The Society for engineering in agricultural, food, and biological systems

Paper Number: 01-3072
An ASAE Meeting Presentation

Quadtree-Based Triangular Mesh Generation for Finite Element Analysis of Heterogeneous Spatial Data.

Prabhakar Reddy GVS

Biological Resources Engineering Department

University of Maryland at College Park

College Park, MD, 20742.

Email: reddyg@eng.umd.edu

Hubert J. Montas

Biological Resources Engineering Department

University of Maryland at College Park

College Park, MD, 20742.

Hanan Samet

Computer Science Department

University of Maryland at College Park

College Park, MD, 20742.

Adel Shirmohammadi

Biological Resources Engineering Department

University of Maryland at College Park

College Park, MD, 20742.

Written for presentation at the
2001 ASAE Annual International Meeting
Sponsored by ASAE
Sacramento Convention Center
Sacramento, California, USA
July 30-August 1, 2001

Abstract. Applying mathematical models to practical situations often requires the use of discrete geometrical models of the solution domain. In some cases destructive measurements of the objects under examination is acceptable, but in several areas of research the measurements comes from imaging techniques such as X-ray, computer assisted tomography (CAT), magnetic resonance imaging (MRI), satellite imagery, or aerial photographs. A crucial preprocessing step for such analysis involves the extraction of measurements/features from these images, which form the basis of geometrical models and finite element mesh. In this paper, we describe a simple algorithm for triangulating the solution domain represented in images without a need for such prior feature extraction, albeit such a step may reduce the size of the resulting mesh. The proposed algorithm generates quality triangular meshes with: (a) provably good angle bounds between 26.565^{o} and 90^{o}, and (b) an aspect ratio of at most 2.5.

The proposed mesh generation algorithm (imageMesher) extends the mesh generation technique of Bern et al. (1990) to images as input. Previous algorithms with shape and size bounds have all been based on triangulating domains that are either: (a) vertex set, (b) lines, (c) polygons, or (d) planar straight line graphs (PSLGs). The proposed algorithm matches their bounds, but uses a fundamentally different kind of input. The implementation of the algorithm is discussed and the theoretical bounds on the size and shape of the triangular patches are evaluated. As an intermediate result, we also describe an improved algorithm for constructing balanced quadtree. Finally, we illustrate real-time applications of the proposed approach, which demonstrates its ability to use the solution domain described in images to fit directly into the finite element analysis.

Keywords. finite element method (FEM), mesh generation, quadtree, balanced quadtree, Delaunay triangulation, image analysis.

Quadtree-Based Triangular Mesh Generation for Finite Element Analysis of Heterogeneous Spatial Data.

Prabhakar Reddy G.V.S., Hubert J. Montas, Hanan Samet and Adel Shirmohammadi

Biological Resources Engineering Department
(except H. Samet: Computer Science Department)
University of Maryland at College Park
College Park, MD, 20742.

Introduction

A necessary early step in finite element method is mesh generation, and the most versatile type of two-dimensional mesh is an unstructured triangular mesh. A *mesh*, in general, is a spatial discretization of geometric domain (usually, $\Re^2$ and $\Re^3$) into small simple shapes (*simplices*), typically triangles or quadrilaterals in two-dimension (2D) and tetrahedra or hexahedra in three-dimension (3D). Since late 1980's, there has been tremendous advancement in mesh generation. Presently, even the most complicated domains can be meshed gracefully with proven theoretical bounds using the algorithms described in (Bern et al., 1990; Chew, 1989; Mitchell, 1994; Preparata and Shamos, 1985; Ruppert, 1993 and 1995). As far as two dimensional triangular unstructured mesh generation is concerned, significant amount of work has been be dedicated on triangulating domains like: (1) vertex sets, (2) lines, (3) simple polygons, and (4) planar straight line graphs (*PSLGs*). Since, most of the solution domains associated with physical processes can be represented using a combination of either: points, lines, polygons, or PSLGs these algorithms can be used for generating quality meshes, thus, facilitating the numerical simulation of such processes using numerical techniques.

However, with the advent of the sophisticated imaging techniques research community begun to focus on using images to study several physical phenomena. For example, the growth of tumor - using computer assisted tomography (CAT scans), spatial variation of geophysical parameters - using satellite imagery, ground penetrating radar, etc. The absence of appropriate mesh generation algorithm, which respects the heterogeneous geometric domains represented in images, has compelled researchers to use a regular grid for spatial discretization in their numerical simulations. (We are unaware of any theoretical papers on this subject). In most instances, the pixels of the raster images are used as the underlying grid in such numerical simulations (Tracqui et al., 1995; Montas et al., 2000). The motivation of this paper has been the will to design a robust and efficient algorithm capable of handling domains of arbitrary heterogeneity represented in (preprocessed) digital images. The ultimate goal is to provide a quality triangular mesh generation tool, which can adapt itself to the heterogeneous boundaries in images.

This paper is divided into four sections. Section 1 recalls the terminology used with the quadtree-based decomposition and summarizes the general scheme of the quadtree-based Delaunay mesh generation. Section 2 proposes the improved balanced quadtree

algorithm along with *imageMesher* algorithm. The algorithm we describe in this paper extends the algorithm of Bern et al. (1990) for triangulating images, unlike PSLGs as the input. We then prove that the all the triangles in the output have angles bounded between 26.56^o and 90^o and with aspect ratio[1] of at most 2.5. As an intermediate result, we also describe an improved algorithm for building a balanced quadtree. In the penultimate section, two application examples are described to emphasize the practical utility of the proposed algorithm. A brief section concludes the paper by mentioning the possible extensions of this work.

1.Background

In this section, we recall the basic terminology related to quadtree-based Delaunay triangulation technique. We also discuss the technique of generating quality triangular mesh using quadtree-based techniques, first introduced by Bern et al. (1990). The ensuing subsection(s) expands those topics pertinent to the content of this paper. For further details, the two books by Samet (1990(a) and 1990(b)) provide extensive information of the various types of quadtrees and their applications. See the lecture notes of Shewchuk (1999) and the chapter of Bern and Plassmann (1999) for various methodologies and recent developments in unstructured mesh generation.

1.1.Terminology

According to Samet (1990(a)), quadtree is proposed as a representation for raster images because its hierarchical nature facilitates the performance of a large number of operations (e.g. quadrant creation, finding cells adjacent to a given cell in a given direction, finding cell(s) that contain a particular value, etc.) close to optimal computational cost. The basic concept of the quadtree decomposition consists of enclosing the domain Ω into a bounding box $B(\Omega)$, usually a square, corresponding to the *root* of the spatial decomposition tree. This box is subdivided into four equally sized *sons*, one in each of the four directions: North-East (NE), North-West (NW), South-West (SW), and South-East (SE), each of which is in turn recursively subdivided until a *stopping criterion*[2] is reached based on the local geometry of the domain (e.g., the local curvature of the boundary) or user-defined maximum refinement. Any node that is not subdivided is a *leaf* node and the subdivided cells are *non-leaf* nodes. The *corner* of a node is the vertex of the square and the edges connecting consecutive corners are the *sides* of the node. The *size* of a node *c* is the length of the side of *c*. Two nodes are said to be *adjacent* if they share an edge and any node in the quadtree has four possible *edge-neighbors*, one in each of the four cardinal directions (North - N, East - E, West - W, and South - S). We say that the side of the node is split if either of the neighboring nodes sharing it is split. The *level* of a node corresponds to its depth in the tree structure, i.e., the number of subdivisions required to obtain the node. It is conventional practice to represent the root at level 0.

[1] Aspect ratio of an element is the ratio of its maximum to minimum width, width being the distance between parallel supporting hyper-planes. For a triangle, the aspect ratio of a triangle is the length of the longest edge divided by the length of the shortest altitude.

[2] In *imageMesher* algorithm, a node is subdivided until all the pixel(s) inside it have the same intensity/value.

At each stage of recursive decomposition, any node can be subdivided into four sub-cells. Hence, the resulting decomposition of the tree may be quite unbalanced and a triangulation of such an unbalanced tree could result in ill-shaped elements in the final output mesh. In order to overcome this, it is a normal practice to impose the *balancing condition*: no node in the quadtree should be adjacent to one less than one-half its size. This condition is known as *2:1 rule*, first introduced by Yerry and Shephard (1983). Figure 1 shows an example of a quadtree subdivision that has been balanced after applying the 2:1 rule. Solid lines show the original quadtree subdivision whereas its refinement is shown by dotted lines. From Figure 1, one may concluded that the complexity of a balanced quadtree subdivision is quite higher than that of its unbalanced version. But it has been proved that the balancing can be done efficiently and that a balanced quadtree contains no more than the eight times as many nodes as its non-restricted counterpart (Moore, 1992). Restricted quadtrees were initially used in terrain modeling and computer graphics (Herzen and Barr, 1987). However, investigators realized that balanced quadtrees could be a useful intermediate step towards generating quality triangular meshes (Bern et al., 1990; Tanaka 95).

