\mathcal{F}$ has its center in the domain $\mathcal{D}$. As such, at least a quarter of each disk lies inside the domain and contributes to the integral of $\#_{\triangle}(\mathsf{P}, \mathcal{D})$. Also, every point in the plane in the domain is covered by at most c disks of $\mathcal{F}$, by Claim 12.15. As such, we have that

$$\#_{\triangle}(\mathsf{P}, \mathcal{D}) = \int_{\mathsf{t}\in\mathcal{D}} \frac{1}{\left(\operatorname{lfs}_{\mathsf{P}}(\mathsf{t})\right)^2}\, d\mathsf{t} \geq \frac{1}{c} \sum_{\mathbf{b}(\mathsf{p},r)\in\mathcal{F}} \int_{\mathsf{t}\in\mathbf{b}(\mathsf{p},r)\cap\mathcal{D}} \frac{1}{\left(\operatorname{lfs}_{\mathsf{P}}(\mathsf{t})\right)^2}\, d\mathsf{t}$$

$$\geq \frac{1}{c} \sum_{\mathbf{b}(\mathsf{p},r)\in\mathcal{F}} \int_{\mathsf{t}\in\mathbf{b}(\mathsf{p},r)\cap\mathcal{D}} \frac{1}{(3r)^2}\, d\mathsf{t} \geq \frac{1}{c} \sum_{\mathbf{b}(\mathsf{p},r)\in\mathcal{F}} \frac{1}{4} \int_{\mathsf{t}\in\mathbf{b}(\mathsf{p},r)} \frac{1}{9r^2}\, d\mathsf{t}$$

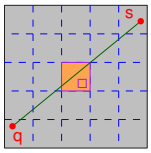
$$= \frac{1}{36c} \sum_{\mathbf{b}(\mathsf{p},r)\in\mathcal{F}} \frac{\pi r^2}{r^2} = \frac{\pi}{36c} |\mathcal{F}|.$$

■

12.3.3. Upper bound. The problem at hand is to bound the number of triangles in any "minimal" fat triangulation (of $\mathcal{D}$) that has P as a subset of the vertices. Specifically, we would like to bound the number of triangles in the triangulation generated by the algorithm of Section 12.2.2.

Intuitively, a triangle in such a triangulation containing a point p should have diameter roughly $\mathrm{lfs}_{\mathsf{P}}(\mathsf{p})$. Naturally, we can use smaller triangles than the local feature size recommends, but luckily, we do not have to. To this end, we argue that the local feature size at a point p is bounded by the size of the leaf of the quadtree containing p. In fact, the other direction also holds (that is, the local feature size, up to a constant, bounds the diameter of the leaf), but we do not need it for our argument.

First, we will prove that leaves of the well-balanced quadtree of P are not too small. This implies that the size of the triangulation that the algorithm extracts is $O(\#\triangle(\mathsf{P},\mathcal{D}))$, see (12.2). As for the other direction, we will prove that any good fat triangulation of P must have $\Omega(\#\triangle(\mathsf{P},\mathcal{D}))$ triangles in it.



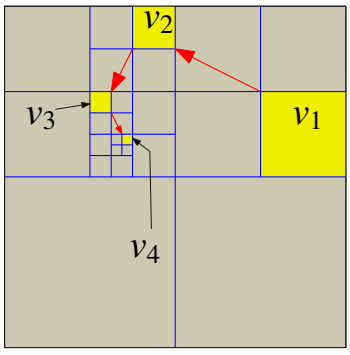
Observation 12.17. *Let $\square$ be a cell in some canonical grid. Then, for any two points p and s that lie in the extended cluster of $\square$, we have that $\|\mathsf{p}-\mathsf{s}\| \leq \tau(\square) = 5\sqrt{2}\varkappa(\square)$, where $\varkappa(\square)$ denotes the sidelength of $\square$. See the figure on the right.*

