Geol. Jb.	**A 122**	45—58	5 fig.	Hannover 1992

Hierarchical Data Structures for Three-Dimensional Data

HANAN SAMET

Data structures (octree, quadtree), graphic display, three-dimensional models, morphology

A b s t r a c t : An overview, with an emphasis on recent results, is presented of the use of hierarchical data structures such as the octree and quadtree to represent three-dimensional data. They are based on the principle of recursive decomposition. The focus is on the representation of data used in solid modeling and the representation of terrain data. The emphasis is on three-dimensional regions and surfaces.

[Hierarchische Datenstrukturen für dreidimensionale Daten]

K u r z f a s s u n g : Unter besonderer Berücksichtigung aktueller Ergebnisse soll die vorliegende Arbeit einen Überblick über den Gebrauch hierarchischer Datenstrukturen (z.B. Octree und Quadtree) für die Darstellung dreidimensionaler Daten geben. Sie basieren auf dem Prinzip rekursiver Aufspaltung. Das Hauptaugenmerk liegt auf der Darstellung von Daten für dreidimensionale Modelle und der Darstellung von Geländedaten mit Betonung dreidimensionaler Regionen und Oberflächen.

Contents

1 Introduction

Hierarchical data structures are important in the domains of image processing, solid modeling, computer graphics, and geographic information systems. They are based on the principle of recursive decomposition (similar to divide and conquer methods). They are used primarily as devices

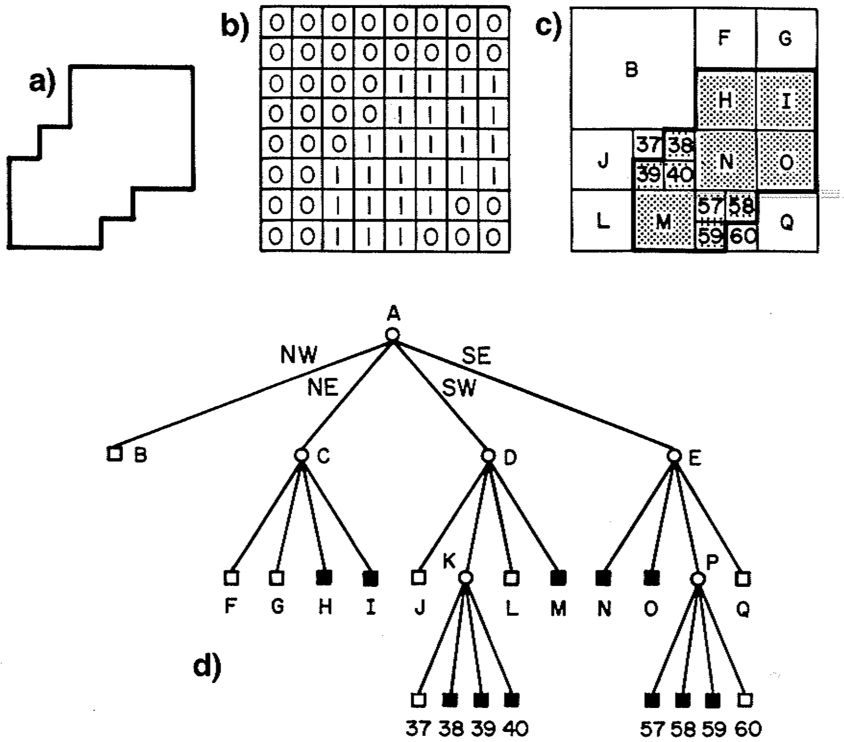
Author's address: Dr. H. SAMET, Computer Science Department, Center for Automation Research and Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland 20742, USA.

to sort data of more than one dimension and different spatial types. The term *quadtree* is often used to describe this class of data structures. For a more extensive treatment of this subject, see SAMET (1990a, b).

In this paper we review the use of hierarchical data structures to represent three-dimensional regions and surfaces. Our presentation is organized as follows: Section 2 describes the region quadtree and octree and briefly reviews the historical background of the origins of hierarchical data structures for regions. Section 3 discusses the region octree while Section 4 discusses the PM octree. Section 5 describes hierarchical surface-based object representations. Section 6 contains concluding remarks in the context of a geographic information system that makes use of these concepts.

2 Historical Background

The term *quadtree* is used to describe a class of hierarchical data structures whose common property is that they are based on the principle of recursive decomposition of space. They can be differentiated on the following bases: (1) the type of data that they are used to represent, (2) the principle guiding the decomposition process, and (3) the resolution (variable or not). Currently, they are used for points, rectangles, regions, curves, surfaces, and volumes. The decomposition may be into equal parts on each level (termed a *regular decomposition*), or it may be governed by the input. The resolution of the decomposition (i.e., the number of times that the decomposition process is applied) may be fixed beforehand or it may be governed by properties of the input data.



0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	0	0
0	0	1	1	1	0	0	0

Fig. 1: A region, its binary array, its maximal blocks, and the corresponding quadtree: (a) region; (b) binary array; (c) block decomposition of the region in (a); blocks in the region are shaded; (d) quadtree representation of the blocks in (c)

The most common quadtree representation of data is the *region quadtree*. It is based on the successive subdivision of the image array into four equal-size quadrants. If the array does not consist entirely of ones or entirely of zeros (i.e., the region does not cover the entire array), it is then subdivided into quadrants, subquadrants, etc., until blocks are obtained (possibly single pixels) that consist entirely of ones or entirely of zeros.