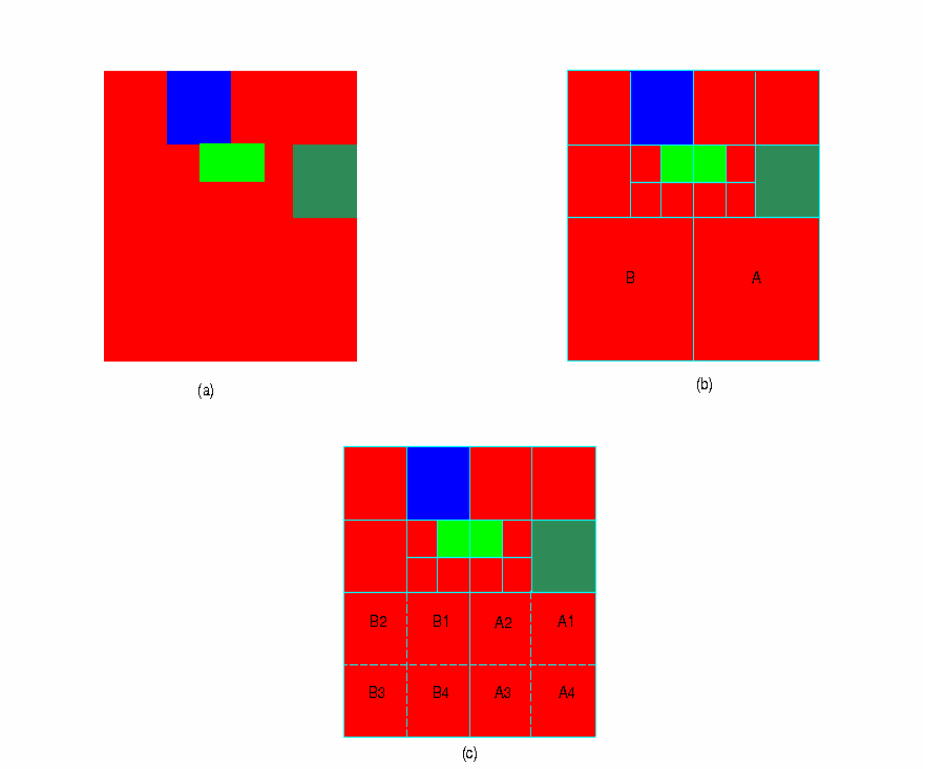


Figure 1. An example illustrating the region quadtree and balanced quadtree decomposition of an image. (a) A 16 x 16 image, (b) its quadtree decomposition, and (c) balanced quadtree decomposition after the imposing 2:1 rule. Notice that the nodes labeled A and B in (b) are divided into four sub-blocks in (c). The original linear quadtree subdivisions are shown as solid lines whereas the refinement is shown as dotted lines.

The triangulation phase usually involves the well-known *Delaunay Triangulation*. In 2-D, the Delaunay triangulation of a set of vertices **{V}** is a set of triangles **{T}**, whose: (a) vertices collectively are **{V}**, (b) interiors do not intersect with each other, (c) union is the convex hull of **{V}**, and (d) every triangle in the **{T}** intersects only at the vertices. Figure 2.1(b) shows Delaunay triangulation of a 2D vertex set shown in Figure 2.1(a). We can also define Delaunay triangulation of **{V}**, first introduced by Delaunay (1934), as the graph defined by the *empty circle condition*: a triangle *abc*, with vertices v_a v_b v_c, appears in Delaunay triangulation **DT**(**V**) if and only if its circumcircle encloses no other points of **{V}**. Figure 2(c) shows the triangulation satisfying empty-circle condition. However, there is an exception for the latter definition when the points lie at special position: if an empty circle passes through four or more points of **{V}**, we can complete the triangulation arbitrarily. Figure 3 shows the exceptional case, with six cocircular vertices, and the various possible arbitrary triangulations. Hence, **DT**(**V**) is a triangulation of the convex hull of **{V}** and is unique if any only if no four points, or more, in the given vertex set, **{V}**, are co-circular. Another promising feature of Delaunay triangulation is: of all the possible triangulations for a given vertex set Delaunay triangulation is the only triangulation that maximizes the minimum internal angles. Since it is desirable to have the triangles in the mesh closer to equilateral triangle, Delaunay triangulation has been the most celebrated approach in 2-D triangular mesh generation. There are several Delaunay triangulation algorithms available today, some are theoretically elegant and some are better for practical implementation. Here we cite the well-known algorithms:

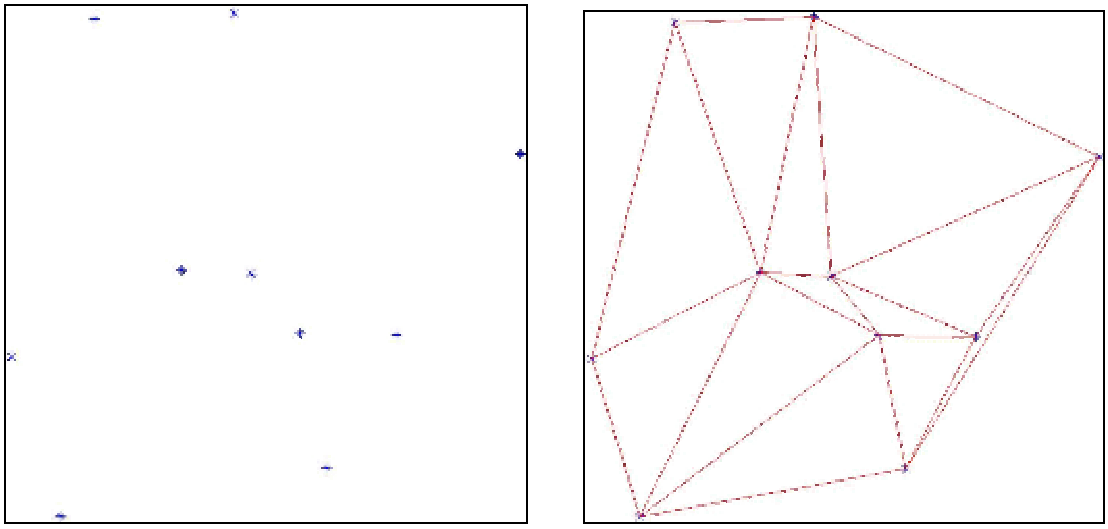
- *Edge Flipping* (Sibson, 1973).
- *Divide-and-Conquer* (Shamos and Hoey, 1975; Guibas and Stolfi, 1985; Lohner, 1988; Chew, 1989)
- *Alternating Divide-and-Conquer* (Dwyer, 1987)
- *Sweepline* (Fortune, 1987)
- *Regular Grid and Sparse Matrix* (Fang and Piegl, 1992 and 1993)
- *Incremental Insertion* (Bowyer, 1981; Watson, 1981; Joe, 1991,1993, and 1995; Rajan, 1994)
- *Randomized Incremental Insertion* (Seidel, 1972; Clarkson and Shor, 1989; Chew, 1990; Guibas et al., 1992; Devillers, 1998)

Delaunay triangulation by itself does not generate a satisfactory mesh because of the following reasons: (1) elements of poor quality may appear, and (2) input boundaries may fail to appear in the final mesh. Both these problems have been treated in the literature. The former problem is typically dealt by adding additional vertices at the circumcenters and centriods of the ill-shaped elements. It is at times also treated with the advancing front approach (Barth and Jesperson, 1988; Mavriplis, 1991).

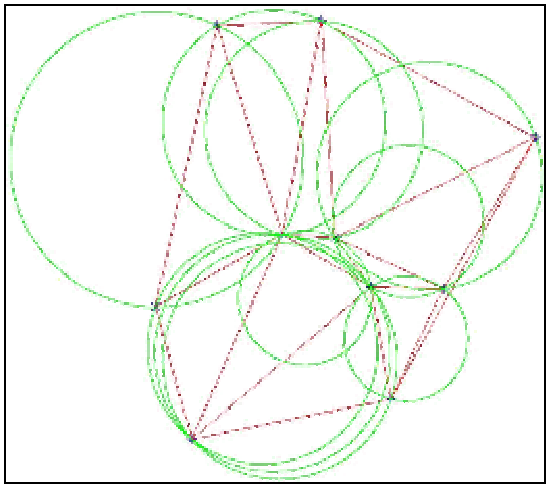
1.2.Quadtree[3] based mesh generation

The quadtree mesh generator due to Bern et al. (1990) starts by enclosing the entire domain, Ω, inside an axis-aligned square (2^n x 2^n dimension). The *provably good mesh generation algorithm* recursively divides each node(s) until each leaf node contains at most one connected component of the domain's boundary, with at most one vertex. This splitting phase was improved (Mitchell and Vavasis, 1992) by "cloning" those squares

[3] In this section, quadtree refers to a point/line quadtree depending on the nature of the input.



(a) (b)



(c)

Figure 2. Illustration of Delaunay triangulation of a vertex set. (a) Vertex set. (b) Delaunay triangulation of the vertex set. (c) Delaunay triangulation with *empty-circumcircles*.