Lemma 12.18. *Let v be a leaf of the well-balanced quadtree of P (computed by the algorithm of Lemma 12.8), and let p be any point in $\square_v$. Then, $\mathrm{lfs}_{\mathsf{P}}(\mathsf{p}) \leq c\,\varkappa(\square_v)$, where $\varkappa(\square_v)$ is the sidelength of the cell $\square_v$ and c is some constant.*

Proof. If the cell $\square_v$ was created because there was an extended cluster that contained two points $\mathsf{q},\mathsf{s} \in \mathsf{P}$ and the algorithm refined the parent $\overline{\mathsf{p}}(v)$ of v because of this, then q,s are in distance at most $\alpha = 5\sqrt{2}\varkappa\left(\square_{\overline{\mathsf{p}}(v)}\right) = 2\cdot 5\cdot \sqrt{2}\varkappa(\square_v)$ from any point of $\square_v \subseteq \square_{\overline{\mathsf{p}}(v)}$, by Observation 12.17. We have

$$\mathrm{lfs}_{\mathsf{P}}(\mathsf{p}) \leq \max(\|\mathsf{p}-\mathsf{q}\|, \|\mathsf{p}-\mathsf{s}\|) \leq \alpha = 10\sqrt{2}\varkappa(\square_v),$$

establishing the claim in this case.



From this point on, the remainder of the proof is a merry-go-round game of blame. A node u where its parent got refined because of balancing considerations ***blames*** the node $w = \mathrm{J'accuse}(u)$ that caused this splitting for its existence. The cell $\square_w$ is adjacent to the cell $\square_{\overline{\mathsf{p}}(u)}$ and $\varkappa(\square_w) \leq \varkappa\left(\square_{\overline{\mathsf{p}}(u)}\right)/4 = \varkappa(\square_u)/2$. Otherwise the parent of $u' = \overline{\mathsf{p}}(u)$ would be balanced with relation to its neighbors and would not get split. Note that w must have been created in the tree before u' got split (and u got created).

So, let $v_1 = v$, and let v_i be the $\mathrm{J'accuse}(v_{i-1})$, for $i = 1,\ldots,k$; see the figure on the right. Clearly, the blame game must end at some node

v_k (because the quadtree is finite, and every time we play the blame game, the cell of the current node shrinks by half at least). The vertex v_k must have been inserted because it participates in some extended cluster of a node u_k that contains some point of P, such that the algorithm inserted all the nodes of the extended cluster of u_k into the quadtree. However, this happens only if the extended cluster of the parent of u_k contains two points of P. It easy to verify that since the cells of v_k and u_k are in the same grid (and belong to the extended clusters of each other), then the extended cluster of the grandparent of v_k contains the extended cluster of the parent of u_k. Namely, the extended cluster of the grandparent of v_k contained two points of P.

As such, we have that

$$\operatorname{lfs}_{\mathsf{P}}(\mathsf{p}) \leq \sum_{i=1}^{k-1} (\operatorname{diam}(\square_{v_i}) + \mathbf{d}(\square_{v_i}, \square_{v_{i+1}})) + \tau(\overline{\mathsf{p}}(\overline{\mathsf{p}}(v_k))),$$

where $\mathbf{d}(X, Y) = \min_{x \in X, y \in Y} \|x - y\|$ and $\tau(\square) = 5\sqrt{2}\varkappa(\square)$ (see Observation 12.17). By the above, we know that $\operatorname{diam}(v_i) \leq \operatorname{diam}(v_{i-1})/2 \leq \operatorname{diam}(v)/2^{i-1}$, for all i. Using similar argumentation, we have that

$$\mathbf{d}(\square_{v_i}, \square_{v_{i+1}}) \leq \operatorname{diam}(\overline{\mathsf{p}}(v_i)) \leq 2\operatorname{diam}(\square_{v_i}) \leq \operatorname{diam}(\square_v)/2^{i-2}.$$