As an example of the region quadtree, consider the region in Figure 1a, which is represented by the $2^3 \times 2^3$ binary array in Figure 1b. Observe that the ones correspond to picture elements (termed *pixels*) that are in the region and the zeros correspond to picture elements that are outside the region. The resulting blocks for the array of Figure 1b are shown in Figure 1c. This process is represented by a tree of degree 4 (Fig. 1d). The leaf nodes correspond to those blocks for which no further subdivision is necessary. A leaf node is black or white, depending on whether its corresponding block is entirely inside or entirely outside of the represented region. All non-leaf nodes are gray.

Quadtrees can also be used to represent non-binary images. In this case, we apply the same merging criteria to each color. For example, in the case of a landuse map, we simply merge all wheat-growing regions, and likewise for corn, rice, etc. This is the approach taken by Samet et al. (1984).

Unfortunately, the term *quadtree* has taken on more than one meaning. The region quadtree, as described here, is a partition of space into a set of squares whose sides are all a power of two long (Klinger 1971). A similar partition of space into rectangular quadrants, termed a *point quadtree* (Finkel & Bentley 1974), is an adaptation of the binary search tree to two dimensions (which can be easily extended to an arbitrary number of dimensions). Its shape is dependent on the order in which the points are added to it.

Quadtree-like data structures can also be used to represent images in three dimensions and higher. The octree (Hunter 1978; Jackins & Tanimoto 1980; Meagher 1982; Reddy & Rubin 1978) data structure is the three-dimensional analog of the quadtree. It is constructed in the following manner: We start with an image in the form of a cubical volume and recursively subdivide it into eight congruent disjoint cubes (called octants) until blocks are obtained of a uniform color or a predetermined level of decomposition is reached. Figure 2a is an example of a simple three-dimensional object whose raster octree block decomposition is given in Figure 2b and whose tree representation is given in Figure 2c.

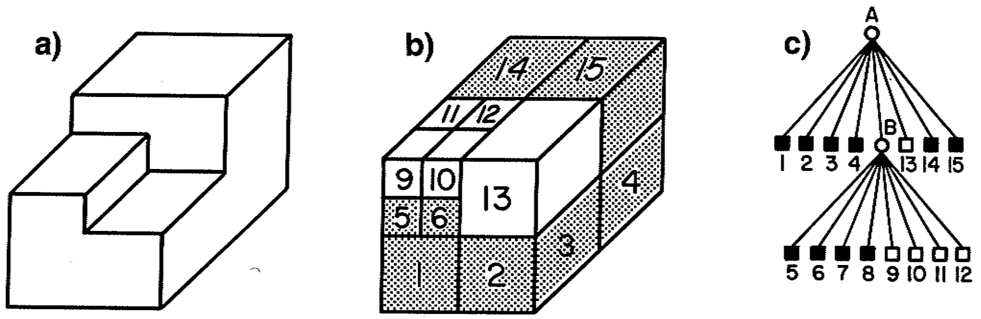


Fig. 2: (a) Example of a three-dimensional object; (b) its octree block decomposition; and (c) its tree representation

Depending on the particular implementation of the quadtree, we may not necessarily save space (e.g., in many cases a binary array representation may still be more economical than a quadtree). However, the effects of the underlying hierarchical aggregation on the execution time of the algorithms are more important. Most quadtree (and octree) algorithms are simply pre-order traversals of the tree and, thus, their execution time is generally a linear function of the number of nodes in the quadtree. A key to the analysis of the execution time of quadtree algorithms is the Quadtree Complexity Theorem (HUNTER 1978, HUNTER & STEIGLITZ 1979) which states:

For a quadtree of depth q representing an image space of $2^q \times 2^q$ pixels where these pixels represent a region whose perimeter measured in pixel-widths is p, then the number of nodes in the quadtree cannot exceed $16 \cdot q - 11 + 16 \cdot p$.

Since under all but the most pathological cases (e.g., a small square of unit width centered in a large image), the region perimeter exceeds the base 2 logarithm of the width of the image containing the region, the Quadtree Complexity Theorem means that the size of the quadtree representation of a region is linear in the perimeter of the region.

The Quadtree Complexity Theorem holds for three-dimensional data (MEAGHER 1980) where perimeter is replaced by surface area, as well as higher dimensions for which it is stated as follows:

The size of the k-dimensional quadtree of a set of k-dimensional objects is proportional to the sum of the resolution and the size of the (k—1)-dimensional interfaces between these objects.

The Quadtree Complexity Theorem also directly impacts the analysis of the execution time of algorithms. In particular, most algorithms that execute on a quadtree representation of an image instead of an array representation have an execution time that is proportional to the number of blocks in the image rather than the number of pixels. This means that the application of a quadtree algorithm to a problem in d-dimensional space executes in time proportional to the analogous array-based algorithm in the (d—1)-dimensional space of the surface of the original d-dimensional image. Therefore, quadtrees (and octrees) act like dimension-reducing devices.

3 Region Octrees

The region octree is the simplest variant of the octree data structure. It is also usually the one that requires the most space. It has the same drawback as the region quadtree in the sense that it is an approximation and, thus, it is not so suitable for some applications. Constructing a region octree from a three-dimensional array representation of an image is quite costly because of the sheer amount of data that must be examined. In particular, the large number of primitive elements that must be inspected means that the conventional raster-scanning approach used to build quadtrees spends much time detecting the mergibility of nodes.

The easiest way to speed up the region octree construction process is to reduce the amount of data that needs to be processed. FRANKLIN & AKMAN (1985) show how to build a region octree from a set of rectangular parallelepipeds approximating the object. This data can be acquired, for example, by casting parallel rays along the z-axis and perpendicular to the x,y-plane.