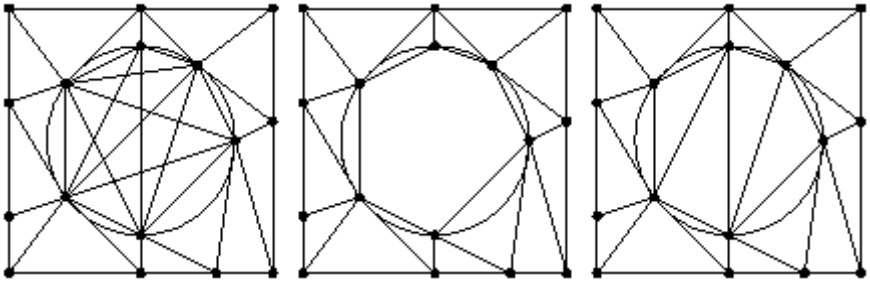


Figure 3.Three ways to define the Delaunay diagram in the presence of cocircular vertices. (a) Include all Delaunay edges, even if they cross. (b) Exclude all crossing Delaunay edges. (c) Choose a subset of Delaunay edges that forms a triangulation.

that intersects Ω in more than one connected component, so that each copy contains only a single connected component of the domain. The algorithm then splits squares near the vertices of Ω two more times, so that each vertex lies within the buffer zone of equal size squares. Quadtree squares are then wrapped and cut to conform the boundary. Finally, the cells of the wrapped quadtree are triangulated so that all angles are bounded away from 0^o. Every triangle in the mesh generated using this technique will have an aspect ratio of at most 4 and the number of triangles in the output will be a constant factor of *optimal*: the minimum number of triangles in *any* triangulation of the given input achieving the same aspect ratio bound.

This was the first triangulation technique that guaranteed theoretical size optimality and bounded aspect ratio. As first presented, the algorithm assumes Ω to be a polygon with holes. However, the approach can be easily adapted for multiple polygons and curved domains. In fact, this approach handles curved domains more gracefully than the *Delaunay refinement algorithm* approach (Ruppert, 1993 and 1995), because the splitting phase can automatically conform to the curvature of the domain.

Neugebauer and Diekmann (1996) extended the results of Bern et al. (1990) by replacing the squares of the quadtree with rhomboids so that the triangles in the final mesh tend to be nearly equilateral. Assuming that there are no small input angles, polygonal domains with polygonal holes and isolated interior points can be triangulated with all the angles between 30^o and 90^o. Remarkably, provably good mesh generation algorithm has been extended to polyhedra of arbitrary dimensionality based on octrees (and their higher dimensional brethrens), which triangulates polyhedra producing size-optimal meshes with guaranteed bounds on element aspect ratios (Mitchell and Vavasis, 1992, 1996 and 2000). However, the generalization to more than two dimensions is quite intricate and the theoretical bounds on element quality are not strong enough to be entirely satisfying in practice.

2. imageMesher

This section expands on the different procedures, data processing steps, and general setup of triangular mesh generation for images. The entire process of mesh generation for images can be subdivided into four phases: (a) *building quadtree from a raster image*, (b) *balancing the quadtree by imposing the 2:1 rule*, (c) *triangulation of the balanced quadtree*, and (d) *post-processing*. These steps are similar algorithm given by Bern et al. (1990) for PSLGs. For further discussion, we assume that: (a) the input of the problem is an arbitrary raster image[4] *I* which is of dimension [0:U] x [0:U], where $U = 2^n$ for some positive integer *n*, and (b) the compression technique of the image is known and can conveniently extract the pixel value. Typically, the input image that comes from an imaging device is preprocessed using: (a) commonly available picture editors and/or (b) image analysis toolbox. However, this topic is not dealt in this paper. For details on preprocessing of images, refer to Montas et al. (2001). It should be mentioned here that

[4] A raster image is generally defined to be a rectangular array of regularly sampled values, known as pixels. Each pixel (picture element) has a value associated with it, generally specifying a color, in which the pixel should be displayed.

the type of quadtree referred in all the ensuing sections is a *region* quadtree, unless mentioned otherwise.

2.1 Building a quadtree from raster

The problem of converting a raster-scanned image into a linear quadtree has been dealt in literature quite extensively (Samet, 1980(a), 1980(b), 1981, 1984(a), and 1990(a); Burton, 1986;Goel and Venkatesh, 1991; Sivan, 1996). The implementation we use is a variant of *ALGORITHM G-V* (Goel and Venkatesh, 1991) with appropriate modifications made for accommodating a color image as the input instead of a binary image. The modified ALGORITHM G-V, hereafter referred to as *MGV* (Image *I*), takes *I* as the input and returns a linear quadtree, *QT*(*I*). In view of the constraints of the space, the algorithm has been skipped.

2.2 Building a balanced quadtree

In this phase, we impose the *2:1 balancing rule* on the quadtree constructed from the input image. The balanced quadtree for an image, *BQT*(*I*), is generated from *QT*(*I*) using a modified algorithm proposed by Sivan (1996). See also the algorithm proposed by Sivan and Samet (1992). The pseudocode is described in Figure 4.

The procedure *BalancedQuadtree* takes a linear quadtree constructed from an raster image as an input and then loads the entire collection of leaf nodes of size greater than two and less than the size of root into a linear list, *L*. The procedure then enters a loop until the list *L* is empty. During each cycle of the loop, the procedure pops out the top-most node, μ, from *L* and fetches all its edge-neighbors into another linear list, *NL*. The next step in the procedure performs a check for 2:1 balancing rule on the current node μ. If the node has to be split, then μ is made as a non-leaf node and its four newly created sons are added to the initial linear list *L*. Before inserting the newly created nodes into *L*, a size check on μ is performed to ensure that the only potential nodes are being stored in it. Since, μ has been split the neighbors (stored in the linear list *NL*) are also checked for 2:1 balancing rule. The above process if repeated for all those neighbors in the list *NL* of size greater than two and less than the size of root node.

The *BalancedQuadtree* is essentially same as the algorithm described by Sivan (1996), except for the “extra size-filtering” conditions imposed at lines 4, 10, 13, 17, and 19 (refer to the line numbers marked with an asterisk in Figure 4). According to Sivan (1996), all the nodes of the quadtree, without any size-filtering criterion, are loaded into the dynamic linear list *L*. This leads to unnecessary increase in processing time, because nodes of size 1 and 2 will never be split. Hence by eliminating such nodes, we can: (a) reduce the number of nodes to be processed, and (b) avoid the expensive step of finding the edge-neighbors for a node (recollect that for checking if a node is to split, we first need to fetch all it's edge neighbors). The balanced quadtree construction time using both the algorithms is compared in Table 1. In Table 1, BQT1 is the algorithm proposed by Sivan (1996) and BQT2 is the proposed algorithm with the size-filtering steps. For illustration purpose, the algorithms were compared on: (a) raw CBV-MR image of a normal brain (refer to Figure 7(a)) and (b) preprocessed image of normal brain (refer to Figure 7(b)). The execution/construction time is reported in seconds.

```
                    Procedure BalancedQuadtree (Quadtree QT(I))

1.    Input: A quadtree QT(I) .
2.       Output: A balanced version of QT(I) .

3.       begin
4.*         Insert all the leaf nodes of QT(I) whose size is greater than two and less
            than the size of root node into a linear list, L.
5.          while  L is not empty do begin
6.             Remove a node, μ, from L.
7.             Find the edge-neighbors of μ and load them into a linear list, NL.
8.             if the size of any of the neighbor's of μ is less than
               one-half the size of μ then
9.                Make μ as non-leaf node with four children,
                  which are the sons of μ.
10.*              if size of μ is greater than 4 then
11.                  Insert the four new leaves into L.
12.               end if
13.*              Retain only those neighbors in NL whose size is greater
                  than two and less than the size of root node.
14.               while NL is not empty do begin
15.                  Remove a node, μ, from NL.
16.                  Find the edge-neighbors of μ.
17.*                 if the size of any of the neighbor's of μ is less than
                     one-half the size of μ then
18.                     Make μ as non-leaf node with four children, which
                        are the sons of μ.
19.*                    if size of μ is greater than 4 then
20.                        Insert the four new leaves into L.
21.                     end if
22.                  end if
23.               end while
24.            end if
25.         end while
26.   return QT(I) .
```

Figure 4. Procedure BalancedQuadtree which converts a linear quadtree into a balanced quadtree, *BQT*(*I*).

The results clearly demonstrate the following facts: (a) the proposed balanced quadtree algorithm works faster than Sivan's algorithm (1996), and (b) a meaningful preprocessing step prior to discretization can help in reducing the number of triangular elements in the final mesh.

Table 1: Comparison of the constructing balanced quadtree using Sivan's algorithm (BQT1) and *BalancedQuadtree* algorithm (BQT2). The execution times were obtained by running both the algorithms, under similar conditions, implemented in Java™ 2 on an Intel® PII desktop operating on SuSe® Linux 7.0 (kernel 2.2.16).

Input Image	**Size (pixels)**	**Number of Colors**	**Construction Time (secs.)**	
			BQT1	**BQT2**
Brain (refer to Figure 7(b))	270x315	2	15.771	10.025
Brain MRI (refer to Figure 7(a))	267x307	160	421.667	140.002

2.3 Triangulation

In the triangulation phase all we need to do is to triangulate the leaf nodes of *BQT*(*I*). There is more than one approach to accomplish this task. Nevertheless, here we describe the Delaunay triangulation technique, refer to the work of Sivan (1996) for an alternative approach.

In Section 1.1, we listed several practical Delaunay triangulation algorithms, which take a set of vertices as input and return a set of triangles. Hence, our foremost task in this phase would be to construct a set of vertices from the balanced quadtree, *BQT*(*I*). This is accomplished by, building a unique sorted vertex list, *SV*(*L*), comprising of the four corners of all the leaf nodes in *BQT*(*I*). Recall that one or more nodes share a corner, hence the need to construct a "unique" sorted vertex list. Once we have *SV*(*L*), we can construct Delaunay triangulation of such a list with the aide of any one of the algorithms listed in Section 1.1. For convenience, the Delaunay triangulation procedure is called as *DelaunayTriangulation*. This procedure takes a sorted list of vertices, *SV*(*L*), as input and returns a set of triangles, *DT*(*SV*(*L*)). It is noteworthy as this juncture to mention that the Delaunay triangulation of *SV*(*L*) will not be unique because we will encounter the special case discussed earlier, refer to Section 1.1. Since the four corners of a node in *BQT(I*) will always lie on the circumscribing circle of that node, this case is handled by arbitrarily completing the triangulation (refer to Figure 3 (c)). Figure 5 illustrates the complete pseudocode for generating quadtree-based triangular mesh using imageMesher algorithm.

