View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Infoscience - École polytechnique fédérale de Lausanne

Solving the Coplanarity Problem of Regular Embedded Triangulations

Laurent Balmelli†‡, Jelena Kovačević† and Martin Vetterli‡

†Mathematical Sciences Research Center
Bell Laboratories, Lucent Technologies
Murray Hill, NJ, USA
{balmelli,jelena}@research.bell-labs.com

‡Laboratoire des Communications Audio-Visuelles
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
{Laurent.Balmelli,Martin.Vetterli}@epfl.ch

Abstract

A regular embedded triangulation is constructed by recursive subdivision starting from a square formed by two triangles and maps a uniform gridded set of vertices. To decimate such a mesh, vertices are removed from the initial set and a new triangulation is constructed on the remaining ones. The coplanarity of the resulting triangle edges is needed, otherwise cracks (also known as T-vertex) appear when the mesh is rendered. In this paper, we first compute and prove the size of the coplanarity problem in the worst case, as well as in expectation. Then, we give a computationally optimal algorithm to solve the problem for a triangulation stored in a linear (or pointerless) quadtree data structure, using the results in [5]. This problem has been studied previously [2, 3, 6, 7, 9, 10], but either partial solutions based on heuristics, or suboptimal algorithms for the general case have been proposed. Moreover, all of these methods are restricted to surface simplification. Our algorithm is valid for regular triangulations modeling surfaces, for surfaces modeled with multiple triangulated patches and finally for regularly tessellated spheres.

1 Introduction

Regular triangulations are standard tessellations of the plane, and are used in many disciplines such as finite element calculation [8] and computer graphics [2, 4, 6]. A regular triangulation is formed by a set of similar triangles, residing on a uniform grid of vertices, as shown in Figure 1(a). A regular triangulation can map a nonuniform grid however, by removing vertices for the initial set and recomputing the set of triangles. Figure 1(b) depicts such construction, where deleted vertices are shown in dotted points. In visualization, regular triangulations provide efficient storage and rendering for surfaces or closed objects. We are particularly interested in regular triangulations having an embedding property. Two triangulations are embedded if all edges of the triangulation with fewer triangles exist in the denser triangulation. In Figure 1 for example, (b) is embedded in (a). These triangulations provide a framework for multiresolution rendering, often used when dealing with large computer graphics models. Embedded triangulations are particularly suited to the construction of adaptive meshes. Adaptivity is obtained when large triangles approximate low curvature regions while small triangles are concentrated around the model de-

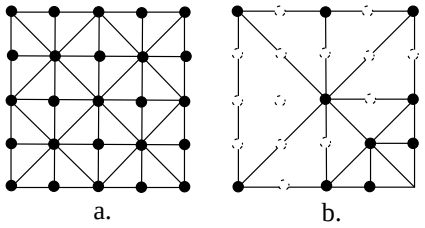

Figure 1: Regular triangulation: (a) mapping of a uniform grid, (b) mapping a irregular set of vertices.

tails. The construction of adaptive regular triangulations is obtained by removing vertices from the initial set used to generate the full resolution model. This operation is correctly carried out only if all the edges in the resulting triangulation are coplanar. Coplanarity is required to ensure that no discontinuities (or cracks) will appear when the model is rendered. Solving the *coplanarity problem* [2, 3, 6, 7, 9, 10] involves computing the set of vertices to be removed along with a desired one in order to avoid discontinuities in the triangulation. It is also important to be able to compute the "true" cost of removing a vertex, or equivalently of merging two triangles. For example, in the simplification algorithm presented in [1], for each vertex we compute those vertices to be removed as well, which is equivalent to solving the coplanarity problem.

The paper is organized as follows: we introduce the class of regular embedded triangulations and their construction. Then, we explain the coplanarity problem and compute its size in the worst case, as well as in expectation. We store the triangulation using a linear (or pointerless) quadtree, and use the results in [5] to construct an algorithm solving the problem in minimal computational time.