In many applications, an even more fundamental problem than building the octree is acquiring the initial boundary data to form the boundary of the object being represented. One approach is to use a three-dimensional pointing device to create a collection of samples from the surface of the object. After the point data is collected, it is then necessary to interpolate a reasonable surface to join it.

Interpolation can be achieved by triangulation. A surface triangulation in three-dimensional space is a connected set of disjoint triangles that forms a surface with vertices that are points in the original data set. There are many triangulation methods currently in use. POSDAMER (1982) suggests use of the ordering imposed by an octree on a set of points (e.g., by bit interleaving (SAMET 1990b) as the basis for determining the points that should be connected to form the triangles.

POSDAMER's algorithm uses an octree for which the leaf criterion is that no leaf can contain more than three points. The initial set of triangles is formed by connecting the points in the leaf nodes that contain exactly three points. Whenever a leaf node contains exactly two points, these points are connected to form a line segment that is associated with the leaf node. This is the starting point for a bottom-up triangulation of the points. It merges disjoint triangulations to form larger triangulations.

The isolated points (i.e., leaf nodes that contain just one point) and isolated line segments are treated as degenerate triangulations. The triangulation associated with a gray node is the result of merging the triangulations associated with each of its sons. By *merging* or *joining* two triangulations, we mean that a sufficient number of line segments is drawn between vertices of the two triangulations so that we get a new triangulation containing the original two triangulations as subtriangulations.

When merging the triangulations of the eight sibling octants, a number of heuristics can be used to guide the choice of which triangulations are joined first. The order in which we choose the pair of triangulations to be joined is determined, in part, by the following factors: First, it is preferred to merge triangulations in siblings whose corresponding octree blocks have a common face. If this is impossible, then triangulations in nodes that have a common edge are merged. Again, if this is not feasible, then triangulations in nodes that have a common vertex are merged. For each preference, the triangulations that are closest, according to some distance measure, are merged first.

There are many other methods of building an octree representation of an object. The simplest is to take quadtrees of cross-sectional images of the object and merge them in sequence. This technique is used in medical applications in which the cross sections are obtained by computed tomography methods (YAU & SRIHARI 1983). YAU & SRIHARI (1983) discuss this technique in its full generality by showing how to construct a k-dimensional octree-like representation from multiple (k—1)-dimensional cross-sectional images. YAU & SRIHARI's algorithm proceeds by processing the cross sections in sequence. Each pair of consecutive cross sections is merged into a single cross section. This pairwise merging process is applied recursively until there is one cross section left for the entire image.

In other applications, the volume of the available data is not as large. Often, a small number of two-dimensional images is used to reconstruct an octree representation of a three-dimensional object or a scene of three-dimensional objects. In this case, projection images (termed *silhouettes*) are taken from different viewpoints. These silhouettes are subsequently swept along the viewing direction, thereby creating a bounding volume, represented by an octree, that serves as an approximation of the object. The octrees of the bounding volumes, corresponding to views from different directions, are intersected to yield successively finer approximations of the object. In the rest of this section, we elaborate further on methods based on silhouettes.

MARTIN & AGGARWAL (1983) use this method with volume segments that are parallelepipeds stored in a structure that is not an octree. CHIEN & AGGARWAL (1984) show how to use this method to construct an octree from the quadtrees of the three orthogonal views. HONG & SHNEIER (1985) point out that the task of intersecting the octree and the bounding volume can be made more efficient by first projecting the octree onto the image plane of the silhouette, and then performing the intersection in the image plane. In contrast, NOBORIO et al. (1988) perform the intersection check directly in the three-dimensional space rather than preceding it by a projection. In the rest of this section, we assume that the silhouettes result from parallel views, although perspective views have also been used (e.g., SRIVASTAVA & AHUJA 1987).

Generally, three orthogonal views often are insufficient for an accurate approximation of the object. Thus, more views are needed. CHIEN & AGGARWAL (1986) overcome this problem by constructing what they term a *generalized octree* from three arbitrary views having the requirement that they are not coplanar. The generalized octree differs from the conventional region octree in that each node represents a parallelepiped with faces parallel to the viewing planes.

The approximation is refined by intersecting the projection of each object node P in the generalized octree with the image plane of the additional view. P is relabeled as a non-object node or a gray node unless its projection lies entirely within the object region in the additional view (see also HONG & SHNEIER 1985).

A problem with using additional views from arbitrary viewpoints is that intersection operations must be explicitly performed to determine the relationship between the projections of the octants in the octree space and the silhouette of the new view. In the general case, the silhouette can be approximated by a polygon. The intersection of the polygonal projection of an octant with the polygon approximation of a silhouette is a special case of the polygon clipping problem.

CHIEN & AGGARWAL (1984) and AHUJA & VEENSTRA (1989) point out that sweeping the silhouette image of an orthographic parallel projection and restricting the views enable the exploitation of a regular relation between octants in the octree space and quadrants in the image space. This means that the intersection operation can be replaced by a table-lookup operation. The key idea is to represent the image array by a quadtree and to make use of mappings between the quadrants and the octants so that the octree can be constructed directly from the silhouettes of the digitized image. We thereby avoid the need to explicitly perform the sweep operation.

The image array corresponding to the silhouette is processed as if we were constructing its quadtree. CHIEN & AGGARWAL (1984) use three face views while AHUJA & VEENSTRA (1989) use 13 views. The 10 additional views correspond to 6 edge views and 4 vertex views. Face views are taken with the line of sight perpendicular to a different face of the octree space; the three faces must be mutually orthogonal. Edge views are taken with the line of sight passing through the center of an edge and the center of the octree space. Vertex views are taken with the line of sight passing through a vertex and the center of the octree space. The vertex views are also known as isometric projections.