As such, we have that

$$\operatorname{lfs}_{\mathsf{P}}(\mathsf{p}) \leq \sum_{i=1}^{k-1} \left(\frac{\operatorname{diam}(\square_v)}{2^{i-1}} + \frac{\operatorname{diam}(\square_v)}{2^{i-2}} \right) + 20\sqrt{2}\varkappa(\square_{v_k}) = O(\varkappa(\square_v)),$$

establishing the claim. ∎

Lemma 12.19. *The number of leaves in the well-balanced quadtree of* P *(computed by the algorithm of Lemma 12.8) is* $O(\#\triangle(\mathsf{P}, \mathcal{D}))$*, where* $\mathcal{D} = [0, 1]^2$*. Namely, the size of the triangulation* $\mathcal{QT}(\mathsf{P})$ *is* $O(\#\triangle(\mathsf{P}, \mathcal{D}))$*.*

Proof. Consider a leaf v of the well-balance quadtree of P, and observe that by Lemma 12.18, we have that

$$\int_{\mathsf{p} \in \square_v} \frac{1}{(\operatorname{lfs}_{\mathsf{P}}(\mathsf{p}))^2}\, d\mathsf{p} \geq \int_{\mathsf{p} \in \square_v} \frac{1}{(c\,\varkappa(\square_v))^2} d\mathsf{p} = \frac{\operatorname{area}(\square_v)}{(c\,\varkappa(\square_v))^2} = \frac{(\varkappa(\square_v))^2}{(c\,\varkappa(\square_v))^2} = \frac{1}{c^2}.$$

Namely, since the leaves of the well-balanced quadtree are interior disjoint, and each one contributes (at least) $1/c^2$ to $\#\triangle(\mathsf{P}, \mathcal{D})$, it follows that the number of leaves of the quadtree is at most $\#\triangle(\mathsf{P}, \mathcal{D})/(1/c^2) = O(\#\triangle(\mathsf{P}, \mathcal{D}))$; see $(12.2)_{\text{p171}}$. (Note that c is a universal fixed constant that depends only on the dimension; see Claim 12.15). ∎

12.3.4. Lower bound. The other direction of showing that every triangle (in a fat triangulation) contributes at most a constant to $\#\triangle(\mathsf{P}, \mathcal{D})$ requires us to prove that the local feature size anywhere inside such a triangle $\triangle$ is $\Omega(\operatorname{diam}(\triangle))$. The key observation is that a triangle in such a triangulation is an island – it has no vertices of the triangulation close to it, thus implying that the local feature size is indeed sufficiently large inside it.

Lemma 12.20. *Let* $\mathcal{M}$ *be an* α*-fat triangulation, and let* p *be a vertex of this triangulation. Consider the shortest edge* $\mathsf{e}_{\text{short}}$ *and the longest edge* e_{long} *in the triangulation adjacent to* p*. We have that* $\left\|\mathsf{e}_{\text{long}}\right\| / \|\mathsf{e}_{\text{short}}\| \leq c_2$*, where* c_2 *is a constant that depends only on* α*.*

PROOF. Let $\mathsf{e}_1, \mathsf{e}_2, \ldots, \mathsf{e}_k$ be the edges of $\mathcal{M}$ in (say) clockwise order around p, where $\mathsf{e}_1 = \mathsf{e}_{\text{short}}$ and e_k is adjacent to e_1. Let $\triangle_i$ be the triangle formed by the edges e_i and e_{i+1}, for $i = 1, \ldots, k-1$, and let $\triangle_k$ be the triangle formed by e_k and e_1. These triangles appear in $\mathcal{M}$, and since this triangulation is α-fat, we have that the aspect ratio of all these triangles is bounded by α (see Definition 12.5). As such, we have that

$$\frac{\|\mathsf{e}_i\|}{\|\mathsf{e}_1\|} = \frac{\|\mathsf{e}_i\|}{\|\mathsf{e}_{i-1}\|} \cdot \frac{\|\mathsf{e}_{i-1}\|}{\|\mathsf{e}_{i-2}\|} \cdots \frac{\|\mathsf{e}_2\|}{\|\mathsf{e}_1\|} \leq \alpha^{i-1}.$$

Similarly, by going around p in the other direction, we have that $\|\mathsf{e}_i\|/\|\mathsf{e}_1\| \leq \alpha^{k-i+1}$.

