Available online at www.sciencedirect.com
ScienceDirect
Computers & Graphics 30 (2006) 854–879
COMPUTERS & GRAPHICS
www.elsevier.com/locate/cag

A survey of the marching cubes algorithm

Timothy S. Newman*, Hong Yi

Department of Computer Science, University of Alabama in Huntsville, Huntsville, AL 35899, USA

Abstract

A survey of the development of the marching cubes algorithm [W. Lorensen, H. Cline, Marching cubes: a high resolution 3D surface construction algorithm. Computer Graphics 1987; 21(4):163–9], a well-known cell-by-cell method for extraction of isosurfaces from scalar volumetric data sets, is presented. The paper's primary aim is to survey the development of the algorithm and its computational properties, extensions, and limitations (including the attempts to resolve its limitations). A rich body of publications related to this aim are included. Representative applications and spin-off work are also considered and related techniques are briefly discussed.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Marching cubes; Isosurface extraction; Indirect volume rendering; Volume visualization; Scientific visualization

1. Introduction

Volume visualization has been applied in many problem domains and, as such, has become an important tool for exploring data and discovering knowledge. Commonly, the domain data to be visualized is *scalar volumetric data*. Scalar volumetric data can be defined as a collection A of 3D points $P_i = (x_i, y_i, z_i)$, each of which has its own scalar value or property [1]. Volumetric data sets are often rendered using indirect volume rendering (IVR) techniques. The indirect techniques involve rendering of an intermediate structure—such as an isosurface—that has been extracted from the data, typically via automatic means.

This paper is a survey of the literature involving one IVR method for scalar volumetric data sets, the marching cubes (MC) algorithm [2]. The MC is probably the most popular isosurfacing algorithm [3], and a rich body of literature has grown up around it. In addition to being a subject of ongoing visualization research, the MC is also influencing research in other areas.

1.1. Historical background

The MC is a sequential-traversal method that was described in 1987 by Lorensen and Cline [2]. MC, while popular, is not the oldest isosurface extraction method. For instance, Wyvill et al. [4] presented a propagation-based method in 1986. That method is somewhat similar to MC, and it is sometimes identified as an MC approach [5]. However, MC and the Wyvill et al. method are distinctive in several ways. For example, they use quite different traversal orderings. The isosurfaces they extract also differ. Due to the differences, and since most teams who have described application of an "MC" methodology have employed the Lorensen–Cline approach, we restrict the MC designation to the Lorensen–Cline approach.

1.2. Some preliminaries

An isosurface can be defined as follows. Given a scalar field $F(P)$, with F a scalar function on $\mathbb{R}^3$, the surface that satisfies $F(P) = \alpha$, where α is a constant, is called the *isosurface* defined by α. The value α is called the *isovalue*. In practice, isosurface extraction usually involves generation of an approximate, piecewise isosurface (usually composed of a collection of triangles) on a sampled scalar field.

*Corresponding author. Tel.: +1 256 824 6619; fax: +1 256 824 6239.
E-mail address: tnewman@cs.uah.edu (T.S. Newman).

0097-8493/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2006.07.021

The isosurface could consist of a single component or of multiple, disjoint ones.

1.3. Utility and motivation

Past usage of many volumetric data sets often involved viewing only 2D cross-sectional "slices" (i.e., a plane of data points) through the data. For example, a slice image from a CT data set of a lobster is shown in Fig. 1. Isosurface extraction, in contrast, can enable one or more phenomena or structures of interest in a data set to be isolated and rendered (using conventional surface-based graphics methods). Moreover, isosurface display is usually fast since most isosurfacing methods output a mesh composed of triangular facets; rendering of triangles is fast on typical graphics hardware. An isosurface extracted from the lobster data set is shown in Fig. 2. The isosurface shown approximately corresponds to the exoskeleton's boundary.

Although MC is not the oldest isosurfacing method, it is very well-known and widely applied. Part of its appeal is that it follows a straightforward, practical approach. The MC has been applied in many application areas, including biochemistry (e.g., [6]), biomedicine (e.g., [7]), deformable modeling (e.g., [8]), digital sculpting (e.g., [9]), environmental science (e.g., [10]), mechanics and dynamics (e.g., [11]), natural phenomena rendering (e.g., [12]), visualization algorithm analysis (e.g., [13]), etc. Processing involving depth maps (e.g., [14]) has also been influenced by MC isosurface extraction, especially in the development of methods based on distance fields (e.g., [15]).

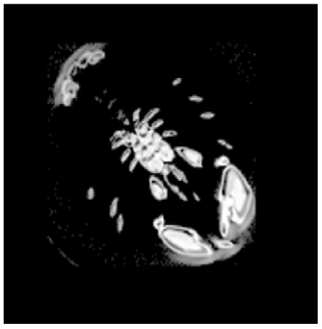

Fig. 1. Cross-sectional view through CT lobster data set.

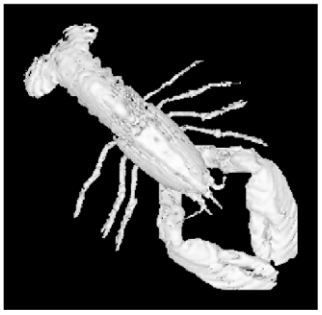

Fig. 2. An isosurface extracted from CT lobster data set.

The MC does pose certain challenges, including achievement of high performance and the potential for the extracted isosurface to have degeneracies and ambiguities. Although extensive work has been done to address these challenges, aside from a brief summary of three early isocontouring methods [16] and brief reviews of some of the isocontour extraction literature [17–19], to our knowledge there has been no survey of the development and extensions of MC. This paper presents such a survey, focusing on publications in the graphics and visualization literature.

1.4. Organization

The paper is organized as follows. First, the standard MC approach is described (Section 2). Then, extensions of the algorithm, in particular those related to traversal, are discussed (Section 3). In Sections 4–6, the algorithm's isosurface assembly component, which includes a triangulation mechanism, is considered. These three sections survey variant isosurface representations (Section 4), the ambiguities in the triangulation mechanism and the known methods for disambiguation (Section 5), and methods that reduce the triangulation's facet count (Section 6). Work related to the final output, especially computing quantities from isosurfaces, is discussed in Section 7. Section 8 concludes the paper.

2. The MC algorithm

The standard MC algorithm, as originally described by Lorensen and Cline [2], takes as its input a *regular scalar volumetric data set*. Such a data set has a scalar value residing at each lattice point of a rectilinear lattice in 3D space. Thus, the lattice point at row y_i ($\forall i$) and column x_j ($\forall j$) of slice S_k ($1<k<n$, where n is the number of slices) is directly adjacent to the lattice points at row y_i and column x_j of slices S_{k-1} and S_{k+1}. The MC processes the volumetric data set by considering the "cubes" C_l that make up the volume. The cubes are defined by the volume's lattice. Each lattice point is a corner vertex of a cube. Fig. 3

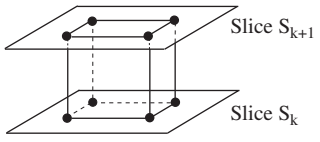

Fig. 3. Illustration of a cube formed on lattice points.

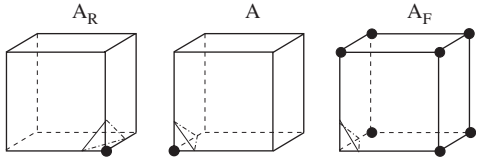

Fig. 4. Illustration of reflective (A with A_F) and rotational (A with A_R) symmetries.

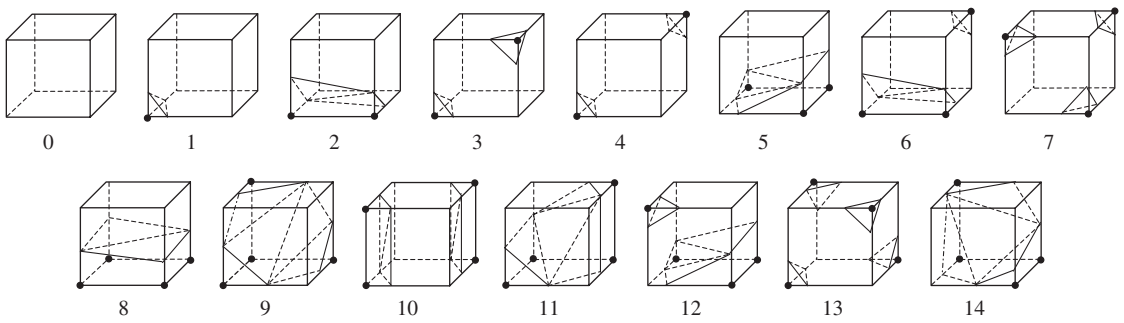

Fig. 5. The 15 basic intersection topologies (using the numbering of [2]).

illustrates one cube defined by lattice points in adjacent slices S_k and S_{k+1}. The lattice lines shown in the figure (i.e., the line segments that join adjacent corner vertices) define the edges of the cube.

2.1. Methodology

The standard MC constructs a facetized isosurface by processing the data set in a sequential, cube-by-cube (scanline) manner. Thus, the approach first processes the m cubes of the first row of the first layer of the data set in sequential order: $C_1, C_2, \ldots, C_m$. During this processing, each cube vertex V_i that has a value equal to or above the isovalue α is marked; all other vertices are left unmarked. The isosurface intersects each cube edge E_j terminated by one marked vertex V_{j_s} and one unmarked vertex V_{j_e}. Any cube that contains an intersected edge is *active*. The computations that find the active cubes can be viewed as the active cube determination component of MC. This component can be implemented as a stand-alone early processing step or integrated with other processing, but in either case it involves data set traversal in sequential, forward-marching order.

Since each of the eight vertices of a cube can be either marked or unmarked, there are 256 (2^8) possible marking scenarios for a cube. Each cube marking scenario encodes a cube-isosurface intersection pattern (i.e., configuration). However, the standard MC considers reflective and rotational symmetry, which results in just 15 marking scenarios. Cubes C and $\hat{C}$ are *reflectively symmetric* if each vertex at position V_i in C has the opposite marking as the vertex at the same position V_i in $\hat{C}$. Two reflectively symmetric cubes (and their cube–isosurface intersections) are shown at the center and right (i.e., cubes A and A_F) in Fig. 4. In the figure, circle symbols denote marked vertices. Cubes that are reflectively symmetric have the same cube–isosurface intersection pattern. Two cubes C and $\hat{C}$ are *rotationally symmetric* if there is some series of rotations R which, when applied to C, transforms C to a new orientation in which the marking at each transformed vertex position V_i is identical to the marking at the same position V_i in $\hat{C}$. The cubes A and A_R in Fig. 4 are rotationally symmetric. Rotationally symmetric cubes have equivalent cube–isosurface intersection patterns (the patterns vary only rotationally by the aligning transformation R).