In some situations, even 13 views are inadequate to obtain a sufficiently accurate approximation of the object, particularly when the object has a number of concave regions. Here it is best to use a ranging device to obtain range data. The range data can be viewed as partitioning the scene into three parts: the visible surface of the scene, the empty space in front of this surface, and the unknown space behind the surface. CONNOLLY (1984) constructs an octree representation of the scene that corresponds to the series of such range images. This octree represents a piecewise linear approximation of the surfaces of the scene. A quadtree is used as an intermediate representation of a piecewise linear surface approximating the data comprising a single range image prior to its incorporation into the octree. CONNOLLY (1985) derives an octree-based heuristic for selecting the positions from which to take subsequent range images. However, the issue of determining the next «best» view is still an open problem.

A drawback of CONNOLLY's use of the quadtree as an intermediate representation is that the quadtree must be transformed into the octree coordinate system when the coordinate axes of the quadtree are not aligned with those of the octree. This is a relatively complex process from a computational standpoint. CHIEN et al. (1988) also represent the views by quadtrees. However, they point out that CONNOLLY's approach can be simplified by exercising control over the configuration of the range sensor. In particular, much of its complexity can be reduced (and avoided) by assuming that the ranging device is aligned with the cube that corresponds to the scene. This enables them to take advantage of the interrelationship between the quadtree and octree structures. They make use of six views to yield six range images — one for each of three pairs of orthogonal viewing directions.

CHIEN et al. (1988) generate a quadtree for each range image using a decomposition criterion so that each block has a constant range value (i.e., each block contains a square surface patch parallel to the image plane). They assume that the observed object occupies the space extending from the visible surfaces to the rear boundary of the scene cube (with respect to the ranging device being at the front for the particular view in question). Based on this assumption, they use a segment tree (a one-dimensional region quadtree) to represent the subpart of the object that is behind the visible surface patch associated with the block. Thus, the object corresponding to each view is a quadtree in which each leaf node is, in turn, a segment tree. Whereas the quadtree partitions the two-dimensional image plane into blocks of constant range value, the segment tree decomposes the remaining dimension (i.e., depth) into object and non-object regions.

The actual octree corresponding to each view (termed a *range octree*) is obtained by recursively merging the segment trees of the quadtree nodes. The rationale for using the combination of quadtrees and segment trees instead of the octree is to reduce the intermediate space requirements. Memory and time can be saved by not merging the six range octrees directly. Instead, once a pair of range octrees corresponding to opposite views (e.g., front and rear) are obtained, they are merged. The final step merges these three range octrees to yield the desired octree.

4 PM Octrees

One of the deficiencies of the region octree is that if the faces of the object(s) represented by it are not rectilinear, then the representation is inexact (it is an approximation). The only exception is if the faces are mutually orthogonal in which case a suitable rotation operation can be applied to yield rectilinear faces. In many applications this is not a problem. However, in solid modeling it is preferable to have an exact representation. When the object has planar faces, an extension of the PM quadtree, a representation for polygonal maps (SAMET & WEBBER 1985), can be used and is the focus of this section.

In the approach we describe, the resulting decomposition insures that each octree leaf node corresponds to a single vertex, a single edge, or a single face. The only exceptions are that a leaf node may contain more than one edge if all the edges are incident at the same vertex. Similarly, a leaf node may contain more than one face if all the faces are incident at the same vertex or edge. The result is termed a *PM octree*. The above subdivision criteria can be stated more formally as follows:

(1) At most, one vertex can lie in a region represented by an octree leaf node.
(2) If an octree leaf node's region contains a vertex, then it can contain no edge or face that is not incident at that vertex.
(3) An octree leaf node's region that contains no vertices can contain at most one edge.
(4) An octree leaf node's region that contains no vertices and contains one edge can contain no face that is not incident at that edge.
(5) An octree leaf node's region that contains no edges can contain at most one face.
(6) Each region's octree leaf node is maximal.

Implementation of the PM octree involves leaf nodes of type vertex, edge, and face. For our purposes, it is permissible to have more than two faces meet at a common edge. However, such a situation can not arise when modeling solids that are bounded by compact, orientable two-manifold surfaces (i.e., only two faces may meet at an edge and the surface is two-sided). Nevertheless, it is plausible when three-dimensional objects are represented by their surfaces.

The above PM octree formulation was reported almost simultaneously by three research groups who each gave it a different name (AYALA et al. 1985, CARLBOM et al. 1985; FUJIMURA & KUNII 1985). In fact, it can be traced even further back (HUNTER 1981; QUINLIN & WOODWARK 1982; TAMMINEN 1981, 1982; VANDERSCHEL 1984). The most extensive treatment of this data structure is to be found in the Ph.D. dissertation of NAVAZO (1986a). This work contains a detailed analysis of the storage requirements of the representation. It also includes algorithms for Boolean operations involving it, and conversion between it and a boundary model.

PM octree techniques have also been extended to handle curvilinear surfaces. Primitives including cylinders and spheres have been used in conjunction with a decomposition rule limiting the number of distinct primitives that can be associated with a leaf node (FUJIMOTO et al. 1986; WYVILL & KUNII 1985). Another approach (NAVAZO et al. 1986b) extends the concepts of face, edge, and vertex nodes to handle faces represented by biquadratic patches. The use of biquadratic patches enables a better fit with fewer primitives than can be obtained with planar faces, thereby reducing the size of the octree. The difficulty in organizing curved surface patches by using octrees lies in devising efficient methods of calculating the intersection between a patch and an octree node. Observe that in this approach we are organizing a collection of patches in the image space. This is in contrast to decomposing a single patch in the parametric space by use of quadtree techniques (e.g., CATMULL 1975).