2 Regular Triangulations

Consider a set of vertices V defined as

$$V = \{z = z[x, y] \mid [x, y] \in \Omega \subset \mathbb{N}^2\}.$$

This set can be obtained by sampling a bivariate function $f(x, y)$, and then parameterizing the resulting sequence on the integer grid. It can also be obtained by discrete measurements. A regular embedded triangulation for the set V is constructed as follows: we start with the initial four corners and an edge connecting two opposite corners (see Figure 2a). At each subdivision step, denoted l, new vertices are inserted at each triangle hypothenuse midpoint. Note that, in Figure 2, only the triangulations (a),(c) and (e) reside on a uniform grid of vertices. Assuming that each necessary vertex at a particular step is labeled with the step number l, then the *subdivision process* induces a natural hierarchy in the gridded set. In order to conserve the embedding property,

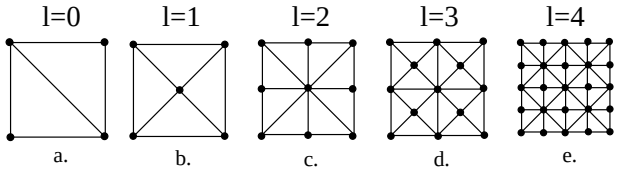

Figure 2: Subdivision process yielding an embedded set of triangulations.

it is important that the vertex used to split a particular triangle is used to recover the same triangle when removed. We illustrate the hierarchy induced by the subdivision process on the gridded set of vertices by splitting the regular grid into the so-called *quincunx* and *cartesian* grids (Figure 3a and 3b). The numbers next to a vertex in the grid indicate at which step of the subdivision process this vertex is inserted. Note that the four vertices at the corners of the grid, forming the two initial triangles, cannot be removed. The grids are actually similar; the quincunx one is just a rotation of the cartesian one by $\frac{\pi}{4}$. Their superposition results in a uniform grid (Figure 3c).

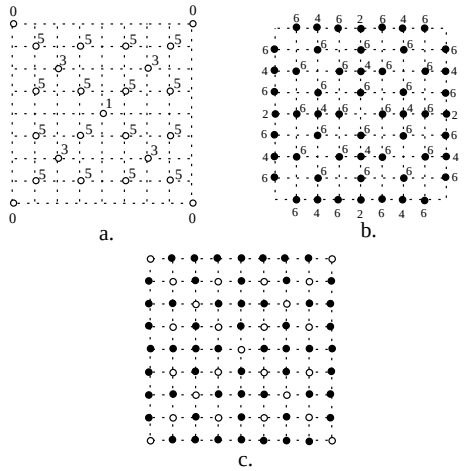

Figure 3: a. Quincunx grid. b. cartesian grid. c. Superposition of both grids. In a. and b. the label next to each vertex corresponds to the step number at which the vertex is inserted during the subdivision process.

3 The Coplanarity Problem

Assume a uniform gridded data set of size $N \times N$ with $N = 2^d + 1$. In the full triangulation obtained at the end of the subdivision process (see for example Figure 2e), pairs of triangles share the same hypotenuse, thus a common midpoint. Removing the midpoint vertex implies merging all triangles attached to it, since otherwise, the removal would induce a shape discontinuity in the rendered structure. Consider the simple example of Figure 4, where the vertex labeled A is considered for removal. The triangles τ_i with $i = 1 \ldots 4$ must be merged jointly in the triangulation in order to avoid a surface crack (Figure 4b). The removal of an arbitrary vertex in an embedded triangulation may induce an additional series of simplifications (see Figure 5). This problem, known as the *coplanarity problem* [9], implies solving a set of constraints for a particular vertex removal. The coplanarity problem for regular triangulations is well-known and has been explored in the literature [2, 3, 6, 7, ?, ?]. In [10], the authors use a data structure called *active border* (see Section 5.1.3 in [9]) and an algorithm to traverse the entire triangulation in order to solve the coplanarity problem. In [3, 6], the authors give algorithms based on a quadtree structure and solve some particular cases of the problem. In [2], the authors use a heuristic approach to avoid cracks when splitting or merging pairs of triangles. More recently, a quadtree structure using pointers as well as an algorithm to solve the coplanarity problem with a computational complexity linear in the size of the tree was proposed in [7]. Unfortunately, the authors neither give an analysis of the algorithm nor prove their result. Moreover, the algorithm is suboptimal since we will show that, in expectation, the complexity of the problem does not depend on the tree size. Finally, our data structure does not require any pointer or any explicit index for the tree nodes and is then more efficient in memory.