The 15 unique cube–isosurface intersection scenarios that result when considering both of these symmetries are shown in Fig. 5. We will use the topological case numbering of Fig. 5, which is the same as the standard MC [2], throughout this survey. For each scenario, the standard MC facetization of the intersecting isosurface is shown. Lorensen and Cline [2] stored facetization information (specifically, the vertices of the triangle(s) to be generated, each identified by the edge on which it lies) about the 15 intersection topologies in a look-up table[1] built offline prior to application of MC. This table has usually been hand-built, which can be tedious, but methods to enumerate the cases (e.g., [20,21]) now exist and are discussed later in this paper.

The isosurface–edge intersection locations can be estimated with subvertex accuracy using an interpolation technique. Standard MC employs linear interpolation to estimate the intersection point for each intersected edge. If a unit-length edge E has end points V_s and V_e whose scalar values are L_s and L_e, respectively, then given an isovalue α, the location of intersection $I = (I_x, I_y, I_z)$ has components of the form:

$$I_{\{x,y,z\}} = V_{s_{\{x,y,z\}}} + \rho(V_{e_{\{x,y,z\}}} - V_{s_{\{x,y,z\}}}),$$

where

$$\rho = \frac{\alpha - L_s}{L_e - L_s}.$$

One advantage of the cube-by-cube processing of standard MC is that each edge intersection location only needs to be computed once. Specifically, since each intersected internal edge E_j (i.e., E_j is not on the data set boundary) is shared by four cubes, the point of isosurface–edge intersection I_j on E_j only needs to be computed for one cube C_a. The intersection point I_j can be reused during later processing of three other cubes C_b, C_c, C_d (where $b, c, d > a$) sharing edge E_j. Nevertheless, algorithms that use other traversal patterns can also achieve the same

[1] Although the original MC paper suggested use of a 15-entry table, many realizations have used larger tables for disambiguation and/or parallelization computational advantages.

computational advantages via use of supplemental data structures (e.g., hash tables).

The last step in MC is to generate triangular facets that represent the portion of the isosurface that intersects each cube. The intersection points define the vertices of the triangles, and the collection of the triangular facets across all the cubes forms the triangular mesh (or meshes) that defines the isosurface. The facetization pattern in each cube can be determined from the intersection topology look-up table. The processing steps that build the facetization can be viewed as the isosurface assembly component of the MC.

2.2. Intersection topology count

While the standard MC exploited reflection and rotation to yield 15 intersection topologies, some works (e.g., [22–24]) have described only 14 basic topologies. These works exploit *mirror symmetry*. Cubes C and $\hat{C}$ are mirror symmetric if their vertex markings are symmetric about some face of C. Since Cases 11 and 14 are mirror symmetric, use of all three symmetries in union results in 14 basic topologies.

Table 1
Number of intersection topologies under different symmetries

Symmetry exploited	# topologies
None	256
Reflection	128
Rotation	23
Rotation + mirror	22
Rotation + reflection	15
Rotation + reflection + mirror	14

Table 1, which is adapted from Roberts and Hill [25], summarizes the topology counts for each type of symmetry exploitation. Banks and Linton [20,26] have confirmed these topologies by modeling the marking scenario counting problem as a computational group theory problem that can be solved with readily available software packages. Their approach also allows determination of the number of basic topologies for extending MC to handle degeneracy (i.e., when the isosurface passes through a lattice point due to the isovalue equaling the lattice point's value). The choice of which symmetries to exploit in an implementation should probably be governed by need, in particular in consideration of speed and consistency requirements. This paper considers consistency issues in Section 5.

The 23 topologies resulting from rotational symmetry exploitation are shown in Fig. 6. The figure uses the same case numbering as used in [27–29]. (Ref. [27] also includes an unnecessary alternate set of topologies, however.) Since the exploitation of rotational—or more accurately, the non-exploitation of reflective [28,30]—symmetry overcomes one of the key problems of the standard MC (described later in Section 5.3.1), the topology table shown in Fig. 6 is a practical choice for new implementations of MC. It is also practical for such implementations to utilize supporting tables (available in [28,29]) that allow quick determination of each cube's topology and rotational orientation based on the vertex markings.

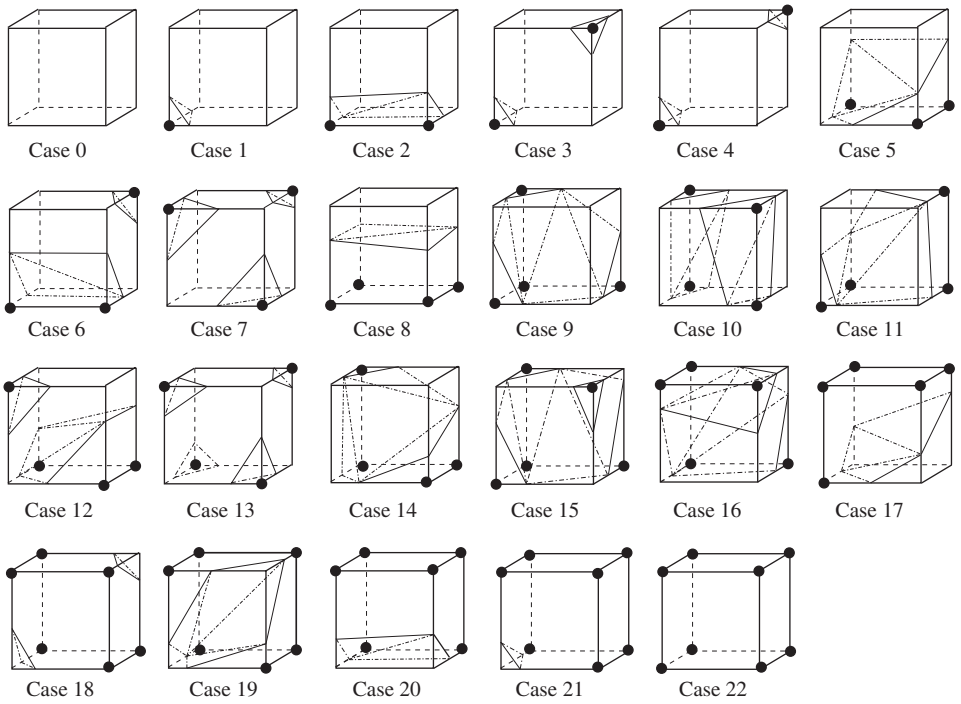

Fig. 6. The 23 intersection topologies (using the numbering of [27–29]) that result when only rotation is exploited.

3. MC extensions: data types and speed-ups

The MC has been extended in a number of ways. Here, the extensions related to traversal through the data are considered. Primary foci include (1) extensions of the traversal mechanism to additional types of data (Section 3.1) and (2) computational improvements that limit unnecessary effort, especially during traversal (Section 3.2), or that use parallel and distributed processing (Section 3.3). For comparison, alternative traversal mechanisms are also discussed (in Section 3.4).

3.1. Extensions to other input types

Next, the extensions of the MC to other input types are described. These extensions widen the utility of MC. To assist in comparison, Table 2 classifies such extensions according to the form of their inputs and outputs.

When the MC is extended to processing of other types of data, the spatial unit of processing is often called a *cell*. In this paper, we will use that term to denote any spatial unit of processing.

3.1.1. Multi-resolution rectilinear data

Weber et al. [31] have extended the MC to rectilinear grid data that has a hierarchy of resolutions in certain regions. An example that contains two levels of such data is shown in Fig. 7; the dense resolution subgrid in the center of the figure refines part of the coarse mesh. Their approach first extracts the isosurface (using MC) separately in each subgrid. It then forms irregularly shaped cells (pyramids, triangle prisms, deformed triangle prisms, deformed hexahedra, or tetrahedra) that "stitch" together the subgrids. Finally, isosurface facets are constructed in the irregularly shaped cells. The approach allows creation of isosurfaces with no "cracks" between subgrids. Other methods for isosurface crack-patching include [50–54].

Table 2
Classification of extensions of marching cubes to other input types

Input type	Output type	
	Triangular mesh	Spline mesh
3D rect. multi-res. grid	[31]	NA
4D grid	[25,32–38]	NA
n-D rect. grid	[20,21,39,40]	NA
Curved grid	[41]	NA
Other cell types	[31,42–44], [45][a], [46,47]	[48,49]

[a]The entry marked extracts the isosurface in tetrahedral subcells (of a rectilinear grid).

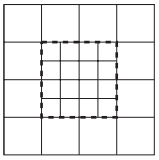

Fig. 7. An example of a component of a multi-resolution data set. The central region enclosed by the dashed line segments was sampled at a higher resolution than the rest of the data.

3.1.2. Non-rectangular data

Means to extend MC to some types of non-rectangular data have also been described. For instance, an extension to cylindrical and spherical data that uses transformation formulas based on the relationships of rectangular coordinates with cylindrical and spherical coordinates has been described [41]. Visualization of such data within its natural coordinate system can aid understanding and analysis.

Gallagher and Nagtegaal [48] reported one of the earliest extensions of MC to unstructured grid data (e.g., sparse data used for finite element analyses). The approach couples sequential, forward-progression march with intersection determination by table look-up and is usable in polyhedral regions of space, particularly for regions with tetrahedral- and wedge-shaped cells. The approach also produces a higher-degree isosurface than does the MC. Interested readers might consult [49] in addition to [48] since [49] contains some extra details.

Payne and Toga [45] have also described extension of the MC to tetrahedral-shaped cells (*tets*). Their approach is the initial demonstration of what has come to be called the *marching tetrahedra* (MT). The term MT was actually coined by Shirley and Tuchman [46], who described the MT steps but did not reduce them to practice. MT involves processing a mesh of tets by sequential, forward march through the tets with intersection topology determination via table look-up in each tetrahedron. Specifically, MT first marks each tetrahedral vertex whose scalar value equals or exceeds the isovalue. There are 16 (2^4) distinct markings for each tetrahedron, although exploiting reflection and rotation results in just three patterns [55,56]. Then, the intersection points are found, typically using linear interpolation. Finally, the triangulation is built on these points.

Often, MT has been applied to rectilinear grids. In such cases, the grid cubes must first be subdivided into tetrahedral subcells. Schemes for subdivision into either five or six tets exist. Several of the schemes have been described by Carr et al. [57]. When MT is applied to rectilinear grids, it will produce more isosurface facets than does MC [42,58]. Methods to simplify the MT triangulation (e.g., the wrapper [59] and regularized marching tetrahedra [56] algorithms) are known, however.

The MT has been popular in many application areas and has spawned spin-off work of its own (e.g., determining the space swept by a solid [60] and reconstructing interfaces between materials [61]). Readers interested in an overview of MT may wish to consult the survey paper of Elvins [62].

Extension of MC to some other cell types, including octahedra (e.g., [42,47]), hexahedra (e.g., [42]), and to some irregularly shaped cell types (e.g., [31]), as mentioned in

Section 3.1.1, has also been demonstrated. The MC has also been extended to non-rectangular data by binning of the data points into cube-shaped containers followed by extracting the isosurface in sequential forward march over the bins, as described recently by He et al. [43].