2.4 Post-processing

In this phase, we clean up the mesh by discarding any unwanted triangles, which falls outside our domain of interest. The process of cleaning the mesh is trivial, because once we build the *BQT*(*I*) we know the value/color associated with each leaf node in the *BQT*(*I*). And as each triangle in the final mesh falls entirely within a leaf node of the *BQT*(*I*), we can use this inherent coloring scheme to retain only those triangles we desire and discard the rest from the final mesh.

2.5 Theoretical bounds

The main result of this section proves that, given a image *I*, we can compute a Delaunay triangulation, *DT*(*SV*(*L*)), of the unique four corners of the leaf nodes of *BQT*(*I*), such that the all the triangles in the triangulation will be bounded by an aspect ratio of at most 2.5 and with a minimum internal angle bounded to 26.56°. This is the most basic result of *imageMesher* algorithm. For further discussion, we define: the aspect ratio *A*(*a*, *b*, *c*) for a Δabc as: the length of the hypotenuse (longest side) divided by the length of its altitude from the hypotenuse (shortest side), and *A*(*T*) is the maximum value of the *A*(*a*, *b*, *c*) over all the triangles in a triangulation **{T}**.

Procedure imageMesher (Image *I*)

1. Input: Raster Image, *I*.
2. Intermediate Output: Quadtree, QT(*I*).
3. Intermediate Output: Balanced Quadtree, *BQT*(*I*).
4. Intermediate Output: Sorted unique vertices of leaf nodes of *BQT*(*I*), *SV*(*L*).
5. Output: Delaunay triangulation, *DT*(*SV*(*L*)).
6. **begin**
7. *QT*(*I*) = **MGV**(*I*).
8. *BQT*(*I*) = **BalancedQuadtree** (*QT*(*I*)).
9. Load sorted unique vertices of the leaf nodes of *BQT*(*I*) into *SV*(*L*).
10. *DT*(*SV*(*L*)) = **DelaunayTriangulation** (*SV*(*L*)).
11. **return** *DT*(*SV*(*L*))

Figure 5. Procedure ***imageMesher*** which generates the guaranteed quality mesh for an input raster image.

Proposition 1. The Delaunay triangulation method gives triangulation $DT(SV(L))$ for which; $A(DT(SV(L))$ is at most 2.5 and the minimum value of θ is bounded by 26.565°.

The balanced quadtree, $BQT(I)$, from imageMesher will comprise of sixteen templates nodes/blocks. As an internal quadrant can have any combination of the four neighboring quadrants, thus corresponding to $2^4 = 16$ possible configurations. Figure 6.1 shows all the possibilities. These sixteen configurations can be further reduced to only six templates after eliminating similar type of configurations, based on symmetry (refer to Figure 6.2). When we triangulate these six nodes, on close inspection, we can notice that the final triangular mesh of the *imageMesher* algorithm will be a combination of only four prototype triangular elements (refer to Figures 6.3 and 6.4). Hence, it is sufficient to analyze these four triangles for our proof. For sake of convenience, we assume that the leaf node in $BQT(I)$ has a side of length 2L.

- Case (a): The triangle, shown in Figure 6.4(a), is a right isosceles with side lengths: $2\sqrt{2}$L, 2L, and 2L; and internal angles: 90°, 45°, and 45°. Hence, the aspect ratio will be 2 (length of the shortest altitude is $\sqrt{2}$L).
- Case (b): The triangle, shown in Figure 6.4(b), is an isosceles triangle with side lengths: $\sqrt{5}$L, $\sqrt{5}L$, and 2L; and internal angles: 63.435°, 63.435°, and 53.13°. Hence, the aspect ratio will be 1.25 (length of the shortest altitude is $4L/\sqrt{5}$).
- Case (c): The triangle, shown in Figure 6.4(c), is a right angle triangle with side lengths: $\sqrt{5}$L, 2L, and L; and internal angles: 63.435°, 26.565°, and 90°. Hence, the aspect ratio will be 2.5 (length of the shortest altitude is $2L/\sqrt{5}$).
- Case (d): The triangle, shown in Figure 6.4(d), is an isosceles triangle with side lengths: $\sqrt{5}$L, $\sqrt{5}L$, and $\sqrt{2}$L; and internal angles: 71.56°, 71.56°, and 36.88°. Hence, the aspect ratio will be 5/3 (length of the shortest altitude is $3L/\sqrt{5}$).
- From the above discussion, it is follows that the minimum bounding angle, θ, is 26.565° and the bounding aspect ratio, $A(DT(SV(L)))$, is at most 2.5 .

□

The *imageMesher*[*] algorithm has been implemented to handle three most-commonly used image formats (.bmp, .gif, and .jpeg) using JDK 1.2.2 (JAVA™). It has a user-friendly interface, which enables the user to use various options (e.g. selecting the background, choosing different Delaunay triangulation algorithms, displaying the nodes of the (balanced) quadtree and mesh, printing the mesh, etc.). The application also writes the computed mesh into an ASCII file, which is compatible with *Triangle* (Shewchuk, 2001). Albeit, it is theoretically possible to mesh any image, certain programming limitations (especially, problems relating to image memory management in JAVA™) restrict the current beta version from being a robust mesh generator for images.

3. Applications

In this section, we describe two examples: one from bio-medical and other from water-resources engineering, which illustrates the application of *imageMesher* for performing numerical simulations.

[*] The JAVA™ application along with documentation is available for download on world-wide-web at: http://www.glue.umd.edu/~reddyg/imageMesher/

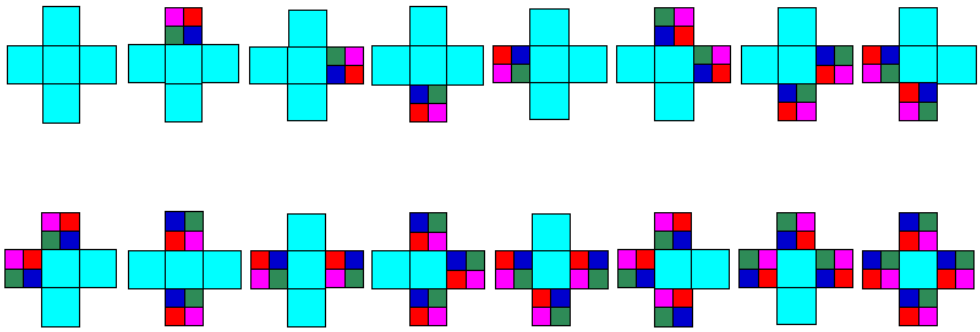


Figure 6.1.The sixteen possible quadnode configurations in a balanced quadtree decomposition.

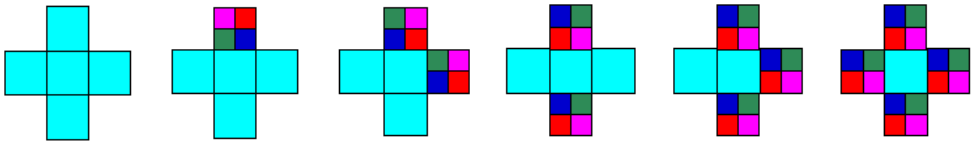


Figure 6.2.The six template quadnode configurations obtained after removing symmetrical cases from the above sixteen possible patterns.

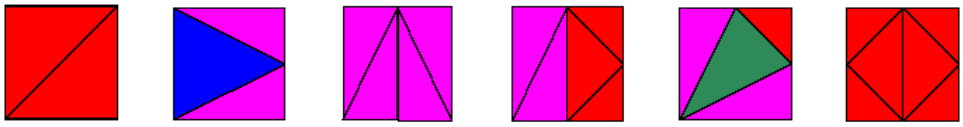


Figure 6.3.Possible Delaunay triangulations of a quadnode in a balanced quadtree decomposition.

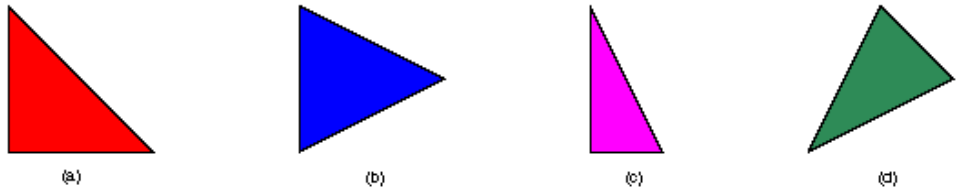


Figure 6.4.Four prototype triangles deduced from the set of Delaunay triangulation of nodes in a balanced quadtree.