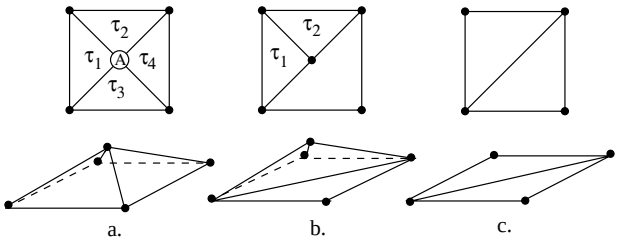

Figure 4: Surface Crack (T-Vertex).

We now describe the coplanarity problem and compute its size. The example in Figure 5 illustrates the constraints induced by the removal of an arbitrary vertex (labeled A). Once removed, additional vertices must be removed at the same time. Thereafter, the edges in the resulting triangulation are coplanar (Figure 5b).

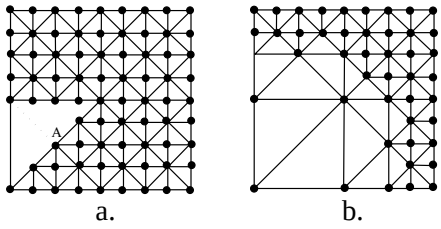

Figure 5: (a) Removal of the vertex A (the resulting triangulation is not coplanar). (b) Coplanar triangulation.

The above example illustrates the resolution of constraints on a bounded surface. For a sufficiently large triangulation, if the removed vertex is located near the boundaries, the set of vertices to remove is expected to be smaller than for a vertex (having the same label l) located in the middle of the triangulation. If no more vertices need to be removed because of the boundary, we say that the *spread* (of constraints) is not complete (see for example Figure 5b). Figure 6 depicts an example of a complete spread. The number of triangles contained in the spread is proportional to the size of the initial triangle merged. The set of all the triangles that need to be merged at the same time as the desired triangle is called the *merging domain* of this triangle. Thus, the merging domain is related to the vertex used to split a particular triangle during the subdivision process. Consider the

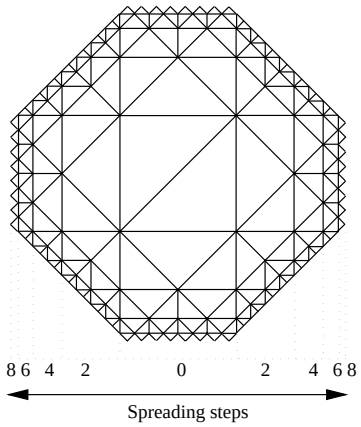

Figure 6: Complete spread of triangulation constraints.

case depicted in Figure 7, where a triangulation is modeled with multiple patches[1]. In the figure, 9 patches of size 17×17 (thick lines) are used to generate the model. In this example, the central patch is completely merged after the removal of the unique vertex labeled $l = 1$ (on the cartesian grid). As a result, the merging domain spreads to the 8 neighbors patches. To determine the size of the

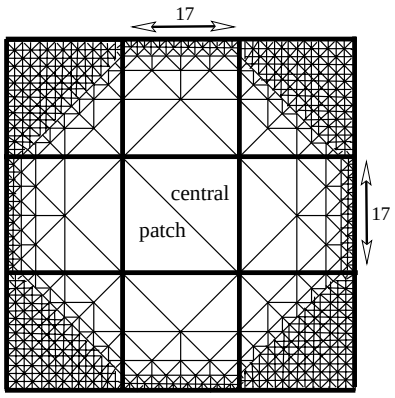

Figure 7: Spreading of the merging domain through triangulated patches.

problem, we need to count the number of triangles in the merging domain. We first compute the size for i spreading steps (in Figure 6, $i = 8$). Note that the merging domain corresponding to a vertex of the quincunx grid is similar to the one in Figure 6 with a rotation of $\frac{\pi}{4}$ or $-\frac{\pi}{4}$. To compute the size of the complete merging domain, we construct a tree scaffolding for the triangulation such that each triangle corresponds to a tree node, as depicted in Figure 8. Computing the size of the merging domain is then equivalent to evaluating the number of tree nodes. Starting

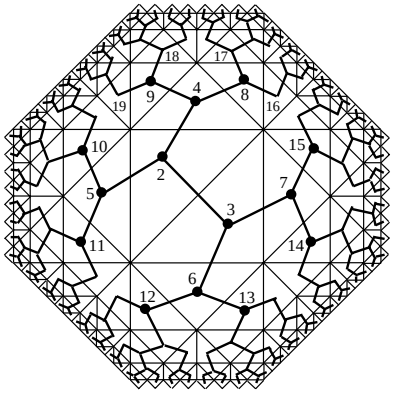

Figure 8: Scaffolding of a triangle merging domain.

at the tree node 4 in Figure 8, we can see that the subtree rooted at this node is balanced[2] until nodes 16,17,18 and 19. Afterwards, the branches become unbalanced. We give below the merging domain size for i spreading steps and prove the equations in the Appendix.