3.1.3. Higher-dimensional space

MC has also been extended to enable application to higher-dimensional data sets. For example, approaches that enable processing of 4D data sets [25,38] have been described.

One difficulty in extending MC to higher dimensions is determining the intersection topology look-up table. To address this difficulty, Bhaniramka et al. [21,39] have developed a convex-hull-based algorithm that automatically generates the table for a regular rectilinear grid (or grids of any type of convex polyhedra) of any dimension. For each given marking scenario of a d-dimensional hypercube, the approach first finds the midpoint of each edge the isosurface intersects, then constructs the convex-hull of the set of all such midpoints together with all marked vertices in the hypercube. After this step, any $(d-1)$-dimensional facets that lie on the faces (or boundary) of the hypercube are removed from the constructed convex-hull. The remaining $(d-1)$-dimensional facets comprise that hypercube's isosurface intersection configuration. Bhaniramka et al. have also shown that the approach can construct a consistent facetization for all intersection scenarios. Banks et al. [20] have independently confirmed Bhaniramka et al.'s results for 4D.

A second difficulty for high-dimensional data is that intersection topology table size increases with dimension. One way to reduce this added memory pressure is to store only the most common topologies in a hash table [40].

3.1.4. Time-varying data

Another area of extension is to time-varying data. Such data often consists of several volumes, each collected at a different time point. Typically, MC is applied to each time point (e.g., [36]), with the collection of resultant isosurfaces commonly displayed as transparent overlays. Another approach for time-varying volumetric data represented as 4D arrays (in which time is the fourth dimension) is to apply the Weigle and Banks [37,38] 4D isosurface/isovolume approach. In their approach, an isovolume is generated that represents the implicit function $f(x,y,z,t)=\alpha$, where t is the time, (x,y,z) represents position, and α is the isovalue. The generated isovolume is the space that is swept by the isosurface over time. Other methodologies that utilize MC-based isosurface extraction from time-varying data sets have also been described (e.g., [32,33,35]). Since those approaches focus mainly on avoiding computation during traversal, they are discussed with the other computation avoidance techniques (Section 3.2).

One challenge in isosurface extraction for such data is that if there are a moderate number of time points, the data is often too large to reside in-core. Nevertheless, display of isosurfaces extracted from time-varying data can aid in understanding dynamic phenomena in many application domains.

3.2. Computation avoidance

Interactive-rate data exploration and visualization can be advantageous for many applications. Achieving such performance levels has always been a challenge and, as the size of data sets has grown, has remained one. One popular way to hasten MC has been to avoid unnecessary operations on the non-active (empty) cells, since 30–70% of the processing time involves those cells [63]. Although some processing involving empty cells is probably inevitable, since generating a correct isosurface requires each cell to be visited at least once to learn if the cell is active, avoiding unnecessary operations on non-active cells is an effective way to accelerate the MC.

Methods that minimize unnecessary operations on non-active cells are surveyed next. Table 3 summarizes the methods according to processing characteristic. Typically, these methods minimize the operations via representations that can efficiently encode regions of non-activity, often by recording *extremes* (i.e., minimum and maximum scalar values) of each region, since any region whose *interval* (i.e., its range of extremes—also called its *min–max range*) does not contain the isovalue is not active. Thus, a major focus here is representations that encode non-active regions. It should be noted that since many representations preprocess the data to determine activity, it is often possible to use either MC or other approaches to form the isosurface facets in the active cells.

3.2.1. Hierarchical geometric approaches

One popular representation to encode non-active regions is the hierarchical octree, which is a hierarchical geometric data structure. Wilhelms and van Gelder [69] were the first to use octrees to avoid examining empty cells. The octree root node refers to the entire volume. Each child node refers to an (approximately) equal-sized subvolume of the volume described by its parent. Normally, each non-terminal node has eight children, although if data set size is not a power of 2 in all dimensions, some tree levels will not be full. (Full octrees do not need to store explicit node addressing information. Octrees that do not store addressing information are *pointerless*.) Octrees that produce equal-sized subdivisions are called *even-subdivision octrees*.

Table 3
Non-active cell avoidance papers, categorized by approach

Class	Paper citations
Hierarchical geometric	[34,35,64–69]
Interval-based	[32,33,70–79]
Propagation-based	[52,80–83]

In such octrees, the terminal nodes describe regions composed of eight (or fewer) contiguous cells. To aid in avoiding empty cells during isosurface construction, each octree node also stores the extremes of the region described by the node. It is worth noting that some early uses of octrees did not store cell extremes, which reduces storage requirements but necessitates visiting non-active cells of active regions during isosurface extraction. Building the octree (i.e., *preprocessing*) requires visiting each cell once. However, each subsequent isosurface extraction utilizes the octree to only visit the cells of nodes whose interval includes the isovalue. Usually, the time saved by avoiding empty cells is more than preprocessing time. In some environments, fast octree creation is also possible (e.g., using multi-threading on shared-memory computers [84]).

If there are n cells, p of which are active, octree-based isosurface extraction has a worst-case time complexity of $O(p + p\log(n/p))$ [77]. Octree creation time complexity is $O(n\log(n))$, but multiple extractions can reuse the octree. Of course, the standard MC has $O(n)$ complexity.

Octrees can also be used to extract only the part of an isosurface that is visible from a given viewpoint, as first described by Livnat and Hansen [66]. Avoiding non-visible regions makes such approaches well-suited for remote visualization, especially on slow networks. In the Livnat and Hansen approach, the octree is traversed in a top-down, front-to-back order to avoid non-active cells and to enable coarse visibility testing. Specifically, node occlusions are detected during traversal through a viewpoint-centric projection of each non-terminal node's subvolume onto a virtual screen. Z-buffering is then applied to cells that survive the coarse visibility test. Recently, approach extensions that replace small isosurface facets with point primitives and that use multi-threading have been presented [85]. Other improvements, such as pruning occluded regions from the octree as soon as they are known to be occluded [86] and partitioning space along spherical, rather than cartesian, coordinates [87] are also possible.

The even-subdivision octree of a volumetric data set whose size is not a power of 2 cannot be stored with optimal efficiency. One way to reduce the storage needed for such octrees is to not subdivide any node that describes a region composed of few (e.g., 32) cells, as first described by Globus [65]. Although such an approach can reduce the octree's storage requirements, Wilhelms and van Gelder [69] have proposed a more space-efficient approach for octree organization, the *branch-on-need octree* (BONO). During the recursive subdivision process that constructs the BONO, the volume subdivisions are created only when necessary. In particular, the BONO does not subdivide any subvolume composed entirely of non-active cells. The BONO is space-efficient because unnecessary divisions are avoided and because its divisions are of optimal size; each division in a dimension D_i will create one subdivision of size $2^{q_i} < n_i$, where n_i is the size in dimension D_i of the region to be divided and q_i is the largest integer that maintains the condition. (The other subdivision will be of size $n_i - 2^{q_i}$.) In addition, subdivisions that define subvolumes composed entirely of non-active cells are allocated no space in the BONO. It should be noted that, unlike pointerless octrees, the BONO does require storage of pointers between levels (i.e., to child nodes). The storage requirements can be reduced somewhat by using clusters of adjacent cells that have nearly uniform scalar values as the units that BONO organizes [68]. The resultant isosurface can have "cracks" between blocks of different sizes, however.

The temporal branch-on-need (T-BON) tree [34,35] extends the BONO to time-varying data. For each time point, a T-BON stores an octree of node extreme values. In T-BON-based isosurface extraction, the time point's octree is first traversed to determine the needed data blocks. These blocks are then read from disk, and isosurface extraction is performed in them. Subsequent extractions using a different isovalue at the same time point reuse resident data blocks and load only the additional needed data blocks. Thus, the T-BON can minimize disk accesses, which is especially useful when the data set cannot reside entirely in-core.

The 3D pyramid [64,67] is another hierarchical geometric data structure useful to avoid traversal of empty cells. Like the BONO, each node in the pyramid structure stores the interval for the subvolume the node describes. The 3D pyramid is essentially equivalent to a pointerless full octree, but it includes a mechanism to handle data sets that are not powers of 2 in all dimensions. Unlike a BONO, the 3D pyramid subdivides all nodes that are at non-terminal levels of the representation; it need not be reconstructed for each isovalue.

3.2.2. Interval-based approaches

Interval-based representations, which group cells based on cell intervals, are another class of data structures that are useful in avoiding traversal of empty cells. One advantage of many interval-based approaches is their operational flexibility; since these approaches operate in an interval space rather than in the geometric space of the mesh, they are typically applicable on either regular or unstructured (e.g., irregular tetrahedral) data. Many interval-based approaches have been presented in the literature, and these approaches are summarized next.

The *active list* of Giles and Haimes [76] was an early interval-based representation for avoiding empty cells. The active list is built from two supplemental lists: an ascending-order list of cell minima and a descending-order list of cell maxima. The initial active list contains the minima list cells whose minima are less than the isovalue α but greater than $\alpha - \Delta L$, where ΔL is the largest cell interval. Isosurface extraction is performed only in the active list cells that are actually active. When different isoqueries are made, the lists of minima and maxima are used to add cells to the active list. Although active list-based isosurfacing has worst-case complexity of $O(n)$,

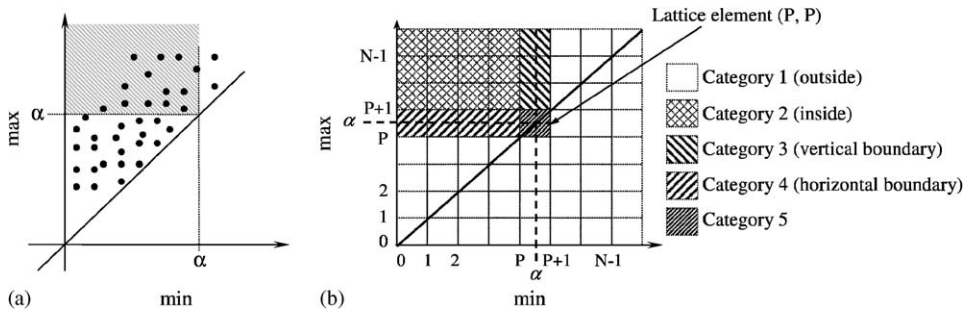

Fig. 8. Span space illustrations. In (b), quantile categorization based on isovalue α is shown ((b) is an adaptation from [78]): (a) search in span space and (b) quantization of span space.

where n is the number of cells, it can be effective when ΔL is small.

Another early approach was the *span filtering* [75], which can operate on either regular or unstructured data. Span filtering equally divides the data set interval into subranges. A bucket is associated with each subrange and holds lists of the cells whose minima are within its subrange. Each list holds the cells of a certain *span length* (cell span length is the number of buckets whose ranges are intersected by its interval). Active cells are located by first searching the lists of the bucket U_l whose range includes the isovalue. Each bucket U_j, $j<l$, is also searched for active cells, although only the linked lists that store cells of span length greater than $l-j$ need be searched. Span filtering has been reported to significantly reduce finite element data isosurfacing time [75].