The first example deals with numerical simulation of a simple mathematical model for describing the proliferation and infiltration of glioma – anaplastic astrocytoma – based in part on quantitative image analysis of histological sections of human brain, using a finite element method (FEM) formulation. We use the equations and model parameters estimated by (Tarcqui et al., 1995) in our numerical simulation for evaluating the tumor growth dynamics of a patient suffering from anaplastic astrocytoma who received two treatments of chemotherapy and neutron irradiation during the terminal year. The first chemotherapy treatment was 5 cycles of 6 drugs (UW protocol) and the second treatment comprised of *cis*-platinum dosage. The time course of the treatment along with the spatial effects of tumor growth due to the geometry of the brain and its natural barriers are also incorporated into the modeling. The investigators suggested a two cell population hypothesis for modeling glioma growth: the first type, $C_1(x, t)$, being sensitive to both the first course of chemotherapy and the second course: the second type, $C_2(x, t)$, is assumed to be resistant to the first course of chemotherapy, but possibly sensitive to the second course. The spatio-temporal evolution of the two cell populations: $C_1(x, t)$ and $C_2(x, t)$ are evaluated by the solutions of the following governing partial differential equations (*GPDE*s):

$$\left.\begin{aligned} \frac{\partial C_1}{\partial t} &= D\nabla^2 C_1 + r_1 C_1 - K_1(t)C_1 - K_2(t)C_1 \\ \frac{\partial C_2}{\partial t} &= D\nabla^2 C_2 + r_2 C_2 - K_2(t)C_2 \end{aligned}\right\} \tag{3.1}$$

where

$$K_1(t) = \begin{cases} k_1 \;\; if \;\; t_{1,i} \le t \le t_{1,i+1} & i = 1,3,5,7,9 \\ 0 & otherwise \end{cases} \tag{3.2.1}$$

$$K_2(t) = \begin{cases} k_2 \;\; if \;\; t_{2,1} + 4 \le t \le t_{2,2} + 6 \\ k_2 \;\; if \;\; t_{2,3} \le t \le t_{2,4} + 2 \\ 0 \qquad\qquad otherwise \end{cases} \tag{3.2.2}$$

D = diffusion coefficient
r_1 and r_2 = growth rates of the two types of cancer cells.
C_1 and C_2 = concentrations to two types of cancer cells.

As an approximate initial condition, the investigators assumed that at the time of starting of the series of chemotherapies, the diffusion process has already broken any previously established uniform distribution of the cells. Hence, the cells are normally distributed with a maximum cell density at the center, x_o. That is:

$$I.C \equiv \begin{cases} C_1(\bar{x},0) = a_1 \exp\left(\dfrac{-\left|\bar{x}-\bar{x}_o\right|^2}{b}\right) \\ C_2(\bar{x},0) = a_2 \exp\left(\dfrac{-\left|\bar{x}-\bar{x}_o\right|^2}{b}\right) \end{cases} \tag{3.3}$$

where the parameters a_1 and a_2 are the maximum initial density of the two types of cells and b measures the spread of the tumor cells centered at x_o. As associated boundary conditions, Neumann condition (zero flux condition) is applied at the boundaries of the brain and at ventricles. That is:

$$B.C \equiv \hat{n}.\nabla C_i(\bar{x},t) = 0\Big] \qquad i = 1,2 \tag{3.4}$$

Figures 7(a) and 7(b) show the MRI image of brain and the processed cross-section of brain extracted from the MRI image, respectively. Figures 7(c) and 7(d) show the balanced quadtree block-decomposition and the triangular mesh computed using *imageMesher*, respectively. Figures 8(i), 8(ii), and 8(iii) show the spatio-temporal growth of: type I, type II and total cancerous cell concentration (expressed in number of cells per sq. cm), respectively, after: day 1, day 113, day 230, and day 300 during the treatment in the terminal year. It should be mentioned here that all the model parameters in this simulation are constant over the entire domain, i.e., they are not spatially varying. The next example illustrates the modeling of equations that have spatially heterogeneous parameters.

The second example models two-dimensional steady-state subsurface flow with accretion for a spatially heterogeneous aquifer with a non-horizontal bottom. The GPDE describing such a system, after applying Dupuit's approximation (Bear, 1972), is given by:

$$\frac{\partial}{\partial x}\left[K(h-\eta)\frac{\partial h}{\partial x}\right] + \frac{\partial}{\partial y}\left[K(h-\eta)\frac{\partial h}{\partial y}\right] + N = 0 \tag{3.5}$$

where

$K = K(x, y)$ = spatially heterogeneous profile soil permeability
$\eta = \eta(x, y)$ = non-horizontal aquifer bottom.
$h = h(x, y)$ = phreatic surface.
N = accretion.

The boundary of the study area is assumed to act as no-flow condition (Neumann condition) while water-bodies within the study area act as constant water table elevation boundaries (Dirichlet boundary condition).

The model equation (3.5) is applied to simulate the sub-surface water movement in a small agricultural watershed located in Dorchester County, on the Maryland eastern shore, within the costal plain physiographic region. The watershed has a relatively flat topography with sandy and loamy soils. The data required for this analysis was imported from a raster-based GIS environment. Figure 9(a) shows the outline and streams of the study area. The profile soil permeability, obtained from NRCS SSURGO digital

coverage, underlying the watershed is shown in Figure 9(b). The bedrock elevation developed for this watershed is shown in Figure 9(c). For simplicity, K and η are assumed to be constant within the element while forming the elemental equations in a Galerkin's weighted-residual finite element formulation. This assumption simplifies equation (3.5) into:

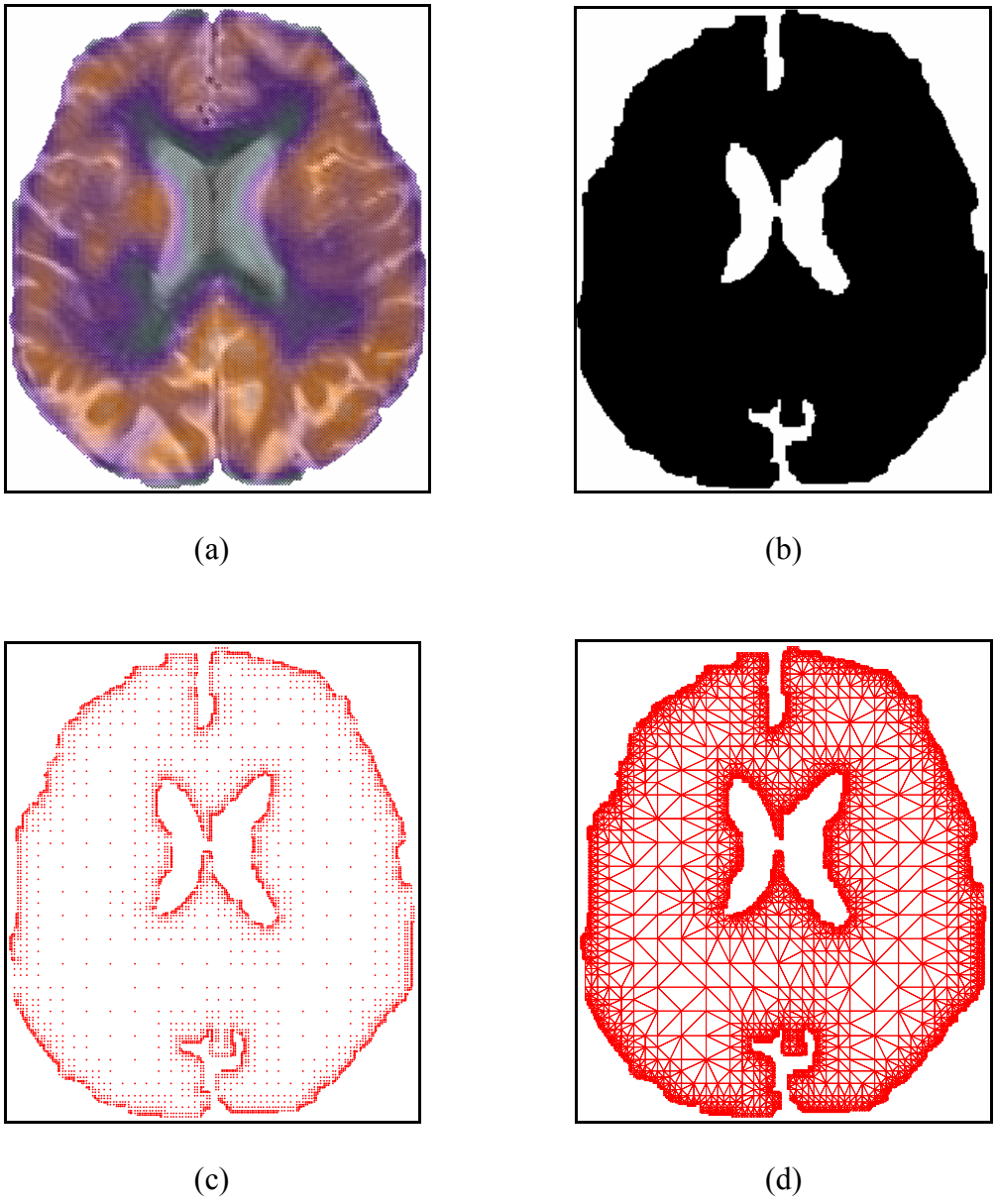


Figure 7. Illustrates the various steps usually involved in domain discretization of images: (a) shows the CBV-MR map of normal brain (Data Source: *The Whole Brain Atlas*, Harvard Medical School), (b) shows the preprocessed black-white image of brain, (c) shows the corners of the nodes from the balanced quadtree decomposition of the black-white brain image, and (d) shows the triangular mesh generated using *imageMesher* algorithm.