5 Surface-Based Object Representations

Often, three-dimensional objects can be represented in terms of their surfaces. For some applications the primary interest is in the representation of surfaces as 2.5-dimensional images — i.e., for each pair (x,y), there corresponds a unique value of z. In applications in solid modeling the requirement that the value of z be unique is relaxed. In this section we give a brief overview of hierarchical representations of 2.5-dimensional images in the context of processing topographic data. The problem usually arises as one of reconstructing a surface in a digital environment. It is usually formulated as the interpolation of a function of two variables (say x and y) with values given at points that are either arbitrarily located or drawn from a uniformly-spaced grid. For an in-depth overview of this field, see the recent survey of DE FLORIANI (1987).

Regardless of how the surface is sampled, the representations should adapt to the changes in the terrain. The most common way of representing topographic data is to record it in a fixed rectangular or triangular grid (known as a gridded digital terrain model). An alternative method, which is more compact, is capable of capturing point and line features (e.g., peaks, pits, passes, ridges, and valleys) in the surface by approximating the surface by a network of planar non-overlapping triangles. The result is known as a Triangular Irregular Network (TIN) (PEUCKER & CHRISMAN 1975). Unfortunately, an arbitrary triangulation is usually unsatisfactory for the purpose of interpolation, due to the high likelihood that the triangles are thin and elongated (i.e., the triangles should be as equiangular as possible).

When the amount of data is large, the triangular network approach becomes unwieldy in terms of storage requirements. In this case, there are two possible solutions. The first is a pyramid-like approach that represents the surface at different predefined levels of precision. The second approach, and the one we focus on, represents different parts of the surface at different levels of resolution. Representations based on such an approach are usually characterized as being hierarchical. The hierarchical methods that are commonly used are based on either triangulations or rectangular decompositions.

Hierarchical triangular decomposition methods are differentiated on the basis of whether the decomposition is into three (t e r n a r y) or four (q u a t e r n a r y) parts. Ternary decompositions are formed by taking an internal point of one of the triangles, say T, and joining it to the vertices of T (e.g., Fig. 3). Quaternary decompositions are formed by joining three points, each on a different side of a given triangle (e.g., Fig. 4). Hierarchical triangulations are represented by trees where the root corresponds to the initial enclosing rectangle. For a ternary decomposition, each triangle is adjacent to at most one triangle on each side. In contrast, for a quaternary decomposition, each triangle may be adjacent to a number of triangles along a side.

Hierarchical triangulations result in the approximation of a surface S by planar triangular patches whose vertices are a subset of the data points that define S. For each such patch, an approximation error is computed that is usually the maximum error of the data points with projections on the x,y-plane overlapping the projection of the patch on the x,y plane. If the approximation error exceeds a predefined tolerance, then the patch is subdivided further. The resulting surface depends on the nature of the decomposition.

In the case of a ternary decomposition, the surface described by the triangulation is usually continuous at every level. However, the triangles are often thin and elongated, since the point at which the triangle is decomposed is internal to the triangle. Thus, equiangularity is not satisfied (but see the prism tree (FAUGERAS et al. 1984; PONCE & FAUGERAS 1987)). The ternary decomposition is usually used when the surface is defined at points that are randomly located. DE FLORIANI et al. (1984) discuss its use for surface interpolation as well as serving as a data compression mechanism.

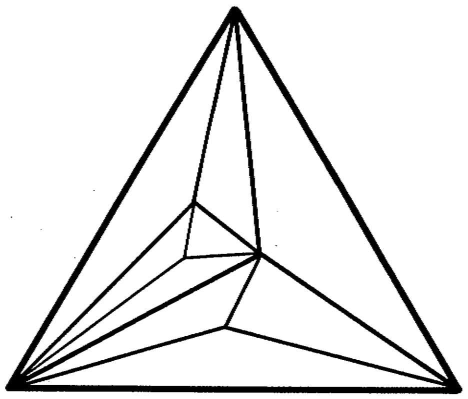


Fig. 3: Example of ternary decomposition.

In the case of a quaternary decomposition, each triangle can be adjacent to a number of triangles on each of its sides. Thus, the interpolating surface defined on it is generally not continuous unless all of the triangles are uniformly split — i.e., the resulting tree is a complete quadtree. If the initial approximating triangle is equilateral and the triangles are always subdivided by connecting their midpoints, then equiangularity holds and the interpolation is ideal. The quaternary decomposition is especially attractive when the data points are drawn from a uniformly spaced grid. The quaternary decomposition is used as as surface representation by GOMEZ & GUZMAN (1979) and BARRERA & VAZQUEZ (1984). The approach of BARRERA & VAZQUEZ uses regular decomposition while GOMEZ & GUZMAN's decomposition is data driven (like a point quadtree).

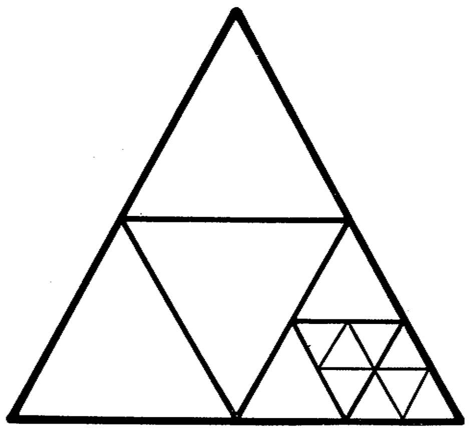


Fig. 4: Example of quaternary decomposition.