Proposition 3.1. Merging Domain Size *We denote by $m(i)$ the size of the complete merging domain for i spreading steps. It is given by*

$$m(i) = \begin{cases} 2^{i+1} - 2 & 1 \leq i \leq 4, \\ 32(2^{i-\lfloor \frac{i-4}{2} \rfloor - 4} + \\ \quad 3 \cdot 2^{\lfloor \frac{i-4}{2} \rfloor - 1}) - \\ \quad 8i - 18 & i > 4. \end{cases} \tag{1}$$

In the merging domain of Figure 6, the merging domain spreads completely in eight steps; it thus contains

$$32(2^{8-\lfloor \frac{8-4}{2} \rfloor - 4} + 3 \cdot 2^{\lfloor \frac{8-4}{2} \rfloor - 1}) - 8 \cdot 8 - 18 = 238 \text{ triangles.}$$

We now express i as a function of the total number of triangles, denoted n, and the index

[1] For example, when only part of the data is available, or the data is simply too large to be modeled with a unique triangulation.

[2] A tree is balanced when each nonleaf node has exactly two children.

l labeling the steps of the subdivision process. As seen previously, an embedded triangulation can only be constructed on an $N \times N$ grid with $N = 2^d + 1$, and that the grid contains $n = 2 \cdot 4^d$ triangles. Moreover, l satisfies $0 < l < 2d + 1$. We can therefore compute $m(l, n)$ with $i = 2\log_4 n - l$.

$$m(l,n) = \begin{cases} 2^{1-l}n - 2 & 1 \leq i \leq 4, \\ 32(2^{-(l+4)}n \cdot c(l,n)^{-1} + \\ \ 3 \cdot c(l,n) - \\ \ 16(\log_4 n - \frac{l}{2}) - 18 & i > 4, \end{cases} \tag{2}$$

where $l > 0$, $n = 2 \cdot 4^d$, $d > 0$ and $c(l,n) = 2^{\lfloor \log_4 n - \frac{l+4}{2} \rfloor}$. We first compute the problem in the worst case (depicted in Figure 7), namely when removing the unique vertex labeled $l = 1$. We assume therefore that for each vertex l, the merging domain is complete, and therefore that $l < k \Rightarrow m(l,n) > m(k,n)$. For a bounded surface, this equation is likely to be true for any l, k such that $0 << l < k$. This case is simply obtained by replacing $l = 1$ in (2)

$$m(1,n) = \begin{cases} n - 2 & 1 \leq i \leq 4, \\ n \cdot c(n)^{-1} + 48 \cdot c(n) - \\ \ 16\log_4 n - 26 & i > 4, \end{cases} \tag{3}$$

where $n = 2 \cdot 4^d$, $d > 0$ and $c(n) = 2^{\lfloor \log_4 n - \frac{5}{2} \rfloor}$. We need now to bound the term $c(n)$ to derive the problem complexity, thus

$$\frac{1}{8}\sqrt{\frac{n}{2}} \leq c(n) \leq \frac{1}{4}\sqrt{\frac{n}{2}}. \tag{4}$$

We can now lower bound (3) to obtain:

$$\begin{aligned} m(1,n) &\geq 4\sqrt{n} + 3\sqrt{2}\sqrt{n} - 16\log_4 n - 26, \\ &\approx 8.24\sqrt{n} - 16\log_4 n - 26, \\ &\in \Omega(\sqrt{n}). \end{aligned} \tag{5}$$

Similarly, we upper bound (3):

$$\begin{aligned} m(1,n) &\leq 2^{\frac{7}{2}}\sqrt{n} + 6\sqrt{2}\sqrt{n} - 16\log_4 n - 26, \\ &\approx 19.79\sqrt{n} - 16\log_4 n - 26, \\ &\in O(\sqrt{n}). \end{aligned} \tag{6}$$

With (5) and (6), we have shown that, in the worst case, the coplanarity problem has complexity $\Theta(\sqrt{n})$.