The *sweeping simplices* [79] is an approach that can allow fast isosurface extraction from unstructured data. It also uses bucketized subranges, although its subranges overlap. Each cell is assigned to the bucket that completely contains the cell's interval. Candidate active cells are located by retrieving the cells from each bucket whose interval contains the isovalue. Then, each candidate active cell is stored in two lists, one sorted by cell minima and the other by cell maxima. Finally, actual active cells are determined by searching the lists; each cell whose minimum is not greater than the isovalue α and whose maximum is not less than α is active.

3.2.2.1. Span space-based methods. Span space-based methods exploit a 2*D span space* to avoid examining empty cells. The 2D span space, which was first described by Livnat et al. [77], has axes that are cell minima and maxima, which we will denote as the min and max axes, respectively. Each cell is mapped to a point in span space; a cell with minimum scalar value M_a and maximum scalar value M_b maps to (M_a, M_b) in span space. Active cells are located by searching span space, as illustrated in Fig. 8(a) for an isovalue α. The cells that are active have a cell minimum less than or equal to α and a cell maximum that is greater than or equal to α. The area in span space containing the active cells is the *active area*. An active area for an isovalue α is shown as a shaded region in Fig. 8(a).

Several span space-based methods for efficient isosurface extraction have been presented and are described next. The primary difference in the approaches is the search mechanism used.

The first span space-based method was the Near Optimal ISosurface Extraction (NOISE) algorithm of Livnat et al. [77]. The NOISE algorithm organizes the span space data using a Kd-tree,[2] which can be traversed quickly to find the active cells. Livnat et al. have reported that NOISE-based isosurfacing has excellent computational performance; its time complexity is $O(\sqrt{n}+p)$, where n is the number of cells and p is the number of active cells. The time complexity of the NOISE algorithm's preprocessing activities is $O(n\log(n))$.

The use of the Kd-tree to organize a span space that is based on cell minima and maxima allows fast isosurface extraction, although the cost of the speed is the storage of the cell intervals. (Cell addressing information must also be stored.) The storage requirement could be reduced by using NOISE on *region* minima and maxima. Such a practice would, of course, increase isosurface extraction time due to the need to visit all cells (i.e., not just the active cells) within an active region.

Another span space-based method, the isosurfacing in span space with utmost efficiency (ISSUE), has been presented by Shen et al. [78]. In ISSUE, span space is quantized into an $N \times N$ regular lattice, where N is user-specified. Given an isovalue α, each lattice element (i.e., quantile) is classified into one of five categories based on its location, as illustrated in Fig. 8(b). A quantile can be either outside the active area (Category 1), completely inside the active area (Category 2), on the vertical (but not horizontal) boundary of the active area (Category 3), on the horizontal (but not vertical) boundary of the active area (Category 4), or on both the horizontal and vertical boundaries of the active area (Category 5). The cells that map into Category 2 quantiles are active. The cells that map into Category 3–5 quantiles are potentially active. A Category 3 cell is active only if its minimum is less than the α. Likewise, a cell in a Category 4 quantile is active if its maximum is greater than the α. For the cells that map into

[2]The Kd-Tree is a *K*-dimensional generalization of a binary tree. Kd is an acronym for *K*-dimensions.

the Category 5 quantile, both minima and maxima must be examined to determine activity. By restricting the span space search, ISSUE enables faster detection of active cells than many other span space based techniques. Shen et al. have reported the average time complexity of ISSUE is $O(\log(n/N)+\sqrt{n}/N+p)$, assuming n cells, p of which are active, and a user-specified lattice size of N.

ISSUE tends to yield very efficient computation. Yet, standard ISSUE can exhibit only modest I/O performance and consume moderate amounts of storage. Methods to address these challenges using bucketing and transform coding have been described. In bucketing, a row or column of span space quantiles are stored into contiguous memory locations or disk blocks. The bucketized span space was introduced by Sulatycke and Ghose [88] and has been used by a few teams (e.g., [89,90]). As used by Bordoloi and Shen [70], the transform coding performs a fast transform operation on the span space followed by a non-uniform quantization that effects a lossy compression of the transformed span space. Locating the active cells can be performed directly on the compressed representation. Use of transform coding has been observed to consume only one-third the storage of the ISSUE and interval tree data structures (the interval trees are discussed later in this section) without materially impacting isosurface extraction time [70].

For time-varying data, the *temporal hierarchical index tree* (T-HIT) of Shen [33] can be used. The T-HIT is a binary tree of *time* intervals that organizes cells based on cell *temporal variation* (the change of a cell's extreme values over a time interval). Each node stores all the cells that have a low temporal variation for the time interval associated with the node. To determine active cells for a time point t, the T-HIT is searched along a path from the root to the node for time point t, and at each visited node, ISSUE-based MC is applied. Using T-HIT enables good time and space performance [33].

3.2.2.1.1. Interval trees and related structures. Most other interval-based approaches have organized cell extreme values in *interval trees* or related data structures. The interval tree is a balanced binary search tree that enables efficient retrieval of ranges that contain a given query value.

Cignoni et al. [73,74] have utilized the interval tree to search span space for active cells. Each node of their interval tree is associated with a distinct extreme scalar value, and each non-terminal node stores two lists that organize the cells whose intervals include the value associated with the node. One list, T_a, is an ascending-order list of cell minima and the other, T_d, is a descending-order list of cell maxima. To find the active cells associated with a given isovalue α, the interval tree is visited in *modified preorder*. Specifically, if α is less than the extreme value ω associated with the node, all cells stored in the node's list T_a that have a minima less than α are active, and the node's left subtree (which holds the nodes associated with values $L<\omega$) is visited next. The right subtree is not visited. If $\alpha>\omega$, all cells in T_d with maxima greater than α are active, and the node's right subtree (which holds the nodes associated with values $L>\omega$) is visited next. The left subtree is not visited. If $\alpha=\omega$, all the cells in the node's lists are active and neither subtree is visited. The Cignoni et al. algorithm has a low-order time complexity $O(p+\log(v))$, where p is the number of active cells and v is the number of unique extreme values.

For data sets that are too large to fit into main memory, slow disk access, especially due to excessive paging [88], can become a major performance problem for MC. A few methods to improve performance for large data sets by reducing memory requirements or organizing secondary memory accesses via interval tree-based processing have been described. For example, Chiang and Silva [71] have described the *I/O-optimal interval tree* data structure and its use in isosurface extraction. The I/O-optimal interval tree is constructed in secondary storage and used to find the active cells. The active cells are the only ones loaded into RAM. More recently, Chiang et al. [72] have proposed a two-level indexing scheme that improves the computation time for extractions based on the I/O interval tree. The scheme partitions the original data set into clusters of cells called *meta-cells*, and the interval tree is based on meta-cell intervals. Only meta-cells that are determined to be active are loaded into RAM. Chiang [32] has also extended the meta-cell concept for I/O-efficient processing for large time-varying data sets. The extension amends the meta-cells with a data structure that organizes vertex data by time. This extension has been coupled with an indexing structure called a *time tree* that enables I/O-optimal search for and access to the active meta-cells [32]. For visualization pipeline application, it may be beneficial to store meta-cells on disk in isosurface geometry order rather than forward-marching order [91].

Another strategy to reduce I/O for a series of isoqueries is to use the skip-list data structure to encode extreme values [92]. The skip-list stores, for each extreme value, a list of the cell edges whose activity state changes at that value. It can be maintained hierarchically to allow rapid determination of the changed cells even for large changes in the isovalue [92]. Thus, only the cells whose activity status changes from inactive to active need to be loaded when the isovalue is changed.

3.2.3. Propagation-based approaches

Propagation-based approaches to isosurface construction can also naturally avoid traversal of empty cells since the propagation process visits only active cells. Propagation-based approaches do not use forward cube-by-cube marching but rather propagate outward from some active seed cell. They are discussed here because such techniques can be used to "gather" active cells (or subvolumes) on which MC-like processing is then applied.

Most propagation-based approaches require manual selection of seed cells since automatic selection can be challenging. For example, the method of Shekhar et al. [52]

recursively propagates from a user-specified cell using isosurface connectivity. However, a few automatic seed cell selection schemes exist, including methods based on search through graphs (1) of local minima and maxima (i.e., *extrema graphs*) [81,82], (2) that capture isocontour topology [83], or (3) of cell adjacencies [80].

3.2.4. Comparisons

An advantage of using hierarchical geometric data structures is that they enable efficient isosurface extraction. In particular, these data structures tend to be more compact and allow quicker access than those used for interval-based approaches. However, interval-based approaches that are coupled with seed cell determination methods (such as the method of [80]) can also be compact. In addition, the Kd-tree can be as compact as a full pointerless octree, assuming that the Kd-tree stores a cell's address as compactly as it stores an extreme value. One disadvantage of hierarchical geometric methods is that extension to non-regular data sets can sometimes be challenging.

A comparison of the computational complexity of isosurface extraction based on one typical hierarchical geometric approach (i.e., octrees [69]), several typical interval-based approaches (i.e., active lists [76], NOISE [77], ISSUE [78], and an interval tree-based approach [73,74]), and one typical propagation-based approach (i.e., that exploits extrema graphs [81]) is presented in Table 4. In the table, n is the number of cells, p is the number of active cells, N is the user-specified lattice size, and v is the number of different extreme values. Preprocessing times for the approaches are not included, however.

Sutton et al. [93] have presented a comparative study of the computational performance and memory behavior of the standard MC and five accelerated approaches that use auxiliary data structures to avoid non-active cell traversal (i.e., approaches based on BONO [69], propagation [80], the interval tree [73], ISSUE [78], and NOISE [77]). Sutton et al. self-implemented and tested the approaches on two data sets in the same hardware and software framework. They found all five methods to deliver faster performance than standard MC, with BONO the fastest. They also found that their implementation of the interval tree technique (which, in the worst case, will have theoretically optimal performance) had the least acceleration on their machine due to many cache misses. Another case study by Zhang et al. [90] found that, for single-threaded Pentium 4 CPUs, BONO-based approaches were fastest, although ISSUE-based approaches were nearly as fast. Although the approaches implemented in these studies might exhibit variant behavior on different architectures or for other data sets, the studies suggest that consumption of more memory tends to yield (but does not necessarily deliver) improved performance.

Saupe and Toelke [94] have also studied the time–space tradeoff for isosurface extraction and presented a memory-constrained optimization framework that uses hybrid isosurface extraction approaches to optimally adapt to any given amount of memory. The framework uses a binary spatial partition (BSP) tree whose leaf nodes encode regions. In each leaf node, there is an accelerated search for the isosurface, using whichever acceleration method (from a family of possible methods) which best satisfies an optimality principle.