Hierarchical rectangular decompositions are similar to hierarchical triangulations that are based on a quaternary decomposition. They are used when the data points are the vertices of a rectangular grid. In this case, a rectangle is split by choosing an internal point and joining it to its projections on the four sides of the rectangle. When the data is uniformly spaced, the result is analogous to a region quadtree.

The main drawback of using a rectangular decomposition is the absence of continuity between adjacent patches of unequal width (termed the alignment problem). As an example of the *alignment problem*, see Figure 5a, which is the result of using a quadtree-like decomposition rule in the parameter space representation of a surface patch. Notice the presence of cracks along the boundary of the NE quadrant. BARRERA & HINOJOSA (1987) overcome this problem by using the interpolated point instead of the true point, while VON HERZEN & BARR (1987) triangulate the squares.

There are several ways to triangulate a square. It can be split into two, four, or eight triangles, depending on how many lines are drawn through its midpoint (one, two, or four, respectively). The simplest method is to split it into two triangles. Unfortunately, the cracks still remain. VON HERZEN & BARR (1987) avoid the cracks by converting the rectangular decomposition into a restricted quadtree. At this point, a square can be split into either four or eight triangles.

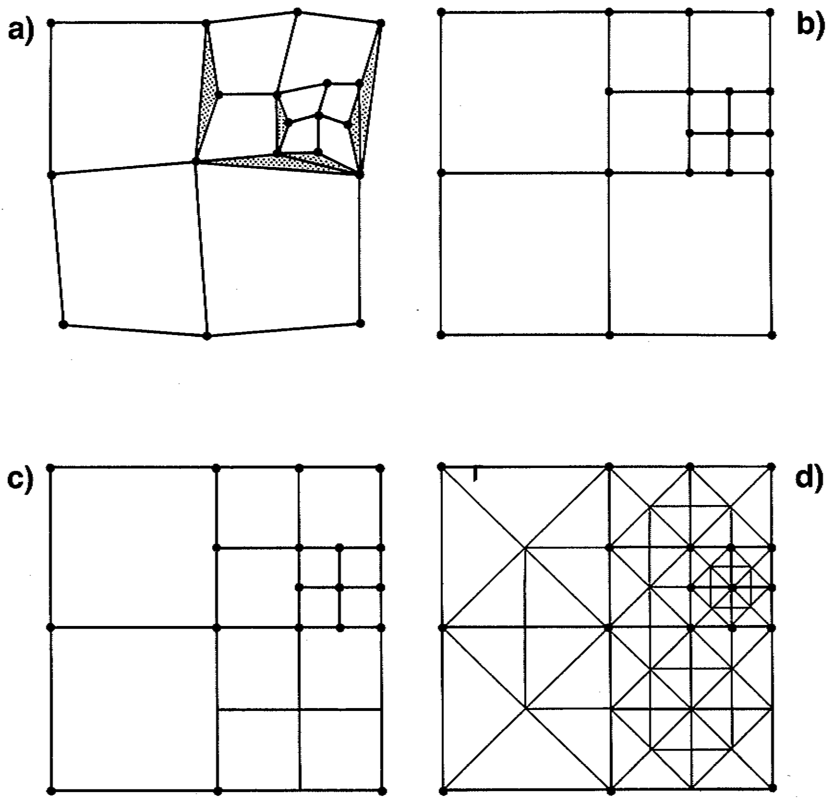


Fig. 5: (a) The three-dimensional view of the resulting subdivision of a surface using a quadtree-like decomposition rule in parameter space (some of the cracks) are shown shaded); (b) the quadtree of (a) in parameter space; (c) the restricted quadtree corresponding to (b); (d) the triangulation of (c)

A restricted quadtree is one where all four adjacent blocks (i.e., nodes) are either of equal size or of ratio 2:1. Given an arbitrary quadtree decomposition, the restricted quadtree is formed by repeatedly subdividing the larger nodes until the 2:1 ratio holds. It results in a quadtree-like decomposition (Fig. 5c), as opposed to a more traditional representation (Fig. 5b). Note that the SE quadrant of Figure 5b had to be decomposed once. This method of subdivision is also used in finite element analysis as part of a technique called h - r e f i n e m e n t by KELA et al. (1986) to adaptively refine a mesh that has already been analyzed, as well as to achieve element compatibility.

VON HERZEN & BARR (1987) overcome the alignment problem by triangulating the quadtree leaf nodes of Figure 5c in the manner shown in Figure 5d. The rule is that every block is decomposed into eight triangles, or two triangles per edge, unless the edge is shared by a larger block. In that case, only one triangle is formed. Observe that there are no cracks. Cracks can also be avoided by varying the basic decomposition rule so that each block is decomposed into four triangles, or one triangle per edge, unless the edge borders a smaller square. In that case, two triangles are formed along the edge. VON HERZEN & BARR prefer the decomposition into eight triangles because it avoids problems when displaying (i.e., shading) the resulting object.

6 Concluding Remarks

The use of hierarchical data structures enables focusing computational resources on the interesting subsets of data. Thus, there is no need to expend work where the payoff is small. Although many of the operations for which they are used can often be performed equally as efficiently, or more so, with other data structures, hierarchical data structures are attractive because of their conceptual clarity and ease of implementation.

When the hierarchical data structures are based on the principle of regular decomposition, we have the added benefit of a spatial index. All features, whether they are regions, points, rectangles, lines, surfaces, or volumes, etc., can be represented by maps which are in registration. In fact, such a system, known as QUILT (SHAFFER et al. 1990), has been built for representing geographic information. It has recently been extended to represent three-dimensional objects and surfaces. It uses the region octree representation. In addition, a three-dimension graphics package has been implemented that utilizes perspective projection. Objects can be translated and rotated. Surfaces are represented as an MX octree (HUNTER 1978; SAMET 1990a) where black nodes correspond to the surface boundary while all remaining nodes (both inside and outside of the surface) are white.