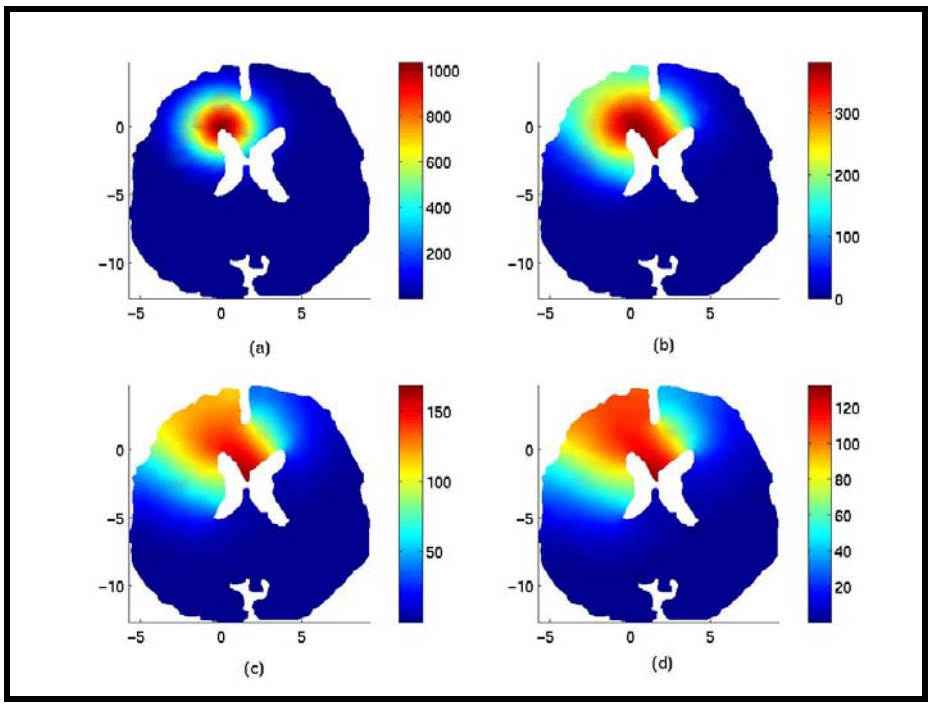


Figure 8(i). The image of the spatio-temporal evolution of type I cancerous cells, after: (a) day 1 (initial condition), (b) day 113, (c) day 230, and (d) day 300.

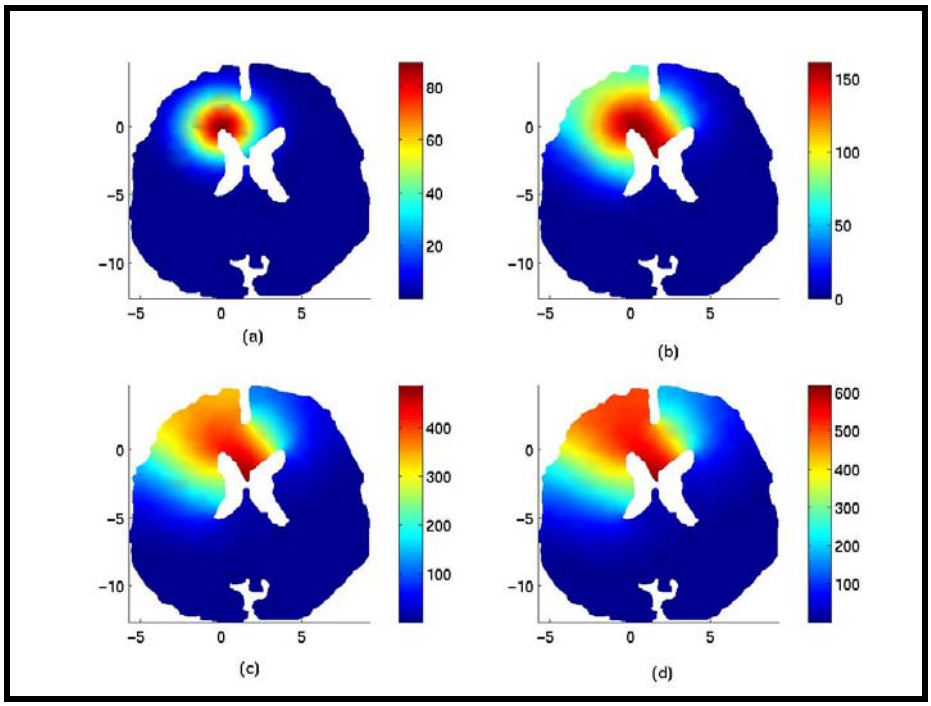


Figure 8(ii). The image of the spatio-temporal spreading of type II cancerous cells, after: (a) day 1 (initial condition), (b) day 113, (c) day 230, and (d) day 300.

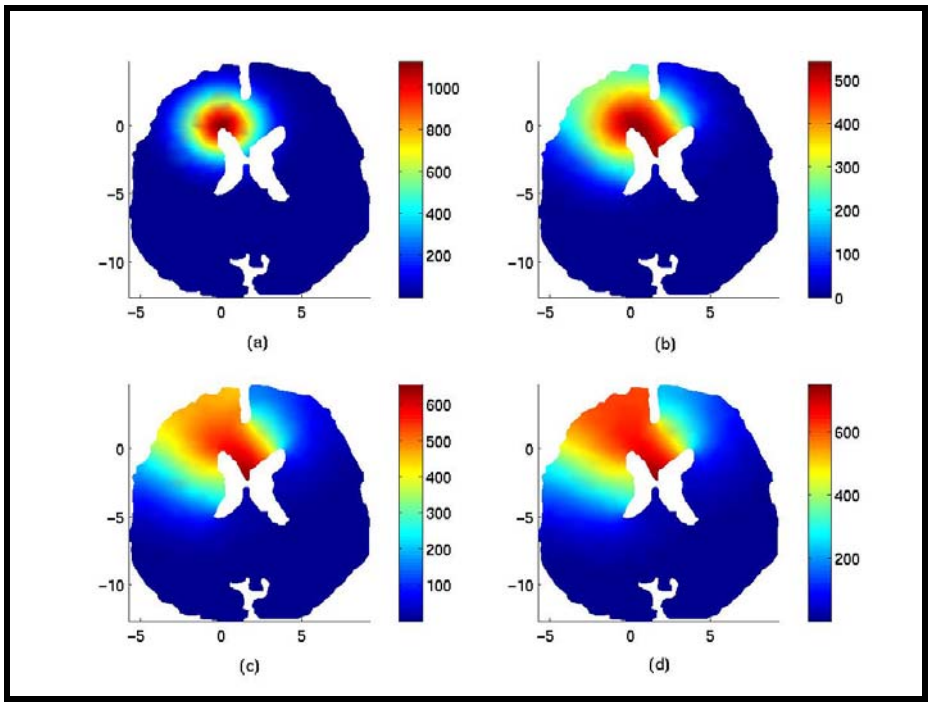


Figure 8(iii). The image of the spatio-temporal spreading of the tumor inside the brain, after: (a) day 1 (initial condition), (b) day 113, (c) day 230, and (d) day 300.

$$\frac{\partial}{\partial x}\left[(h-\eta^{(e)})\frac{\partial h}{\partial x}\right]+\frac{\partial}{\partial y}\left[(h-\eta^{(e)})\frac{\partial h}{\partial y}\right]+\frac{N^{(e)}}{K^{(e)}}=0 \qquad (3.6)$$

where the parameters with superscript *e* are assumed to constant within an element and *h* is nodal value to be estimated. Thus, to use equation (3.6) in a FEM formulation over the study area, we need to discretize the domain in such a manner that the value of *K* and η within each triangular patch remains constant. This was accomplished by taking the intersection of three images shown in Figures 9(a), 9(b), and 9(c). The result of intersection is shown in Figure 9(d).

The nodes of the balanced quadtree decomposition and triangular mesh generated using *imageMesher* are illustrated in Figures 10(a) and 10(b), respectively. Water table elevation obtained from the numerical simulation is shown in Figure 11. The elevation is expressed in meters. Table 2 summarizes the various properties of meshes used in the above two application examples.