To compute the complexity in expectation, we use the following observations: the embedding property implies that for $i \geq 0$

$$m(2^{i+1} - 1, n) = m(1, \frac{n}{4^i}), \tag{7}$$

$$m(2^{i+1}, n) = m(2, \frac{n}{4^i}), \tag{8}$$

where (7) refers to the merging domain of a vertex of the cartesian grid while (8) refers to the one of the quincunx grid. Moreover, the total number of triangles n is asymptotically twice that the number of elevation N^2, as shown below

$$\begin{aligned} \lim_{d\to\infty} \frac{n}{N^2} &= \lim_{d\to\infty} \frac{2 \cdot 4^d}{(2^d + 1)^2}, \\ \lim_{d\to\infty} \frac{2 \cdot 4^d}{4^d + 2^{d+1} + 1} &= 2. \end{aligned} \tag{9}$$

Therefore $n/2$ is a good limit value for the number of vertices. We use the approximation 4^{i+1} for the number of vertices of the quincunx grid at step $l = 2i$ of the subdivision process, which is true for $i >> 0$. The number of vertices of the cartesian grid is 4^i. Finally, recall that $d = log_4(n) - 1/2$. The complexity of the problem in expectation is then

$$\begin{aligned} Em(n) &= \frac{2}{n}\Big(\sum_{i=0}^{d-1} 4^i m(1, \frac{n}{4^i}) + \sum_{i=0}^{d-1} 4^{i+1} m(2, \frac{n}{4^i})\Big), \\ &= \frac{2}{n}\Big(\sum_{i=0}^{d-1} 4^i \Theta(\sqrt{\frac{n}{4^i}}) + \sum_{i=0}^{d-1} 4^{i+1} \Theta(\sqrt{\frac{n}{4^i}})\Big), \\ &= \frac{2}{n}\Big(\sum_{i=0}^{d-1} 4^i c_1 \cdot \sqrt{\frac{n}{4^i}} + \sum_{i=0}^{d-1} 4^{i+1} c_2 \cdot \sqrt{\frac{n}{4^i}}\Big), \\ &= \frac{2^{d+\frac{3}{2}}}{n}(c_1 + 4c_2) \sum_{i=0}^{d-1} 4^{\frac{i}{2}}, \\ &= \frac{(c_1 + 4c_2)}{n} \underbrace{2^{d+\frac{3}{2}}(2^d - 1)}_{(\star)}. \end{aligned} \tag{10}$$

Finally, we need to compute the order of the term $(\star)$

$$2^{d+\frac{3}{2}}(2^d - 1) = \sqrt{2}n - 2\sqrt{n} \ \in \Theta(n), \tag{11}$$

yielding

$$Em(n) = \frac{\Theta(n)}{n}, \quad \in \Theta(c). \tag{12}$$

which proves that the coplanarity problem has constant size in expectation. We conclude with the following proposition:

Proposition 3.2. Size of the Coplanarity Problem
The coplanarity problem for a regular embedded triangulation of size n has size $\Theta(\sqrt{n})$ in the worst case and constant size $\Theta(c)$ in expectation.

4 Optimal Algorithm

Assume that the triangulation is stored using a quadtree, as depicted in Figure 9. Each quadtree node contains a subset of the triangulation. To each nonleaf node corresponds all the triangles contained in its subtree. In

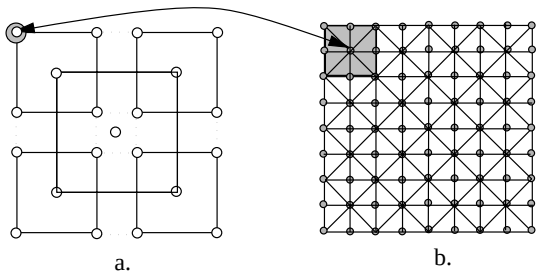

Figure 9: a. Embedded regular triangulation. b. Corresponding quadtree.

[5], we gave a method to traverse such a quadtree in constant time. We use this result to build a computationally optimal algorithm to solve the coplanarity problem. Such algorithm has a computational complexity of the order of magnitude equal to the size of the problem. In the previous section, We computed the size of the coplanarity problem. We explain now how to build an algorithm visiting only and exactly once the nodes of the quadtree containing triangles of the merging domain.