3.2.5. Minimizing number of extractions

An alternate perspective on improving the efficiency of MC is to view isosurface extraction in the larger context of data discovery. Data discovery using MC is often a trial-and-error process involving repeated selection of isovalues until a clear trend or phenomenon is discerned. Several methods to reduce the number of isoqueries through determination of key isovalue(s) have been reported. For example, approaches based on histogramming/moments [95–98], examining the gradient field [99,100], examining the critical points [3] and critical regions [101] of an interpolating field, and consideration of isosurface area [102] have been reported. Another scheme uses look-up tables that consider regional data values [103].

3.3. Parallel and distributed approaches

Parallel approaches have often been considered as a way to improve performance of time-consuming graphics and visualization activities. Many of the issues related to efficient parallelization of traditional graphics and direct volume rendering tasks have been summarized by Crockett [104]. Since the MC exhibits a degree of intrinsic parallelism (e.g., cube faces not shared with previously visited cubes can be processed independently), its paralle-

Table 4
Comparison of computational complexity of five representative acceleration strategies

Approach	Class	Computational complexity for extraction
Octree-based [69]	Hierarchical geometric	$O(p + p\log(\frac{n}{p}))$ (worst case)
Active lists [76]	Interval-based	$O(n)$ (worst case)
NOISE [77]	Interval-based	$O(\sqrt{n} + p)$ (worst case)
ISSUE [78]	Interval-based	$O(\log(\frac{n}{N}) + \frac{\sqrt{n}}{N} + p)$ (average case)
Interval tree-based [73,74]	Interval-based	$O(p + \log(v))$ (worst case)
Extrema graph-based [81]	Propagation-based	$O(n)$ (worst case)

Table 5
Load-balancing schemes, categorized by load-balancing and by computational paradigm

Balancing	SIMD	MIMD
Dynamic	NA	[23], [88][a], [105][a], [106], [107][a]
Static	[108]	[78,85], [88][a], [90,109], [110][a], [111,112], [113][a], [114], [115][a], [116][a], [117][a], [118][a]

[a]The entries marked directly address out-of-core operation.

lization offers the potential for performance improvement. In this subsection, load-balanced-parallel, vector- and pipeline-parallel, and distributed approaches to MC computation are surveyed.

3.3.1. Load-balancing

Many of the parallelized approaches have focused on static or dynamic load-balancing strategies to enable efficient utilization of multiple processors. Static strategies assign a subset of the total work to each CPU before parallel execution begins and do not vary the work assignment during it. Dynamic strategies allow work redistribution during execution. Although dynamic strategies tend to keep computational resources well-utilized, they also tend to consume some resources due to the extra overhead of work redistribution during execution.

The approaches surveyed here are summarized in Table 5, categorized according to load-balancing paradigm on one axis and computational paradigm on the other axis. Some of the approaches have also directly considered the challenges of data sets that are too large to be processed entirely in-core. The parallel out-of-core approaches are discussed within the context of their load-balancing characteristic.

3.3.1.1. Dynamic load-balancing approaches. One of the first dynamic load-balancing approaches based on MC or its extensions was the approach of Miguet and Nicod [23] for distributed-memory computers. In the approach, layers of the data set are first evenly divided among the CPUs. Then, per-layer work estimates are computed based on weighted sums of the number of cells and interpolation points, with weights determined by linear regression. During execution, layer assignments are adjusted between processors to better balance the workload.

Gerstner and Rumpf [106] have described a parallelization of their multi-resolution technique (Section 6.1) for a shared-memory computer. The approach initially assigns each CPU an independent subtree of their representation's hierarchy. Each processor then recursively computes the isosurface in its assigned subtree. CPUs that become idle are supplied the largest remaining unprocessed subtree subcomponent from another CPU.

3.3.1.1.1. Out-of-core dynamic balancing. Sulatycke and Ghose [107] have presented a multi-threaded approach to MC that delivers good performance on dual- and quad-processor SMPs for data sets too large to reside in-core. The approach organizes the data on disk using an interval tree stored in a retrieval-efficient format. A set of circular data buffers are filled from disk by a thread dedicated to I/O. As the buffers fill, computation threads retrieve blocks of data from a buffer and extract the component of the isosurface in the retrieved data. One alternative to the approach in [107] improves I/O times by about 33% by organizing data on disk using a bucketed span space [88]. In the alternative, subsets of span space are initially gathered into regions. Then, within each region, storage of records of nearby span space buckets is interleaved. A second alternative uses chessboarding [119], which stores on disk information about only the even-numbered cells of the data set. Information about the non-stored odd-numbered cells can be retrieved from the stored cells. Chessboarding was found to reduce interval tree storage needs by 75%.

Chiang et al. [105] have described an efficient out-of-core scheme for cluster computers that can be applied to both rectangular and unstructured grids. It uses a master CPU to first find all active meta-cells based on the I/O-optimal interval tree and the two-level indexing scheme (Section 3.2.2). Each slave CPU is then assigned an active meta-cell by the master CPU. Each slave computes the isosurface (via MC for rectilinear data and MT for unstructured grid data) in its assigned meta-cell and then receives a new assignment, until all cells have been processed. The scheme can reasonably balance workload among the processors.

3.3.1.2. Static load-balancing approaches. Techniques for static load-balancing of MC are described next. For comparison and completeness, we also describe a few parallel isosurface extraction methods that do not follow a forward march through the cubes.

Some of the attempts to parallelize MC, especially early attempts, involved evenly dividing cells [108], data set layers [109], or other equal-sized subvolumes [114] among processors. The cell-based approach was implemented initially on a SIMD computer and did not exploit topological symmetries since complex conditional operations (e.g., conditional rotations) inhibit SIMD computation. The subvolume- and layer-based approaches were implemented in MIMD environments. All three approaches achieved reasonable speed-ups, although full balancing of workload was a challenge for the subvolume-based approach due to the different number of active cells (and hence, different workload) in each subvolume.

The lattice subdivision-based approach (Section 3.2.2) has also been parallelized using static load-balancing by

Shen et al. [78]. The parallel technique first distributes quantiles of the span space in a round-robin manner to the CPUs. The degree of load-balancing depends on the quantile resolution, although there is a space cost for the better load-balancing of finer resolutions. In order to achieve goodness in both load-balancing and memory utilization, Shen et al. used high-resolution quantiles during distribution but low-resolution quantiles during the isosurface extraction steps. Recently, Zhang et al. [90] have found that use of a block-based span space can improve the memory performance of the Shen et al. approach. Block-based span space is defined on non-unit-sized subdivisions of the volume; each block represents a set of adjacent cells.

Two static load-balancing approaches that extract only the portion of the isosurface visible from a given viewpoint have been presented. One is the Livnat and Tricoche [85] approach (Section 3.2.1). It splits computation based on screen projection; each CPU is responsible for extraction for an equal portion of the screen area. The other is the octree-based approach of Gao et al. [112]. In it, an image-based partitioning of space is used to distribute approximately equal numbers of active cells to each CPU. The active cells are found by traversing the octree in front-to-back order. Each CPU extracts the portion of the isosurface in its partition. Performing visibility detection (i.e., to find all visible active cells) between the octree traversal and data distribution steps has been found to allow better load-balancing since it allows distribution based on the *visible* active cells [111]. Use of an octree based on spherical partitioning of space can also improve performance [87].

A recent comparative study of the performance of several static load-balancing methods using multi-threading on uni- and dual-processor desktops by Zhang et al. [90] found that methods that evenly divide cells, subvolumes, or layers (e.g., [108,109,114]) were slower than span space-based methods (e.g., [78]). The same study found octree-based methods to be marginally faster than span space-based methods. However, octree-based methods require more memory and set-up time than span space-based methods.

3.3.1.2.1. Out-of-core static balancing. Kurc et al. [113] have developed the active data repository (ADR) framework for very large data sets. ADR stores a data set as a set of variably sized data chunks. The chunks are distributed across the disks to fully utilize main memory and I/O bandwidth. Isosurfacing based on ADR involves use of an indexing scheme to determine the *active* chunks. Isosurfacing is performed only on these chunks.

Bajaj et al. [110] have presented a parallelized version of a propagation-based isosurfacing method that has been the basis for several follow-ons. The method initially partitions the data set into fixed-sized *blocklets*, which are small sets of adjacent cells. Blocklets are then clustered into *cell blocks* based on the *contour spectrum* [95]. The contour spectrum is a collection of plots of a contour attribute (i.e., the isosurface–blocklet intersection count) over all potential isovalues. Blocklets are clustered into cell blocks such that each cell block has a similar contour spectrum. An (approximately) equal number of cell blocks are distributed to each processor. The method can allow reasonably efficient out-of-core performance for large data sets, with linear-class speed-up of the computational steps. Workload on each CPU can vary significantly, however. Zhang and Newman [115] have reported a more accurate load estimation scheme that can allow Bajaj et al.'s basic formulation to exhibit better load-balancing. Zhang and Newman [115] have also shown that the formulation can be applied to MC-based (rather than propagation-based) isocontouring. The more accurate load estimation scheme for MC isosurfacing can also be combined with interval-based organization of data on disk to deliver improved I/O and isosurfacing time [116] through minimization of unnecessary I/O and redundant computations.

Zhang et al. [117] have presented a variation on the parallelized seed-set approach [110] that distributes data among CPUs based on the contour spectrum and then uses the I/O-optimal interval tree [71] (Section 3.2.2) to build a search index structure for each CPU's on-disk data. The search structure enables loading of only the active blocks into RAM. The isosurfaces extracted from active blocks are finally compressed and transmitted to parallel rendering servers whose output images are composited and displayed on a multi-tiled screen. Recently, Zhang et al. [118] have extended their method for view-dependent operation. This extension first finds the active blocks that occlude the viewing of other blocks, then extracts the part of the isosurface in these blocks, and finally extracts the isosurface in any remaining blocks that are not occluded by the initial isosurface extraction. The extension also uses a random block distribution, which yields better parallel performance than contour spectrum-based distribution. However, the time to read the active blocks into memory can be high [116]. One comparative study has also found that while the method of Zhang et al. [118] is faster than the earlier seed-set approach, random block distribution is still not as fast as distributing blocks based on accurate work estimation coupled with interval-based disk data organization [116].

Lastly, Sulatycke and Ghose [88] have demonstrated that their multi-threaded approach (described above as a dynamically balanced approach) can also be coupled with a static work assignment mechanism. Their mechanism uses as its work estimate the number of active cells in a data layer and assigns each computation thread at least two consecutive layers for processing. Their static- and dynamic-balancing mechanisms appear to deliver approximately equivalent performance.

3.3.2. Vector- and pipeline-parallelization

Newman and Tang [120] have shown that MC can be efficiently vector-parallelized—especially for data sets with a moderate number of active cells—by reorganizing

operations in its lattice point marking, intersection interpolation, topology determination, and facetization activities. Their approach processes the data set in several passes, including vertex-by-vertex, edge-by-edge, and cube-by-cube sequential, forward marches. Methods that exploit small-scale vector-parallelism on commodity CPUs for fast MC-based isosurfacing have also been described (for *X*86 [121] and G5 [85] CPUs).