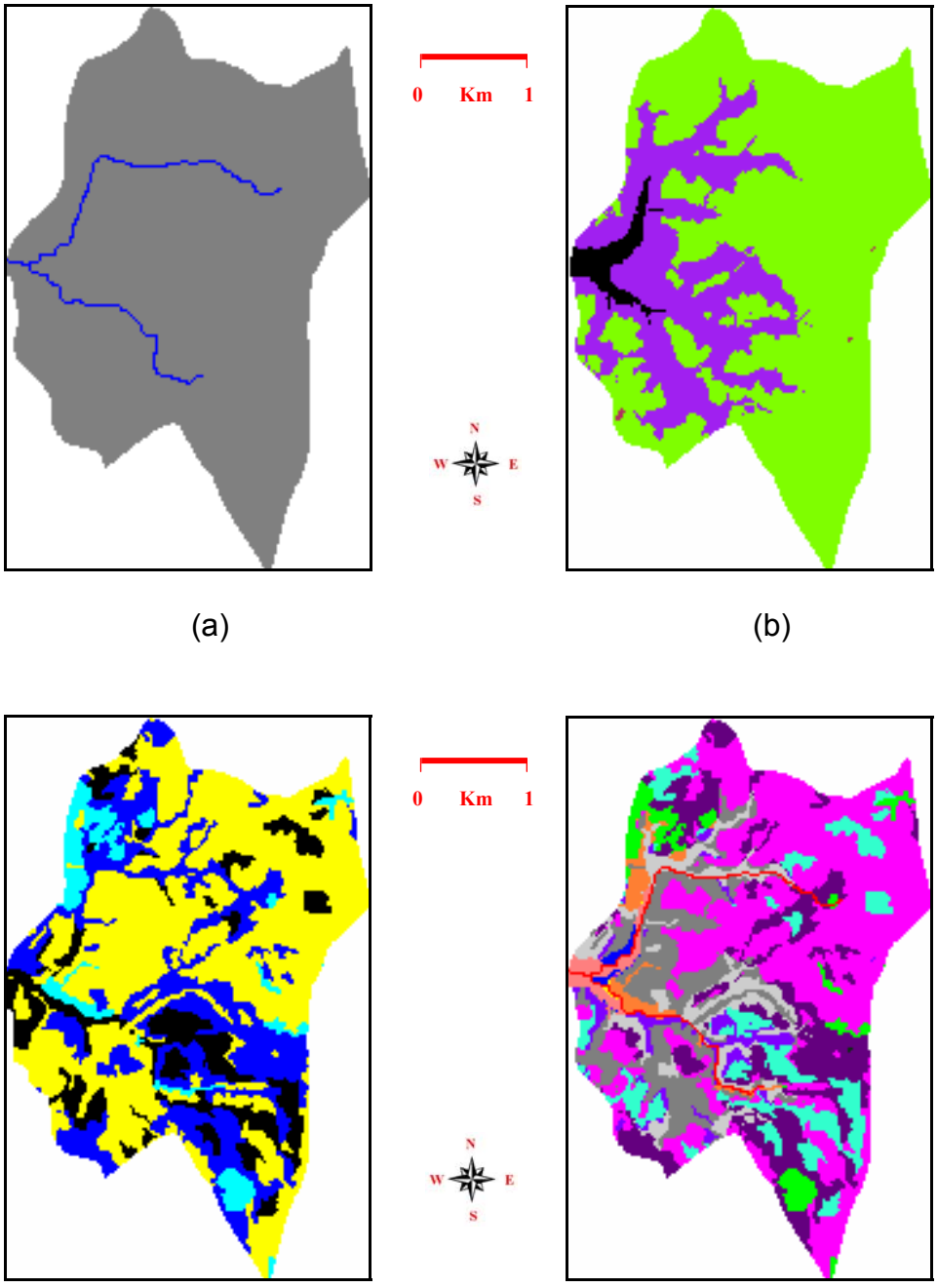
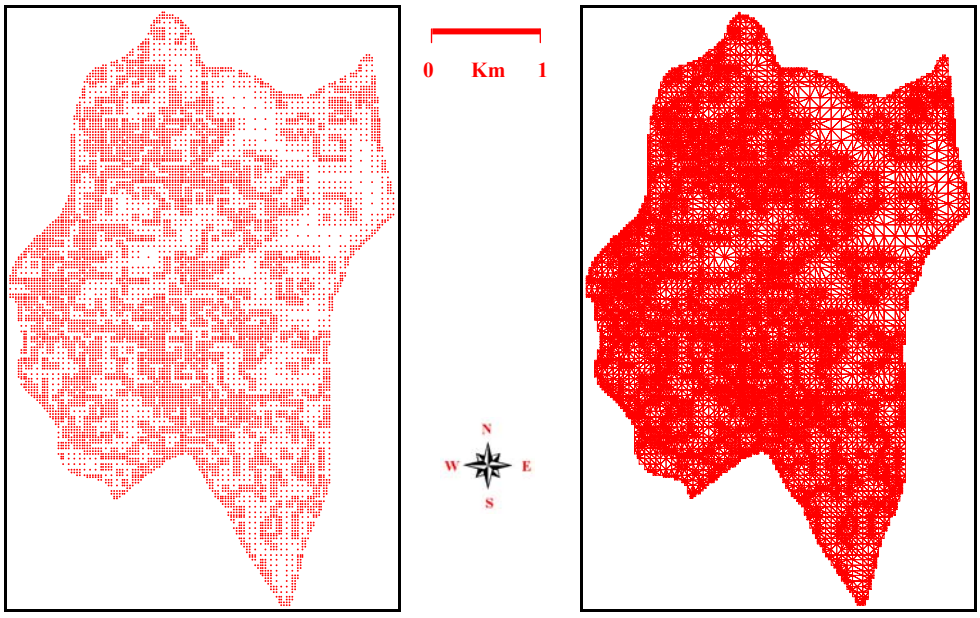


Figure 9. Images used input in the analysis of steady-state subsurface water table movement a watershed located in Dorchester County, Maryland. (a) shows the watershed boundary along with the streams flowing through it, (b) shows the classified bedrock elevation within the watershed, (c) shows the spatial variation of profile soil permeability within the watershed, and (d) shows the thematic map resulting from the intersection of streams, bedrock elevation and soil permeability.



(a) (b)

Figure 10. Shows (a) the nodes of balanced quadtree decomposition, and (b) the domain discretization of the spatially heterogeneous image/map shown in of the image shown in Figure 10(d).

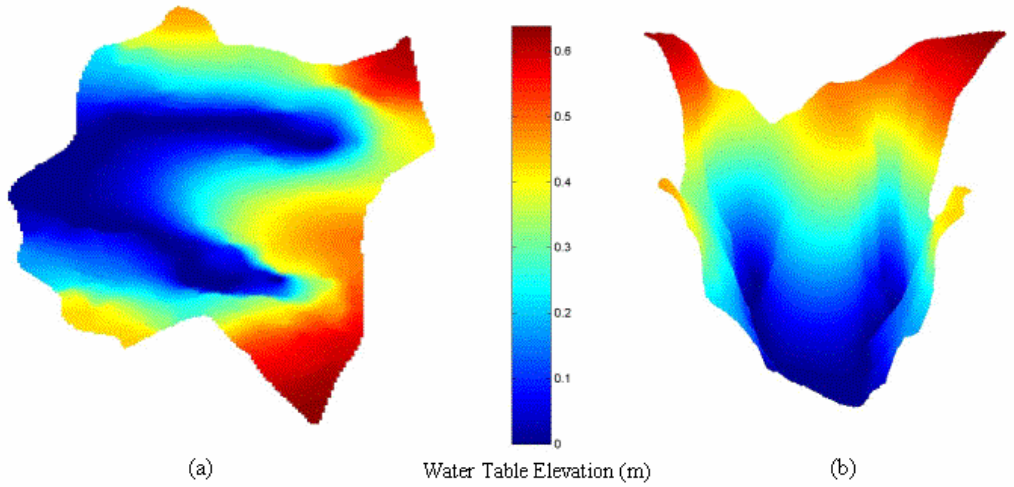


Figure 11. Steady-state water table elevation obtained from the finite element numerical simulation of Dupuit's equation for the study watershed in Dorchester County, Maryland. (a) shows the top view, and (b) shows the perspective view of water table elevation in meters.

Table 2: Summary of the meshes used in the above two application examples. The table shows the various characteristics of images, the number of nodes and elements obtained by using the imageMesher algorithm, and the total processing time in seconds.

Input Image	Size (pixels)	Number of Colors	No. of Nodes		No. of Elements		Total Processing Time. (secs)
			Unprocessed	Processed	Unprocessed	Processed	
Brain (refer to Figure 7(b))	270x315	2	7972	4910	15900	8046	17.56
Intersection Map (refer to Figure 9(d))	288x446	14	12766	11511	25426	22288	23.22

4. Conclusion

In this paper, after presenting a brief review of some elementary definitions related to quadtree-based mesh generation for PSLGs, we described a new method suitable for generating theoretically guaranteed quality mesh for spatially heterogeneous data represented in images. We proved that the triangular elements in the output mesh have: (a) a bounded aspect ratio of at most 2.5, and (b) interior angles bounded between 26.565^o and 90^o. We also described a modified algorithm for constructing a balanced quadtree and illustrated with suitable evidence that the proposed algorithm works faster than the algorithm of Sivan (1996). Finally, we discussed applications of meshes derived from images to illustrate the main features and utility of this approach.

Additional work would be needed to determine an appropriate preprocessing methodology that can fit directly into the proposed algorithm, instead of using an external tool. Secondly, as the proposed algorithm makes use of the complete balanced-quadtree structure for mesh generation, the resulting output mesh will usually contain a large number of triangular elements. For instance, the mesh used in the numerical simulation of the second example comprised of elements in the order of 10^4. For a two-dimensional analysis, this is considered very high. Hence, it would be worthwhile to investigate the mesh generation of *pruned-balanced-quadtree* based on some threshold criterion. Our approach can be extended for implementation on parallel processors based on the algorithms proposed by Bern et al. (1993) for constructing linear and balanced quadtrees on parallel processors Lastly, this work can be considered as a precursor for (re)-construction of three-dimensional geometry and mesh from a stack of registered 2-D images.

Acknowledgements

The authors would like to thank the financial assistance from the College of Agriculture and Natural Resources through the Maryland Cooperative Extension Service (MCES) and Maryland Agricultural Experimentation Station (MAES).

References

Barth, T.J. and D.C. Jespersen . 1989. "The design and application of upwind schemes on unstructured meshes". AIAA Paper No. 89-0366.

Bear, J. 1972. *Dynamics of Fluids in Porous Media*. New York, NY: American Elsevier Publishing Company, Inc.