The disadvantage of quadtree methods is that they are shift sensitive in the sense that their space requirements are dependent on the position of the origin. However, for complicated images the optimal positioning of the origin will usually lead to little improvement in the space requirements. The process of obtaining this optimal positioning is computationally expensive and is usually not worth the effort (LI et al. 1982).

The fact that we are working in a digitized space may also lead to problems. For example, the rotation operation is not generally invertible. In particular, a rotated square usually cannot be represented accurately by a collection of rectilinear squares. However, when we rotate by 90°, then the rotation is invertible. This problem arises whenever one uses a digitized representation. Thus, it is also common to the array representation.

Acknowledgments

This work was supported by the National Science Foundation under Grant IRI-8802457. I would like to thank WALID AREF for extending the QUILT system to include three-dimensional and surface data.

7 References

AHUJA, N. & VEENSTRA, J. (1989): Generating Octrees from Object Silhouettes in Orthographic Views. — IEEE Transactions on Pattern Analysis and Machine Intelligence, **11** (2): 137—149; New York.

AYALA, D. & BRUNET, P. & JUAN, R. & NAVAZO, I. (1985): Object Representation by Means of Nonminimal Division Quadtrees and Octrees. — ACM Transactions on Graphics, **4** (1): 41—59; New York.

BARRERA, R. & HINOJOSA, A. (1987): Compression Methods for Terrain Relief, CINEVESTAV — IPN. — Eng. Projects Sect., Dept. of Electrical Eng., Polytechnic Univ. Mexico; Mexico City.

— & VAZQUEZ, A.M. (1984): A Hierarchical Method for Representing Terrain Relief. — Proc. of the Pecora 9 Symp. on Spatial Inform. Technologies for Remote Sensing Today and Tomorrow: 87—92; Sioux Falls, SD.

CARLBOM, I. & CHAKRAVARTY, I. & VANDERSCHEL, D. (1985): A Hierarchical Data Structure for Representing the Spatial Decomposition of 3-D Objects. — IEEE Computer Graphics and Applications, **5** (4): 24—31; New York.

CATMULL, E. (1975): Computer Display of Curved Surfaces. — Proc. of the Conf. on Computer Graphics, Pattern Recognition, and Data Structure: 11—17; Los Angeles.

CHIEN, C.H. & AGGARWAL, J.K. (1984): A Volume/Surface Octree Representation. — Proc. of the 7th Internat. Conf. on Pattern Recognition: 817—820; Montreal.

CHIEN, C.H. & AGGARWAL, J.K. (1986): Identification of 3-D Objects from Multiple Silhouettes using Quadtrees/Octrees. — Computer Vision, Graphics, and Image Processing, **36** (2/3): 256—273; New York.

— & SIM, Y.B. & AGGARWAL, J.K. (1988): Generation of Volume/Surface Octree from Range Data. — Proc. of Computer Vision and Pattern Recognition 1988: 254—260; Ann Arbor, Mich.

CONNOLLY, C.I. (1984): Cumulative Generation of Octree Models from Range Data. — Proc. Internat. Conf. on Robotics: 25—32; Atlanta.

— (1985): The Determination of Next Best Views. — Proc. Internat. Conf. on Robotics: 432—435; St. Louis, Mo.

DE FLORIANI, L. (1987): Surface Representations Based on Triangular Grids. — Visual Computer, **3** (1): 27—50; Berlin.

— & FALCIDIENO, B. & NAGY, G. & PIENOVI, C. (1984): A Hierarchical Structure for Surface Approximation. — Computers & Graphics, **8** (2): 183—193; New York.

FAUGERAS, O.D. & HEBERT, M. & MUSSI, P. & BOISSONNAT, J.D. (1984): Polyhedral Approximation of 3-D Objects without Holes. — Computer Vision, Graphics, and Image Processing, **25**(2): 169—183; New York.

FINKEL, R.A. & BENTLEY, J.L. (1974): Quad Trees: A Data Structure for Retrieval on Composite Keys. — Acta Informatica, **4** (1): 1—9; Berlin, Heidelberg (Springer).

FRANKLIN, W.R. & AKMAN, V. (1985): Building an Octree from a Set of Parallelepipeds. — IEEE Computer Graphics and Applications, **5** (10): 58—64; New York.

FUJIMOTO, A., TANAKA, T. & IWATA, K. (1986): ARTS: Accelerated Ray-tracing System. — IEEE Computer Graphics and Applications, **6** (4): 16—26; New York.

FUJIMURA, K. & KUNII, T.L. (1985): A Hierarchical Space Indexing Method. — Proc. of Computer Graphics '85, **T1-4**: 1—14; Tokyo.

GOMEZ, D. & GUZMAN, A. (1979): Digital Model for Three-dimensional Surface Representation. — Geo-Processing, **1**: 53—70; Amsterdam.

HONG, T.H. & SHNEIER, M. (1985): Describing a Robot's Workspace Using a Sequence of Views from a Moving Camera. — IEEE Transactions on Pattern Analysis and Machine Intelligence, **7**(6): 721—726; New York.

HUNTER, G.M. (1978): Efficient Computation and Data Structures for Graphics. — Ph.D. dissertation, Dept. of Electrical Eng. and Computer Science, Princeton Univ.; Princeton, NJ.

— (1981): Geometrees for Interactive Visualization of Geology: an Evaluation. — System Sci. Dept., Schlumberger-Doll Research; Ridgefield, CT.