Eldridge et al. [122] have demonstrated that MC can be performed on a pipelined graphics rendering architecture. The architecture can have up to 64 pipes, each with five stages (i.e., for computation of geometry, rasterization, texturing, merger, and display).

3.3.3. Distributed processing

Distributed approaches to MC have also been presented. For example, a framework in which the MC functional modules are viewed as pipeline stages that are distributed among web clients and servers has been described [123,124]. The framework also allows progressive isosurface extraction. A scheme that exploits multi-media pipelines on a remote computer with results delivered over the web and displayed locally via VRML has also been described [125]. Other reports of web-based isosurfacing include [126–128].

3.4. Other isosurfacing methods

Besides the propagation-based approaches mentioned earlier, other alternative methods have been developed, including face-by-face [129] and lattice point-by-lattice point [130] forward-marching methods. Methods using particle attraction [131] and processing on a graph that tracks cell face adjacencies [132] have also been proposed.

4. Mesh considerations I: extending output

Although the standard MC produces an isosurface composed of triangular facets, other representations of the isosurface have been produced by some MC extensions. Table 6 summarizes such extensions, which are discussed in this section of the paper.

4.1. High-degree isosurfaces

One disadvantage of standard MC isosurfaces is that they can exhibit visible faceting artifacts. Data sets with cells of large size (relative to the desired viewing resolution) and data sets with cells of variant sizes, such as finite element data sets, tend to exhibit more severe artifacts. Use of a higher-degree isosurface representation is one means to reduce these artifacts. The existing high-degree extensions of MC use bicubic [48], triangular rational-quadratic Bezier [140], and triangular rational-cubic Bezier [141] spline patches. In the triangular Bezier approaches, trilinear interpolation has been used within each cube to determine patch constraints. Methods that postprocess MC output, rather than modify its steps, also exist but are beyond the scope of this survey.

A key advantage of higher-degree patches is that a more smooth isosurface can be produced. However, parametric polynomial fitting presents a heavier computational burden than triangular mesh fitting. Thus, the tradeoff between smoothness and speed should be considered in any decision to use high-degree surface patches rather than triangular facets.

4.2. Interval volume extractions

One complication that limits the MC is non-crisp structure boundaries. An example is in some medical data where the non-uniform signal response of certain structures makes their boundaries not uniformly distinct. There may not be a fixed isovalue that can be used to accurately extract the boundaries of such structures.

Fujishiro et al. [146,147] have presented a geometric model called the *interval volume* to address the non-crisp boundary issue. The interval volume is the 3D solid space that is associated with a closed interval $[\beta, \gamma]$ of isovalues. Interval volume approaches output polyhedral blocks rather than triangular meshes [147].

Three types of methods have been used to construct the interval volume. The first one, lattice point classification [148,149], requires labeling each lattice point to indicate if its scalar value is within, below, or above the interval $[\beta, \gamma]$. Each cube's labelings are used to find the polyhedral blocks within the cube. The second category of methods uses two special interval volumes based on subintervals $[\beta, max]$ and $[min, \gamma]$, where *max* and *min* represent the maximum and

Table 6
Classification of the papers that extend MC to other types of output or to avoid a visual artifact

Input type	Output type				
	Triangulation	Quads	Spline mesh	Points	Polyhedra
3D Rect. Grid	[28,67,133–138]	[139]	[140,141]	[85,89,142–144], [145][a]	[146–149]
Unstructured grid	NA	NA	[48,49]	NA	NA

[a]The entry marked acts on time-varying 3D data.

minimum scalar value in the data set, respectively [146,147]. Since the basic cube–subinterval volume intersection scenarios exhibit similar topological patterns to those in the standard MC, construction of the polyhedral blocks for each subinterval volume is straightforward. The interval volume for interval $[\beta, \gamma]$ is exactly the intersection of the blocks of the two subinterval volumes. A third approach is to perform isosurfacing of a suitable derived field of a higher dimension and then project the result back to the base dimension [150].

4.3. Pixelized isosurfaces

In some isosurface renderings, multiple facets are projected to a single pixel. This occurrence has led to variations on the MC that save on computation by generating 3D points (instead of triangles) as the output primitives. The dividing cubes [143] is one such variant. The dividing cubes subdivides each cube intersected by the isosurface into subcells such that the subcells project onto a single pixel on the image plane. Other uses of point-based output primitives include [85,89,142,144,145].

4.4. Alternate triangulations

One MC extension that produces an alternative triangulation is Montani et al.'s [136] discretized marching cubes (DMC). Instead of using interpolation to find isosurface–edge intersection points, the DMC uses intersected edge midpoints. Thus, the DMC isosurface resolution is worse but its computation time is better. The use of midpoints in DMC results in some adjacent facets being coplanar. Such facets can be merged, reducing the mesh size and resulting in lower memory consumption and possibly faster mesh rendering. The DMC positional imprecision can be improved by repositioning all intersection points unaffected by the facet merging to the positions described by the linear interpolation scheme of the standard MC [67]. The deformed cubes (DC) [137] also produces a different triangulation. In DC, the triangular mesh has just one triangle vertex per active cell. Each vertex is at the mean of the cell's edge–isosurface intersections. DC is fast, but its mesh can omit small features and contain gaps. Moreover, the mesh usually contains nearly as many facets as the mesh of standard MC.

4.4.1. Overcoming artifacts

A number of methods to overcome MC isosurface artifacts have been presented and are discussed next.

Small structural features are often not well-preserved by the MC. In particular, features of cell size or smaller may not be included in the extracted isosurface [134] due to resolution limitations; in a cell that includes a small feature, all lattice points can have the same marking. Kaneko and Yamamoto [134] have presented an iterative process that can better preserve detail. The first step of each iteration is to extract the isosurface in the usual way. The second step is to adjust the values of lattice points that are adjacent to the isosurface's boundary. The iterations stop when the extracted structure's volume is close to an estimate of the actual volume.

Another approach that can preserve up to one small feature per cell (e.g., when adjacent lattice points have the same marking but the isosurface crosses their connecting edge twice) is Varadhan et al.'s [138] extended dual contouring (EDC). The EDC builds a facetized mesh with mesh vertices that are points within active cells. These internal points are selected using a directed distance measure proposed by Ju et al. [151].

In addition to faceting artifacts, MC renderings can exhibit shading artifacts such as black spots on extracted thin structures [152] and false highlights. For instance, isosurface Gouraud shading, which typically estimates each facet vertex normal vector by linear interpolation of the gradients of the two nearest lattice points, can exhibit such artifacts. The Gouraud shading artifacts result from inappropriate estimates of local orientation. Orientation-related artifacts are especially a concern for applications in which there are meaningful data measures only near the actual isosurface [28]. If such applications determine local orientation based on estimating the gradient from data values in cells adjacent to the facet, the determined orientation may be quite incorrect.

An efficient means to estimate facet orientation using only data within the cell containing the facet has been described by Nielson et al. [28]. Their method can obtain this information directly from the intersection topology look-up table and a supporting table indexed by a bit coding of vertex (or edge) markings. These data tables, which are exhibited in full in [28], can be created offline prior to extraction.

Another type of artifact in MC isosurfaces is aliasing artifacts in isosurface regions that contain sharp edges or corners [135]. One way to reduce such artifacts is to use additional sample points in the vicinity of sharp edges and corners and then generate a triangle fan about each additional point [135]. Sharp edges and corners can be detected using local gradient information [135]. Another way to reduce aliasing artifacts is, instead of extracting the standard isosurface, to extract the dual of a simplified subset of the MC triangle edges, as proposed by Nielson [139]. His approach uses quad patches whose *edges* pass through the MC triangle vertices that lie on the data set's rectilinear lattice. The paper includes a topology table, enabling application directly in data set traversal rather than in postprocessing.

A method to address artifacts when MC is applied to depth maps has also been described [133].

5. Mesh considerations II: challenges

In this section, geometric concerns related to the meshing mechanism of the standard MC are discussed. The

concerns include redundancy (Section 5.1) and correctness and consistency (Section 5.2). The ambiguity problems arising from inconsistency or incorrectness are also described (Section 5.3). Section 5.4 overviews the substantial body of literature concerned with resolving ambiguity in MC. In Section 5.5, computational concerns related to disambiguation are considered.

5.1. Degenerate triangles (redundancy)

When the data set contains lattice points whose values equal the isovalue, some isosurface triangles may be degenerate (i.e., have collinear vertices). For example, when a cube has only one marked lattice point V_1 and V_1's scalar value is equal to the isovalue, the cube will contain one facet whose vertices coincide at V_1. A degenerate triangle in one cube C_d also implies that there will be at least one degenerate triangle in some adjacent cube (unless possibly if C_d is on the data set boundary). The degenerate triangles contribute nothing to the isosurface yet they consume time to build and storage to save. Since degenerate triangles involve redundant computation, the degeneracies in MC have been termed its *redundancy problem* [153].

One way to solve the redundancy problem is to choose an isovalue that is distinct from all scalar values [24]. However, in many cases, not all the scalar values are known. In such cases, an approach such as the detect-and-prevent scheme of Li and Agathoklis [154] can be used. For each cube C, their scheme labels as redundant any intersection point that is coincident with one or two other intersection points in C. In cubes with redundancies, degenerate facets are prevented by toggling the marking of cube lattice points that coincide with redundant intersections.

5.2. Correctness and consistency

In general, end-user understanding of a data set is positively impacted if the extracted isosurface is both *correct* and *topologically consistent*. An isosurface is correct if it accurately matches the behavior of a known function (or some assumed interpolant) that describes the phenomenon sampled in the data set. If each component of an isosurface is continuous (i.e., there are no "holes" in any component [155]), then it is topologically consistent. An isosurface is also considered to be topologically consistent if the isosurface's only holes are on the data set's boundary. It is possible for an isosurface to have a consistent topology but to not be correct.

The standard MC guarantees neither correctness nor topological consistency. For example, Frühauf [156] has noted that the linear interpolation used in MC is unlikely to yield a correct result for computational fluid dynamics (CFD) data since CFD data is usually generated by a non-linear interpolating function. MC can also produce a topologically inconsistent isosurface that contains holes caused by one type of facetization ambiguity. Another type of ambiguity produces topologically consistent but incorrect isosurfaces. Ambiguity analysis and disambiguation methods have been considered extensively. The analyses and approaches are described next.

5.3. Ambiguity in MC

Shortly after the MC was introduced, Dürst [157] discovered that some of the basic intersection topologies actually could be facetized in multiple ways. Although Dürst did not identify the ambiguous topologies, others (e.g., [24,158,159]) have demonstrated that seven topologies are ambiguous—Cases 3, 4, 6, 7, 10, 12, and 13.