The basic idea of the algorithm relies on "sweeping" the surface of the merging domain, starting from the central block, corresponding to node p in Figure 10(a). Since the sweeping of the merging domain surface can be done in an arbitrary order, we do not state a specific algorithm here. In Figure 10(b), we show a possible way to traverse a region of the merging domain. The indices in the figure correspond to the nodes in Figure 8. In Figure 11, we present an example of decimated regular embedded triangulation. The algorithm insures that, after decimation, all triangle edges are coplanar. Our algorithm applies to all constructions described in [5], namely to triangulations constructed with multiple patches (like in Figure 7) and to tessellated spheres obtained by projecting the vertices of an octahedron (like in Figure 12). We provide

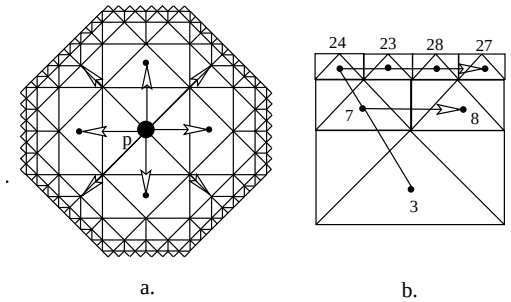

Figure 10: Illustration of the algorithm. a. Starting at node p, the algorithm traverses the surface. b. An example of node traversal.

a Java applet illustrating the algorithm at `http://lcavwww.epfl.ch/Triangulation/` as well as code in C and Java.

Appendix

Proof of Proposition 3.1 To prove (1), we need to find equations describing the unbalancing of the tree in Figure 8. The tree in Figure 8 is balanced up to the fourth subdivision step (that is, from node 2 to node 18). A portion of the same tree is shown in Figure 13, where the gray region denotes the unbalanced part. The indices in the figure match the ones in Figure 8. Level 5 in the tree is the first level where the tree is unbalanced. Define $j = i - 4$ for $i > 4$. Observe that at level

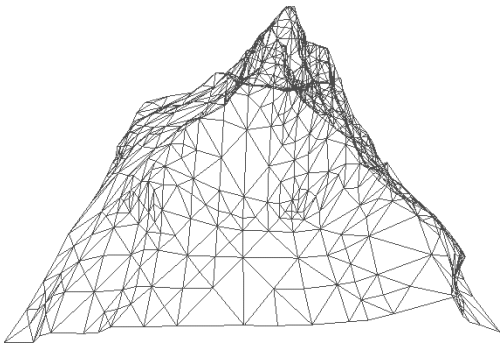

Figure 11: Wireframe representation of a decimated embedded regular triangulation. The data set used is a digital elevation model of the mount Matterhorn in Switzerland.

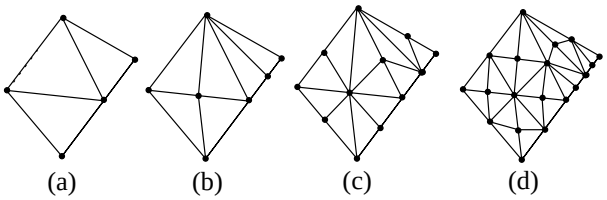

Figure 12: (a)-(d): Regularly tessellated octahedron.

$2j-1$ with $j>0$, we have 2^j nodes with both children and $2^{j+1}-2$ nodes with one single child. Moreover, for every level $2j$ with $j>0$, there are 2^{j+1} nodes with both children and also $2^{j+1}-2$ nodes with one single child. We compute two separate sums, one for the odd indices, denoted by $o(p)$, and one for the even j indices, denoted by $e(q)$, to count the total number of nodes. Note that p and q denote the number of odd and even indices, respectively.

$$\begin{aligned} o(p) &= \sum_{j=1}^{p}(2^{j+1}+2^{j+1}-2), \\ &= \sum_{j=1}^{p}(2^{j+2}-2), \\ &= 4(2^{p+1}-2)-2p, \\ &= 2^{p+3}-2p-8. \end{aligned} \tag{13}$$