— & STEIGLITZ, K. (1979): Operations on Images Using Quad Trees. — IEEE Transactions on Pattern Analysis and Machine Intelligence, **1** (2): 145—153; New York.

JACKINS, C.L. & TANIMOTO, S.L. (1980): Octrees and their Use in Representing Three-dimensional Objects. — Computer Graphics and Image Processing, **14** (3): 249—270; New York.

KELA, A., PERUCCHIO, R. & VOELCKER, H. (1986): Toward Automatic Finite Element Analysis. — Computers in Mech. Eng., **5** (1): 57—71; New York.

KLINGER, A. (1971): Patterns and Search Statistics. — In: RUSTAGI, J.S. [Ed.]: Optimizing Methods in Statistics: 303—337; New York (Academic Press).

LI, M. & GROSKY, W.I. & JAIN, R. (1982): Normalized Quadtrees with Respect to Translations. — Computer Graphics and Image Processing, **20** (1): 72—81; New York.

MARTIN, W.N. & AGGARWAL, J.K. (1983): Volumetric Descriptions of Objects from Multiple Views. — IEEE Transactions on Pattern Analysis and Machine Intelligence, **5** (2): 150—158; New York.

MEAGHER, D. (1980): Octree Encoding: a new Technique for the Representation, the Manipulation, and Display of Arbitrary 3-D Objects by Computer.— Techn. Rep. IPL-TR-80-111, Image Processing Laboratory, Rensselaer Polytechnic Inst., Troy; New York.

— (1982): Geometric Modeling using Octree Encoding. — Computer Graphics and Image Processing, **19** (2): 129—147; New York.

NAVAZO, I. (1986): Contribuciò a les Tecǹiques de Modelat Geomètric d'Objectes Poliédrics Usant la Codificaciò amb Arbres Octals. — Ph.D. dissertation, Escola Tècnica Superior d'Enginyers Industrials, Department de Metodes Informatics, Universitat Politèchnica de Barcelona; Barcelona.

— & AYALA, D. & BRUNET, P. (1986): A Geometric Modeller Based on the Exact Octree Representation of Polyhedra. — Computer Graphics Forum, **5** (2): 91—104; Amsterdam.

NOBORIO, H. & FUKUDA, S. & ARIMOTO, S. (1988): Construction of the Octree Approximating Three-Dimensional Objects by Using Multiple Views. — IEEE Transactions on Pattern Analysis and Machine Intelligence, **10** (6): 769—782; New York.

PEUCKER, T. & CHRISMAN, N. (1975): Cartographic Data Structures. — The American Cartographer, **2** (2): 55—69; Washington, DC.

PONCE, J. & FAUGERAS, O. (1987): An Object Centered Hierarchical Representation for 3D Objects: the Prism Tree. — Computer Vision, Graphics, and Image Processing, **38** (1): 1—28; New York.

POSDAMER, J.L. (1982): Spatial Sorting for Sampled Surface Geometries. — Proceedings of SPIE — Biostereometrics '82: 361; San Diego, CA.

QUINLAN, K.M. & WOODWARK, J.R. (1982): A Spatially Segmented Solids Database — Justification and Design. — Proc. of CAD 82 Conf.: pp 126—132; Guildford, UK (Butterworth).

REDDY, D.R. & RUBIN, S. (1978): Representation of Three-dimensional Objects. — CMU-CS-78-113, Computer Sci. Dept., Carnegie-Mellon Univ.; Pittsburgh, PA.

SAMET, H. (1990a): The Design and Analysis of Spatial Data Structures; Reading, MA (Addison-Wesley).

— (1990b): Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS; Reading, MA (Addison-Wesley).

— & ROSENFELD, A. & SHAFFER, C.A. & WEBBER, R.E. (1984): A Geographic Information System using Quadtrees. — Pattern Recognition, **17** (6): 647—656; New York.

— & WEBBER, R.E. (1985): Storing a Collection of Polygons using Quadtrees. — ACM Transactions on Graphics, **4** (3): 182—222; New York.

SHAFFER, C.A. & SAMET, H. & NELSON, R.C. (1990): QUILT: A Geographic Information System Based on Quadtrees. — International Journal of Geographical Information Systems, **4** (2): 103—131; London.

SRIVASTAVA, S.K. & AHUJA, N. (1987): An Algorithm for Generating Octrees from Object Silhouettes in Perspective Views. — Proc. of the IEEE Computer Soc. Workshop on Computer Vision: 363—365; Miami Beach, FL.

TAMMINEN, M. (1981): The EXCELL Method for Efficient Geometric Access to Data. — Acta Polytechnica Scandinavica, Mathematics and Computer Science Series, 34; Helsinki.

— (1982): Efficient Spatial Access to a Data Base. — Proc. of the SIGMOD Conf.: 47—57; Orlando, FL.

VANDERSCHEL, D.J. (1984): Divided Leaf Octal Trees.— Research Note, Schlumberger-Doll Research; Ridgefield, CT.

VON HERZEN, B. & BARR, A.H. (1987): Accurate Triangulations of Deformed, Intersecting Surfaces. — Computer Graphics, **21** (4): 103—110. — [also Proc. SIGGRAPH '87 Conf., Anaheim, CA, July 1987]

WYVILL, G. & KUNII, T.L. (1985): A Functional Model for Constructive Solid Geometry. — Visual Computer, **1** (1): 3—14; Berlin.

YAU, M. & SRIHARI, S.N. (1983): A Hierarchical Data Structure for Multidimensional Digital Images. — Communications of the ACM, **26** (7): 504—515; New York.