Fig. 9(a) illustrates how an inconsistent isosurface can arise from one type of ambiguity. In the figure, two adjacent cubes that share a face (i.e., the face containing coincident vertex V_c) are shown. The isosurface facetization in each cube follows the standard MC facet pattern (the left and right cubes are the Cases 6 and 3, respectively). In the figure, the shared face is intersected differently in each cube. This difference arises because there are multiple possible cube–isosurface intersection patterns in both cubes, and the default intersection patterns are inconsistent on the shared face. The unresolved ambiguity produces a hole in the isosurface. Another way to understand that an error in facetization has occurred is to consider the line segment V_aV_b between unmarked vertex V_a and marked vertex V_b. Segment V_aV_b must, by definition, intersect the isosurface yet the isosurface facets in Fig. 9(a) are not intersected by V_aV_b.

Resolution of inconsistency requires variation from the fundamental facetizations for one or both cubes. For instance, for Fig. 9(a), just the left cube or just the right cube could be facetized differently, as shown in Figs. 9(b) and (c), respectively. The presence of multiple facetizations to resolve inconsistency also demonstrates that an isosurface that has a consistent topology might not be correct (e.g., the representations in Figs. 9(b) and (c) cannot both be correct).

5.3.1. Face ambiguity

The topological inconsistencies of MC arise due to *ambiguous faces*. In Fig. 9(a), the shared face that contains the vertex V_c is an example of such a face. Any face where there are both two diagonally opposite marked lattice points and two diagonally opposite unmarked lattice points is an ambiguous face. We use the term *face ambiguity* to describe inconsistent facetizations involving an ambiguous face shared by two adjacent cubes. Most of the ambiguity correction methods in the literature have addressed only the face ambiguity. Face ambiguity arises in the intersection topologies of Cases 3, 6, 7, 10, 12, and 13.

Although many face ambiguity resolutions have been proposed, one direct solution is to use basic intersection topologies that exploit only rotational symmetry [28,29], since the exploitation of reflection is what injects face

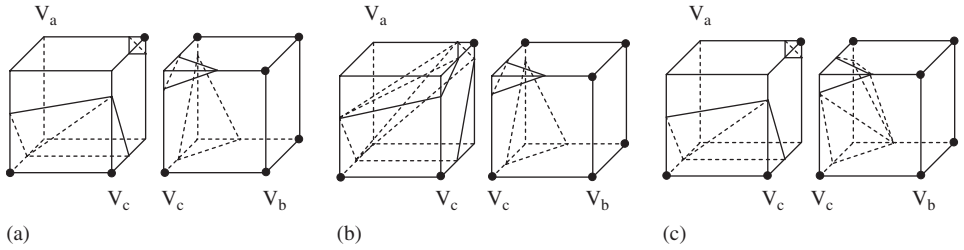

Fig. 9. Illustration of face ambiguity and resolutions: (a) isosurface with holes from an ambiguity in a face shared by two cubes, (b) one alternate facetization that yields a topologically consistent isosurface for the cubes, and (c) another alternate facetization that yields a topologically consistent isosurface for the two cubes.

ambiguity to the standard MC. It is worth noting that Nagae et al. [27] were the first to exploit only rotation, although it is unclear if they realized that exploiting only rotation was the key. Two recent papers [28,29] include an explicit description of a complete, efficient mechanism for employing the rotationally exploitive topologies in MC.

5.3.2. Internal ambiguity

The facetization of a cube that has no ambiguous faces can still have *internal ambiguity* [158,159]. Internal ambiguity does not cause any topological inconsistency but it can yield an incorrect isosurface. Internal ambiguity can arise in Cases 4, 6, 7, 10, 12, and 13. A typical example of the internal ambiguity is Case 4. Fig. 10(a) illustrates its standard facetization, which contains two disjoint facets and may be incorrect. Fig. 10(b) shows a variant facetization that does *not* contain those two facets but rather contains several "linking" facets that form "tubes" [158] or "tunnels" [3,160] that connect two portions of the isosurface. Natarajan [159] was the first to identify the internal ambiguity problem. However, earlier work by Heiden et al. [6] was the first to include a variant facetization for an internally ambiguous case, although that variant was aimed at logically completing a universe of facetizations rather than correcting ambiguity. While MC can have internal ambiguity, Chernyaev [158] has observed that one of its variants (i.e., dividing cubes [143]) is free from internal ambiguity.

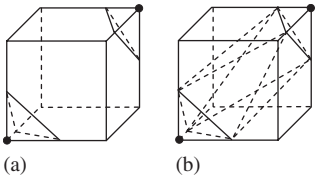

Fig. 10. Illustration of internal ambiguity (two facetizations of the Case 4 marking scenario): (a) disjoint facetizations and (b) a linking facetization.

5.4. Disambiguation approaches

One way to avoid ambiguous facetizations is to resample [24] data at a higher resolution. An isosurface extracted from a data set with a suitable resolution will be topologically consistent and correct. Resampling is quite appropriate when the function underlying the data set is known. However, in most cases, the underlying function is not known. Moreover, resampling may be impractical or impossible. Finally, resampling can result in an isosurface that has an increased number of facets. Other methods to disambiguate MC are discussed next.

5.4.1. Classifying disambiguation approaches

Three types of disambiguation approaches that do not involve resampling have been described by Ning and Bloomenthal [58]: *cell decomposition*, *topology inference*, and *preferred polarity*. In cell decomposition, each cell is partitioned into unambiguous subcells (e.g., tets) to resolve ambiguity. Such approaches produce an increased number of triangles and thus exhibit higher memory and computational requirements. In topology inference, inferencing is used to determine consistent facetizations based on lattice point values. Such approaches can achieve topological consistency and correctness if an appropriate inferencing scheme is used. Preferred polarity approaches achieve consistency by always choosing one of two connection possibilities (polarities) in facetizing the ambiguous faces. This choice is arbitrary, but the same polarity must be chosen for all ambiguous faces. On ambiguous faces, there is an intersection point I_i on each edge E_i. Two pairs of adjacent intersection points are connected (joined) by the isosurface facets. In one connection polarity, these "joins" can be viewed to "separate" the marked lattice points (as shown in Fig. 11(a)), while in the other polarity, the joins can be viewed to separate the unmarked lattice points (as shown in Fig. 11(b)). The former case can be termed the "separated" polarity while the latter case can be termed the "not-separated" polarity [161]. Arbitrary determination of polarity (i.e., without considering the underlying data) can result in a topologically incorrect isosurface [58]. Thus, if correctness is critical, a topology inference method is typically best [58].

An alternate taxonomy has been presented by van Gelder and Wilhelms [24]: the *simple boolean*, *extended boolean*, *simple metric*, and *extended metric* approaches. We will use these classifications in this section. Simple boolean approaches disambiguate a cube based only on its lattice point *markings*; they do not directly consider lattice point scalar *values*. In contrast, the simple metric approaches disambiguate a cube by considering its lattice point values.

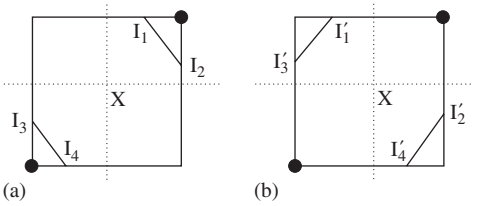

Fig. 11. Ambiguous face connection polarities and criteria used by the AD to determine polarity: (a) $\alpha > B(X)$ (separated) and (b) $\alpha \leqslant B(X)$ (not separated).

Table 7
Papers that consider (or allow) marching cubes disambiguation, categorized by approach

Class	Paper citations
Simple boolean	[6,21,24,27,28,42,44–46,56,58,59,100,106,162–169]
Extended boolean	[22,170–172]
Simple metric	[4,24,29,30,141,158–161,173–180]
Extended metric	[3,24,175,181]

Extended boolean approaches examine inter-cell lattice point markings. Extended metric approaches consider inter-cell values. van Gelder and Wilhelms [24] also empirically analyzed three categories of face disambiguation approaches. (Their paper predates extended boolean approaches.) They found that the approaches that use more information tend to be more correct. For example, they observed that the extended metric approaches tend to produce the most correct isosurfaces.

Disambiguation adds complexity to the MC. Nevertheless, isosurface topological consistency and correctness are necessary in most domains. The remainder of this section describes and summarizes the published disambiguation techniques. The papers concerned with MC disambiguation are listed by category in Table 7.

5.4.2. Simple boolean disambiguations

The simple boolean approaches generally succeed at producing consistent facetizations for ambiguous faces. Two simple boolean approaches already discussed are (1) the convex-hull-based facetization algorithm [21] and (2) exploiting only rotational symmetry, provided that topological base cases are defined consistently, such as in the 23-case look-up table defined in [28]. The latter is also a preferred polarity strategy that uses the separated polarity. Not-separated polarity has also been used (e.g., [27]). The other simple boolean approaches can be dichotomized by the major mechanism—cell decomposition or a modified intersection look-up table—that they use in disambiguation. The approaches that utilize them in MC ambiguity resolution are described next.

5.4.2.1. Cell decomposition. Most of the simple boolean cell decomposition techniques have used the subdivision component of the MT [45,46] to form tetrahedral subdivisions of the data set cubes. Subdivision into octahedra has also been used [42]. As described earlier (Section 3.1.2), after subdivision, the isosurface facets are constructed in the subdivisions. Since processing on tets [58] or octahedra [42] can generate consistent isosurfaces, both MT and marching octahedra can be considered as disambiguation approaches. Work to advance the state of the art and understanding for MT includes [44,56,58,59,100,106,164,165].

Although MT approaches can overcome ambiguity, the different tetrahedral subdivisions of the cube can produce different facetizations [169]. Specifically, there are two schemes to subdivide a cube into five tets, and each scheme results in a differing isosurface facetization. The new marching tetrahedra (NMT) approach [169] can rectify this difference. NMT produces a facetization pattern independent of the subdivision scheme by finding facet vertices using linear interpolation for points on cube edges and using quadratic interpolation for points on cube face diagonals.

5.4.2.2. Modified (extended) look-up tables. Use of a modified look-up table is an attractive strategy because it can be both fast and accurate [24]. Typically, the modified table strategy supplements (extends) the cases in the basic table with subcases for the ambiguous facetization scenarios. Due to the low frequency of occurrence of ambiguous cells, processing time usually is not seriously impacted by subcase determination overhead. For example, one small study suggested that typically about 3% (and at worst 5.6%) of active cells exhibit face ambiguity [24].

Modified table strategies have been used by many (e.g., [162] and in the MC component of the visualization toolkit (vtk) [167]), since the initial use by van Gelder and Wilhelms [24]. The method of Montani et al. [166] also uses a modified table strategy to resolve face ambiguity. Montani et al.'s solution uses one additional facetization pattern for the reflectively symmetric instances of the six scenarios that exhibit face ambiguity.