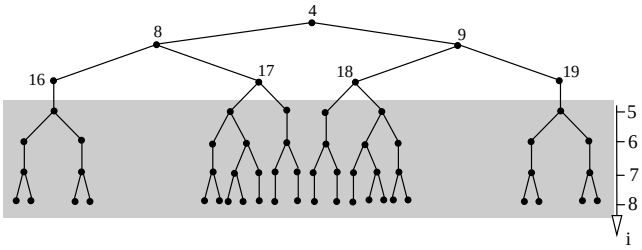

Figure 13: Portion of the tree scaffolding the merging domain in Figure 8.

$$\begin{aligned} e(q) &= \sum_{j=1}^{q}(2^{j+2}+2^{j+1}-2), \\ &= 4\sum_{j=1}^{q}2^j + 2(\sum_{j=1}^{q}2^j - 1), \\ &= 4(2^{q+1}-2)+2(2^{q+1}-2-q), \\ &= 2^{q+3}+2^{q+2}-2q-12. \end{aligned} \tag{14}$$

To select between (13) and (14) for an arbitrary index j, we define

$$p = j - \lfloor \frac{j}{2} \rfloor \text{ and } q = \lfloor \frac{j}{2} \rfloor. \tag{15}$$

We can now generate a single sum to count $c(j)$, the nodes in the unbalanced part of the tree, with

$$\begin{aligned} c(j) &= 2^{p+3}-2p+2^{q+3}+2^{q+2}-2q-20, \\ &= 2^{j-\lfloor \frac{j}{2} \rfloor+3}-2(j-\lfloor \frac{j}{2} \rfloor)+ \\ &\quad 2^{\lfloor \frac{j}{2} \rfloor+3}+2^{\lfloor \frac{j}{2} \rfloor+2}-2\lfloor \frac{j}{2} \rfloor-20, \\ &= 8(2^{j-\lfloor \frac{j}{2} \rfloor}+2^{\lfloor \frac{j}{2} \rfloor})-2j+4\cdot 2^{\lfloor \frac{j}{2} \rfloor}-20, \\ &= 8(2^{j-\lfloor \frac{j}{2} \rfloor}+3\cdot 2^{\lfloor \frac{j}{2} \rfloor-1})-2j-20. \end{aligned} \tag{16}$$

Finally, the sum of nodes for the first balanced levels in the complete tree scaffolding is $2^5-2=30$. Moreover, it has four unbalanced structures of size $c(j)$. Let i denote the tree levels along which the nodes are summed. Since $j=i-4$, for $i>4$, we have that the total number of nodes, or equivalently the total number of triangles in the merging domain, is equal to

$$m(n) = 32(2^{i-\lfloor \frac{i-4}{2} \rfloor-4}+3\cdot 2^{\lfloor \frac{i-4}{2} \rfloor-1})-8i-18, \tag{17}$$

which concludes the proof. ◇

References

[1] L. Balmelli, S. Ayer, and M. Vetterli. Efficient algorithms for embedded rendering of terrain models. *Proc. Int. Conf. Image Processing (ICIP)*, October 1998.

[2] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein. Roaming terrain: Real-time optimally adapting meshes. *IEEE Visualization*, 1997.

[3] B. Von Herzen and A.H. Barr. Accurate triangulation of deformed, intersecting surfaces. *Computer Graphics 21, also Proceeding of ACM SIGGRAPH 87*, 21(4):103–110, July 1987.

[4] D. Koller, P. Lindstrom, W. Ribarsky, L. Hodges, N. Faust, and G. Turner. Virtual gis: A real time 3d geographic information system. *Proceedings Visualization 95*, pages 94–100, 1995.

[5] L.Balmelli, J.Kovačević, and M. Vetterli. Quadtree for embedded surface visualization: Constraints and efficient data structures. *Proc. Int. Conf. Image Processing (ICIP)*, October 1999.

[6] P. Lindstrom, D. Koller, W. Ribarsky, L.F. Hodges, N. Faust, and G.A. Turner. Real-time continous level of detail rendering of height fields. *ACM SIGGRAPH*, pages 109–118, 1996.

[7] R. Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. *Proceedings of IEEE Visualization*, 1998.

[8] M.C. Rivara. Mesh refinement processes based on the generalized bisection of simplices. *SIAM Journal on Numerical Analysis*, 21:604–613, 1984.

[9] H. Samet. *Application of Spatial Data Structures*. Addison-Wesley Publishing Company, 1989.

[10] M. Tamminen and F.W. Jansen. An integrity filter for recursive subdivision meshes. *Computer Graphics 9*, 4(1):351–363, 1985.