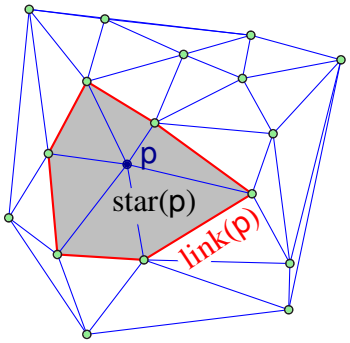
Now, all the angles of these triangles are at least $\geq 1/\mathcal{A}_{\text{ratio}}(\triangle_i) \geq 1/\alpha$, by Lemma 12.3. As such, $k \leq \lceil 2\pi/(1/\alpha) \rceil = \lceil 2\pi\alpha \rceil$. We conclude that

$$\frac{\|\mathsf{e}_i\|}{\|\mathsf{e}_{\text{short}}\|} \leq \max\left(\alpha^i, \alpha^{k-i+1}\right) \leq \alpha^{\lceil 2\pi\alpha \rceil/2},$$

for all i. In particular, setting $c_2 = \alpha^{\lceil 2\pi\alpha \rceil/2}$ implies the claim. ■

It is useful to know the following concepts even if we will make only light use of them.

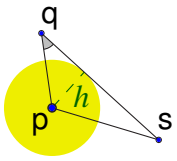
DEFINITION 12.21. In a triangulation $\mathcal{M}$, the ***star*** of a vertex $\mathsf{p} \in \mathcal{M}$ is the union of triangles of $\mathcal{M}$ adjacent to p. The ***link*** of p is the boundary of its star. See the figure on the right.



LEMMA 12.22. *Let $\mathcal{M}$ be an α-fat triangulation, and let p be a vertex of this triangulation. Let $\mathsf{e} \in \mathcal{M}$ be the longest edge adjacent to p. Then, the disk of radius $r = c_3 \|\mathsf{e}\|$ centered at p is contained in* star(p)*, where c_3 is some constant that depends only on α.*

PROOF. By Lemma 12.20, the shortest edge adjacent to p has length at least $\|\mathsf{e}\|/c_2$.

Consider a triangle $\triangle = \triangle\mathsf{qps}$ of $\mathcal{M}$ adjacent to p, depicted on the right, and observe that since this triangle is α-fat, we have that angle $1/\alpha \leq \angle\mathsf{pqs} \leq \pi - 2\alpha$, since, by Lemma 12.3, the minimum angle of this triangle is at least $1/\alpha$. As such, the distance of p to the edge qs is at least



$$h = \|\mathsf{p} - \mathsf{q}\| \sin \angle\mathsf{pqs} \geq \frac{\|\mathsf{e}\|}{c_2} \sin\frac{1}{\alpha} \geq \frac{\|\mathsf{e}\|}{2c_2\alpha},$$

since $\sin x \geq x/2$ for $x \in [0, 1]$. Thus, the claim holds with $c_3 = 1/(2c_2\alpha)$. ■

The following lemma testifies that for a triangle in a fat triangulation there is a buffer zone around it (i.e., a moat) such that no point of the triangulation might lie in it.[6]

[6] Completely irrelevant, but here is a nonsensical geometric statement using biblical language: "Why do you see the moat that is in your brother's triangle, but don't consider the moat that is in your own triangle?" Answer me that!