Bern, M., D. Eppstein and J.R.Gilbert. 1990. Provably good mesh generation. In *Proc. 31st {IEEE} Symp. Foundations of Computer Science*. 31:231-241. Also available in *J. Comp. Sys. Scis.* 48:384-409.

Bern, M., D. Eppstein, and S. H. Teng. 1993. Parallel construction of quadtrees and quality triangulations. In *Proc. 3rd Workshop Algorithms and Data Structures*. LNCS 709: 188-199. New York, NY, Springer-Verlag.

Bern, M and P. Plassmann. 1999. Mesh generation. In *Handbook of Computational Geometry*, Chap. 6, ed. J.-R. Sack and J. Urrutia. Elsevier Science Publishers B.V. North-Holland, Amsterdam.

de Berg M., V.M.Kreveld, M.Overmars, and O. Schwarzkopf. 1997. *Computational Geometry, Algorithms and Applications*. New York, NY: Springer Verlag.

Bowyer, A. 1981. "Computing Dirichlet tessellations". *Comp. J.* 24(2): 162-166.

Burton, W.F., V.J.Kollias, and Y.J.Kollias. 1986. "Real-time raster to quadtree and quadtree to raster conversion algorithms with modest storage requirements". *Angew. Informatik* 4:170-174.

Chew, P.L. 1989. "Constrained Delaunay triangulations". *Algorithmica* 4(1):97-108.

Chew, P.L. 1990. Building Voronoi diagrams for convex polygons in linear expected time. Tech. Report PCS-TR90-147, Department of Mathematics and Computer Science, Dartmouth College.

Clarkson, K.L., and P.W. Shor. 1989. "Applications of random sampling in computational geometry- II". *Disc. and Comp.Geo.* 4(1):387-421.

Delaunay, B.N. 1934. “Sur la Sphere Vide”. Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Mathematicheskii i Estestvennyka Nauk 7:793-800.

Devillers, O. 1998. “Improved incremental randomized Delaunay triangulation”. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, Paper No. 106115.

Dwyer, R.A. 1987. “A faster divide-and-conquer algorithm for constructing Delaunay triangulations”. Algorithmica 2(2): 137-151.

Fang, L.J., L. Piegl. 1992. "Algorithm for Delaunay triangulation and convex hull computation using a sparse matrix". *Comp. Aided Desg.* 24(8): 425-436.

_____. 1993. “Delaunay triangulation using a uniform grid”. *IEEE Comp. Grap. and Appls.* 13(3): 36-47.

Fortune, S. 1987. “A sweepline algorithm for Voronoi diagrams”. *Algorithmica* 2(2): 153-174.

Goel, V., and Y.V. Venkatesh. 1991. “On an optimal and faster construction of linear quadtrees from raster-scanned images”. *The Computer J.* 34(6): A073-A083.

Guibas, L.J., D.E. Knuth, and M. Sharir. 1992. “Randomized incremental construction of Delaunay and Voronoi diagrams”. *Algorithmica* 7(4): 381-413.

Guibas, L.J., and J. Stolfi. 1985. “Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams”. *ACM Trans. on Grap.* 4(2): 74-123.

Herzen, V., and A. H. Barr. 1987. “Accurate triangulations of deformed, intersecting surfaces”. *Comp. Graph.* 21:103-110.

Joe, B. 1991. “Construction of three-dimensional triangulations using local transformations”. *Comp. Aided Geo. Desg.* 8:123-142.

_____. 1993. “Construction of k-dimensional Delaunay triangulations using local transformations”. *SIAM J. Scien. Comp.* 14(6): 1415-1436.

_____. 1995. “Construction of three-dimensional improved-quality triangulations using local transformations”. *SIAM J. Scien. Comp.* 16(6): 1292-1307.

Lohner, R., 1988. “Generation of three-dimensional unstructured grids by the advancing front method”. Proc. 26th AIAA Aerospace Sciences Meeting, Reno, NV.

Mavriplis, D.J. 1991. “Unstructured and adaptive mesh generation for high Reynolds number viscous flows”. ICASE, NASA Langley Research Center, Technical Report 91-25.

Mitchell, S.A. 1994. “Cardinality bounds for triangulations with bounded minimum angle”. In *Sixth Canadian Conf. on Comp.Geo.*, 326—331. Wang., Can.:CCCG.

Mitchell, S.A., and S. Vavasis. 1992. “Quality mesh generation in three dimensions”. In *Proc. 8th ACM Symp. Comp. Geom.*, 212-221.

_____. 1996. “An aspect ratio bound for triangulating a d-grid cut by a hyperplane”. In *Proc. 12th ACM Symp. Comp. Geom.*, 48-57.

_____. 2000. “*Quality mesh generation in higher dimensions*”. *SIAM J. on Comp.* 29(4): 1334-1370.

Montas, H.J., L.B. Moran, C. Peters, K. Shipman, T.H. Ifft, G.K. Felton, and A. Shirmohammadi, 2000. “GIS Evaluation of Riparian Buffer Impacts in a Maryland Watershed.” ASAE Paper No. 00-2182 *Presented at the 93rd Annual International Meeting of ASAE*, July 9 –12, Milwaukee, WI, USA. ASAE, 2950 Niles Rd., St. Joseph, MI, 49085-9659, USA.

Montas, H.J., G.V.S. Prabhakar Reddy, T. Shorabi, W. Devaneny, and A. Shirmohammadi. 2001.“Wavelet-Stochastic Analysis of Two-Dimensional Biological Resources”. ASAE Paper No. 01-3154 *Presented at the 94th Annual International Meeting of ASAE*, July 30 – Aug 1, Sacramento, CA, USA. ASAE, 2950 Niles Rd., St. Joseph, MI, 49085-9659, USA.

Moore, D.W. 1992. “Simplicial mesh generation with applications”. PhD Thesis, Cornell University, Department of Computer Science, Ithaca, NY, Cornell University Technical Report 92-1322.

Neugebauer, F., and R. Diekmann. 1996. “Improved mesh generation: Not simple but good”. In *5th Int. Meshing roundtable*, 257-270. Sandia National Laboratories.

Preparata, F.P., and M.I. Shamos. 1985. *Computational Geometry: An Introduction*. New York, NY: Springer-Verlag.

Rajan, V.T. 1994. “Optimality of the Delaunay triangulation in R^d”. *Disc. and Comp. Geo.* 12: 189-202.

Ruppert, J. 1993. “A new and simple algorithm for quality 2-dimensional mesh generation”. In *4th ACM-SIAM Symp. on Disc. Algos.*, 83--92.

_____. 1995. “A Delaunay refinement algorithm for quality 2-dimensional mesh generation”. *J. Algorithms* 18(3): 548-585.

Samet, H. 1980(a). “Region representation: Quadtrees from binary arrays”. *Comp. Grap. Image Pro.* 13(1): 88-93.

_____. 1980(b). “Region representation: Quadtrees from boundary codes”. *Comm. ACM* 23(3): 163-179.

_____. 1981. “An algorithm for converting rasters to quadtrees”. *IEEE Trans. Pattern Anal. Mach. Intell.* 3(1): 93-95.

_____. 1984(a). “Algorithms for the conversion of quadtrees to rasters”. *Comp. Visi., Grap. and Image Pro.* 26(1): 1-16.

_____. 1984(b). “The quadtree and related hierarchical data structures”. *Computing surveys* 16: 188-260.

_____. 1990(a). *The design and analysis of spatial data structures.* Reading, MA: Addison-Wesley.

_____. 1990(b). *Applications of spatial data structures: Computer Graphics, Image processing and GIS*. Reading, MA: Addison-Wesley.

Seidel, R. 1992. “Backwards analysis of randomized geometric algorithms”. Int. Computer Science Institute, University of California at Berkeley, Berkeley, California. Technical Report TR-92-014.

Shamos, M.I., and D. Hoey. 1975. “Closest-point problems”. In *16th Annual Symp. On Foundations of Computer Science (Berkeley, California)*, 151-162. IEEE Computer Society Press.

Shephard, M., and M. Georges. 1991. “Automatic three-dimensional mesh generation by the finite octree technique”. *Int. J. Numer. Meth. Eng.* 32:709-749.

Shewchuk, J.R. Triangle - A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator. http://www.cs.cmu.edu/~jrs/quake.html. Accessed 21 Jan. 2001.

_____. 1999. *Lecture Notes on Delaunay Mesh Generation*.

Sibson, R. 1973. “Locally equiangular triangulations”. *The Comp. J.* 2(3): 243-245.

Sivan, R. 1996. “Surface modeling using quadtrees”. PhD Thesis, Center for Automation, University of Maryland, College Park, MD. Technical Report CAR-TR-816.

Sivan, R. and H. Samet. 1992. Algorithms for constructing quadtree surface maps. In *Proc.of the 5th Intl. Sysp. on Spatial Data Handling*, 361-370. Charleston, SC.

Tanaka, H. T. 1995. “Accuracy-based sampling and reconstruction with adaptive meshes for parallel hierarchical triangulation”. *Int.J. of Comp. Visi. and Image Und.* 61: 335-350.

Tracqui, P., G.C. Cruywagen, D.E. Woodward, G.T. Bartoo, J.D. Murray, and E.C. Alvord, Jr. 1995. “A mathematical model of glioma growth: The effect of chemotherapy on spatio-temporal growth”. *Cell Prolif*. 28: 17-31.

Yerry, N.P., and M.S. Shephard. 1983. “A modified quadtree approach to finite element mesh generation”. *IEEE Comp. Graps and Appls.* 3(1): 39-46.