LEMMA 12.23. *Let $\mathcal{M}$ be an α-fat triangulation, with V being the set of vertices of $\mathcal{M}$. For any triangle $\triangle \in \mathcal{M}$, we have that*

$$\mathbf{d}(V \setminus \triangle, \triangle) \geq c_4 \mathrm{diam}(\triangle),$$

where $\mathbf{d}(X, Y)$ is the minimum distance between any pair of points of X and Y and c_4 is some constant that depends only on α.

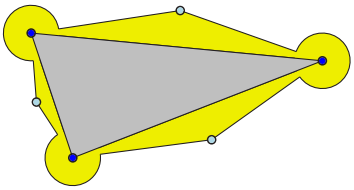


FIGURE 12.5

PROOF. Let e be the longest edge of $\triangle$, and observe that the shortest edge of $\triangle$ has length $\geq \|\mathsf{e}\|/\alpha$. By Lemma 12.22, one can place disks of radius $\beta = c_3 \|\mathsf{e}\|/\alpha$ at the vertices of $\triangle$, such that these disks are contained inside the respective stars of their vertices.

Now, erect on each edge of $\triangle$ an isosceles triangle having base angle $1/\alpha$ and lying outside $\triangle$. Clearly, these triangles are contained inside the real triangles of the mesh lying on their respective edges since all angles (of triangles) in the mesh are larger than or equal to $1/\alpha$.

Let T denote the union of these three triangles together with the protective disks; see Figure 12.5. The "moat" T forms a region that contains no vertex of the mesh except the vertices of the triangle $\triangle$. It is now easy to verify that the outer boundary of T is within a distance at least

$$\beta \sin \frac{1}{\alpha} \geq \frac{c_3 \|\mathsf{e}\|}{2\alpha^2} = \frac{c_3}{2\alpha^2} \mathrm{diam}(\triangle)$$

from any point in $\triangle$, since $\sin(1/\alpha) \geq 1/2\alpha$ for $\alpha \geq 1$. The claim now follows by setting $c_4 = c_3/2\alpha^2$. ∎

LEMMA 12.24. *For any α-fat triangulation $\mathcal{M}$ of $\mathcal{D} = [0,1]^2$ that includes the points of P as vertices, we have that $|\mathcal{M}| = \Omega(\#_\triangle(\mathsf{P}, \mathcal{D}))$; see (12.2)$_{p171}$.*

PROOF. Consider a triangle $\triangle \in \mathcal{M}$, and observe that the length of the edges of $\triangle$ are at least $\mathrm{diam}(\triangle)/\alpha$. By Lemma 12.23, all the other vertices of $\mathcal{M}$ are within a distance at least $c_4 \mathrm{diam}(\triangle)$ from $\triangle$. Thus, for a point $\mathsf{p} \in \triangle$, it might be that the nearest neighbor to p, among the vertices of $\mathcal{M}$, is relatively close (i.e., one of the vertices of $\triangle$), but the second nearest neighbor must be within a distance at least $\min(\mathrm{diam}(\triangle)/2\alpha, c_4 \mathrm{diam}(\triangle)) = c_4 \mathrm{diam}(\triangle)$, as $c_4 \leq 1/2\alpha$ (which can be easily verified). We conclude that

$$\forall \mathsf{p} \in \triangle \quad \mathrm{lfs}_V(\mathsf{p}) \geq c_4 \mathrm{diam}(\triangle), \tag{12.3}$$

where V is the set of vertices of $\mathcal{M}$ (and $\mathsf{P} \subseteq V$).

Observe that the height of a triangle is bounded by its diameter. As such, a triangle $\triangle$ with diameter $\mathrm{diam}(\triangle)$ has $\mathrm{area}(\triangle) \leq (\mathrm{diam}(\triangle))^2/2$. We are now close enough to the end of the tunnel to see the light; indeed, we have that

$$\begin{aligned}\int_{\mathsf{p} \in \triangle} \frac{1}{(\mathrm{lfs}_{\mathsf{P}}(\mathsf{p}))^2}\, d\mathsf{p} &\leq \int_{\mathsf{p} \in \triangle} \frac{1}{(\mathrm{lfs}_V(\mathsf{p}))^2}\, d\mathsf{p} \leq \int_{\mathsf{p} \in \triangle} \frac{1}{(c_4 \mathrm{diam}(\triangle))^2}\, d\mathsf{p} \\ &= \frac{\mathrm{area}(\triangle)}{(c_4 \mathrm{diam}(\triangle))^2} \leq \frac{(\mathrm{diam}(\triangle))^2}{2(c_4 \mathrm{diam}(\triangle))^2} = \frac{1}{2c_4^2},\end{aligned}$$

since $\mathrm{lfs}_{\mathsf{P}}(\mathsf{p}) \geq \mathrm{lfs}_V(\mathsf{p})$ and by (12.3). As such

$$\#_\triangle(\mathsf{P}, \mathcal{D}) = \int_{\mathsf{p} \in \mathcal{D}} \frac{1}{(\mathrm{lfs}_{\mathsf{P}}(\mathsf{p}))^2}\, d\mathsf{p} = \sum_{\triangle \in \mathcal{M}} \int_{\mathsf{p} \in \triangle} \frac{1}{(\mathrm{lfs}_{\mathsf{P}}(\mathsf{p}))^2}\, d\mathsf{p} \leq \sum_{\triangle \in \mathcal{M}} \frac{1}{2c_4^2} = \frac{|\mathcal{M}|}{2c_4^2},$$

implying the claim. ∎

12.4. The result

THEOREM 12.25. *Given a point set* $\mathsf{P} \subseteq [1/4, 3/4]^2$ *with* $\operatorname{diam}(\mathsf{P}) \geq 1/2$*, one can compute a* 4*-fat triangulation* $\mathcal{M}$ *of* $\mathcal{D} = [0,1]^2$ *in* $O(n \log n + m)$ *time, where* $n = |\mathsf{P}|$ *and* $m = |\mathcal{M}|$*. Furthermore, any* $O(1)$*-fat triangulation of* $\mathcal{D}$ *having* P *as part of its vertices must be of size* $\Omega(m)$*.*

PROOF. The aspect ratio of the output and the running time of the algorithm are guaranteed by Lemma 12.11. The upper bound on the size of the triangulation is implied by Lemma 12.19, and the lower bound of the size of such a triangulation is from Lemma 12.24. Putting everything together implies the result. ■

12.5. Bibliographical notes

Balanced quadtree and good triangulations are due to Bern et al. [**BEG94**]. The analysis we present uses ideas from the work of Ruppert [**Rup95**]. While using this approach to analyze [**BEG94**] is well known to people working in the field, I am unaware of a reference in the literature where it is presented in this way.

The problem of generating good triangulations has received considerable attention, as it is central to the problem of generating good meshes, which in turn are important for efficient numerical simulations of physical processes. One of the main techniques used in generating good triangulations is the method of Delaunay refinement. Here, one computes the Delaunay triangulation of the point set and inserts circumscribed centers as new points, for "bad" triangles. Proving that this method converges and generates optimal triangulations is a non-trivial undertaking and is due to Ruppert [**Rup93**]. Extending it to higher dimensions and handling boundary conditions make it even more challenging. However, in practice, the Delaunay refinement method outperforms the (more elegant and simpler to analyze) method of Bern et al. [**BEG94**], which easily extends to higher dimensions. Namely, the Delaunay refinement method generates good meshes with fewer triangles.

Furthermore, Delaunay refinement methods are slower in theory. Getting an algorithm to perform Delaunay refinement in the same time as the algorithm of Bern et al. is still open, although Miller [**Mil04**] obtained an algorithm with only slightly slower running time.

Recently, Alper Üngör came up with a "Delaunay-refinement type" algorithm, which outputs better meshes than the classical Delaunay refinement algorithm [**Üng09**]. Furthermore, by merging the quadtree approach with the Üngör technique, one can get an optimal running time algorithm [**HÜ05**].