Heiden et al. [6] have developed an extended look-up table strategy that is based on their finding that for Cases 8–14 of the standard MC, reflection need not be used since the reflective symmetric cases for these can be alternately expressed by exploiting rotation. Use of only rotation also naturally resolves the face ambiguity for Cases 10, 12, and 13. To overcome face ambiguity in Cases 3, 6, and 7, Heiden et al. designed one additional facetization pattern for the reflectively symmetric instance of each scenario. An extra pattern suggested for Case 4 allows address of that case's internal ambiguity. Heiden et al.'s strategy appears to be the first disambiguation approach for standard MC that addresses both topological consistency and internal ambiguity, although it only solves the internal ambiguity problem for Case 4. A later paper by Zhou et al. [168] independently suggested the same analysis and solution as Heiden et al. for the Cases 3, 6, 7, 10, 12, and 13.

5.4.3. Extended boolean disambiguations

The existing extended boolean approaches are: (1) an extended look-up table method [172], (2) a variant on the MT [22], and (3) a coarse facetization scheme [171]. The first method [172] detects the conditions for a hole (i.e., whenever one cube adjacent to an ambiguous face has five or six marked lattice points and the other adjacent cube has two, three, or four marked lattice points) and then adds triangles to span the hole. The method can produce non-manifold surfaces. The second method [22] achieves a more regular MT facetization for tets formed from subdivision. Most MT facetizations are irregular due to the significant differences in sizes of the subdivided tets. A more regular facetization can be achieved using a variant subdivision scheme based on a single tetrahedral shape, with all tetrahedra spanning portions of two cubes [22]. Each tetrahedron has one edge between the centroids of adjacent cubes, four edges from centroids to lattice points, and one edge between lattice points. The third method [171] is motivated by combinatorial topology and produces an isosurface that has facet vertices only at data set lattice points. The approach can be applied such that facet edges are either 18- or 26-connected (i.e., 26-connected edges are either cube edges, face diagonals, or cube diagonals). The approach can be encoded in a look-up table. It considers adjacent cubes in only two intersection scenarios.

Extended boolean algorithms for binary data also exist [170]. These algorithms optimize properties such as the total number of facets via processing on graphs of certain ambiguous cubes and classes of vertices.

5.4.4. Simple metric disambiguations

Most simple metric approaches are either *facial averaging*, *bilinear*, or *trilinear* methods. These approaches, which are discussed next, are also classifiable as *topology inference* methods.

One rule-based simple metric approach [30] also exists. The approach utilizes logical rules to detect ambiguous faces arising from Cases 3, 6, and 7 of the standard MC and generates the facetizations for these cases using the separated connection polarity strategy. It is unclear if the method is suitable for resolving facial ambiguity arising in other cases.

5.4.4.1. Facial averaging. Facial averaging was first employed in the Wyvill et al. [4] propagation-based isosurface extraction. It can also be used to disambiguate MC faces [24] and the interval volume [175]. Facial averaging disambiguates each ambiguous face using the average scalar value of the face's lattice points. Specifically, if the average value equals or exceeds the isovalue, the not-separated connection polarity is chosen, otherwise, the separated polarity is chosen. Facial averaging cannot resolve internal ambiguity.

5.4.4.2. Bilinear approaches. One well-known bilinear approach designed for MC face (or interval volume [175]) disambiguation is the asymptotic decider (AD) of Nielson and Hamann [161]. The AD uses the bilinear interpolant B of each ambiguous face's lattice values. For any isovalue α, the contour curve (i.e., for α) on the bilinear interpolant is hyperbolic in shape, and the hyperbola's branches intersect the ambiguous face. (If the bilinear interpolant were applied to a non-ambiguous face, at most one branch of the contour hyperbola would intersect the face.) The AD determines the value $B(X)$ of the interpolant at the point X, where X is the place that the hyperbola's asymptotes intersect (i.e., X is the saddle point of the interpolant [159]), and compares $B(X)$ to the α to determine the connection polarity for the ambiguous face. As illustrated in Fig. 11(a), when $\alpha > B(X)$, the separated polarity is used. When $\alpha \leqslant B(X)$, the not-separated polarity is used (since X, like the marked lattice points, is interior to or on the boundary of the isosurface), as illustrated in Fig. 11(b). The AD resolves only face (i.e., not internal) ambiguity.

Two schemes similar to AD that correct both types of ambiguity have also been described. One scheme computes bilinear interpolants on two opposite faces of any internally ambiguous cube and then projects the hyperbolic contour curves for the isovalue onto one of the two faces [158]. If the projected curves intersect, a linked facetization is formed for the cube (as illustrated in Fig. 10(b)). Otherwise, the facets inside the cube must be disjoint (as shown in Fig. 10(a)). A variant scheme [158] considers bilinear variation over any plane that is interior to the internally ambiguous cube and parallel to one of its faces. Both schemes can be implemented in a 33-entry look-up table [158]. A full table-based implementation has also been described [176] (and its source code made available online). One fault in the table is that it can result in a non-C^0 facetization since some facets can lie on cube faces; such facets may share an edge with more than one other facet [160].

5.4.4.3. Trilinear approaches. Simple metric trilinear interpolation strategies generally allow both face and internal ambiguities to be resolved. Moreover, if a trilinear interpolant accurately represents the underlying function of the data set, topologically correct isosurfaces can be produced.

When the underlying function of a data set is unknown (which is usually the case), piecewise trilinear interpolation on lattice point values has often been used as an approximation of the distribution inside each cell. As such, piecewise trilinear interpolation has also been used to determine isosurface facetization, especially to resolve ambiguity. The trilinear interpolation assumes that a data set's scalar values vary linearly between lattice points along the x, y, and z directions inside each cell. The topology of the part of the isosurface that intersects each cube can be determined by comparing the given isovalue with the value of the trilinear interpolant at its saddle points. A piecewise trilinear interpolant yields a piecewise cubic surface [18].

One example of a simple metric trilinear interpolation approach is the previously mentioned triangular Bezier patch MC extension [141]. Another approach, proposed by Matveyev [178,179], disambiguates through organizing the intersections of the isosurface with ambiguous faces and with rectangular patches inside cubes. The approach uses graph theoretic means to form the facetization on the intersection points. It can correct both face and internal ambiguity. Matveyev [180] has also described an extended look-up table approach that uses trilinear interpolation to resolve only internal ambiguity. For internally ambiguous cubes, the technique determines whether to form a disjoint or a linking facetization using the number of intersections of the isosurface with each cube diagonal.

The trilinear interpolation strategy of Natarajan [159] has been a basis for several simple metric techniques. The strategy determines the connection pattern in a cube by comparing the isovalue with the value of the trilinear interpolant at seven saddle points—six on faces and one interior to the cube. The face saddles are saddles of the bilinear interpolant of the face's lattice values. The interior (or body) saddle is a saddle point of the trilinear interpolant. Computing its value is a little time-consuming, although factorizing the interpolant can reduce the number of computations [177]. The Natarajan strategy can be viewed as extending the AD [161] by utilizing an extra (i.e., body) saddle point, which allows resolution of internal ambiguity. However, the facetization can be topologically incorrect since there can be more than one body saddle point in Case 13 [160], as well as in other cases [182]. In addition, sometimes the produced mesh is not C^0 [160].

Two techniques have used the Natarajan strategy in look-up table-based disambiguation (i.e., [173,174]). The techniques select among multiple intersection patterns for each ambiguous scenario using the Natarajan strategy. One of the approaches [174] uses an interior point that may not fall on the interpolant, however [160].

Recently, Lopes and Brodlie [160] have presented a partial correction of the incomplete universe of facetizations in [159] and one of its derivatives [174]. In disambiguating the Case 13 topology, the Lopes and Brodlie approach uses two body saddles to allow correctness. The approach is able to determine the triangulation based on 14 generic patterns of intersection, although these patterns are organized by type of boundary polygon rather than by vertex marking pattern. The boundary polygon type of each cube is found by referencing a look-up table indexed through a scheme that considers the cube's standard MC topological case number in conjunction with evaluation of the trilinear interpolant. The approach uses points internal to the cube to ensure that the facetization closely approximates the interpolant. It also produces an isosurface that smoothly changes as the isovalue changes (e.g., if a gradual change of the isovalue causes a "tunnel" to occur, there will be a gradual change from a non-tunneled to a tunneled topology). Compared to standard MC, the Lopes and Brodlie approach executes a little more slowly and produces several times more facets.

The recent method of Nielson [29] also considers within-cube trilinear interpolant behavior. The method resolves facial and internal ambiguity using a three-stage strategy. In its first stage, it uses a 23-case look-up table (i.e., that exploits only rotational symmetry) to determine basic topology. In the second stage, bilinear variation over cube faces is used to resolve facial ambiguity. Lastly, internal ambiguities are resolved by considering the interpolant within each cube. The paper includes a complete catalog of the facetizations that are topologically consistent and correct.

The topologies identified by Nielson [29] are identical to those identified by Lopes and Brodlie [160]. Use of either method can thus produce a mesh that is free of ambiguities. One advantage to the Nielson approach is that its straightforward first phase can be used by itself to produce a triangulated mesh that is topologically consistent (although not necessarily correct; application of the second and third phases produces correctness). Furthermore, the Nielson approach's data structures and case-by-case facetization patterns are presented in detail in the paper. An advantage of the Lopes and Brodlie method is that its facetization more closely follows the interpolant's shape within the cubes. The cost of this higher level of accuracy is a larger number of facets, however. Also, the method does not consider all cases with two body saddles [182].

5.4.5. *Extended metric disambiguations*

Three types of extended metric approaches have been presented: *interpolation*, *gradient consistency heuristics*, and *type merging*. These approaches can also be classified as topology inference methods.

The interpolation approaches have exploited tricubic or trilinear interpolation functions. The tricubic approach [24], which achieves topological consistency, determines the connection polarity for each ambiguous face through fitting a tricubic polynomial to lattice point scalar values in the vicinity of the face. The fittings require moderate computation, which may limit the approach's applicability. The trilinear approach [3] first corrects face ambiguities using the AD [161]. Then, each face saddle point is examined to determine if it is a true saddle point of a trilinear interpolant. This examination considers the lattice point values of the two cells that share the face containing the candidate saddle point. Finally, internal ambiguity is resolved by evaluating a trilinear interpolating function to find body saddle points. These saddle points are used by a process that generates facet vertices interior to cells. The approach well-facetizes "tunnels" (i.e., "tubes") in linking facetizations, at least when trilinear interpolation well-models the variation in the data. However, the trilinear approach can only be applied to data sets in which no adjacent grid vertices have identical values.

Gradient consistency heuristic methods use lattice point gradients (which are usually not known and hence

estimated) to generate topologically consistent isosurfaces. Two gradient consistency heuristic methods have been described: the *center-pointing gradient* and *quadratic fit* approaches [24]. In both, a bivariate quadratic function is fit on each ambiguous face. The function exactly fits lattice point scalar values and interpolates lattice point gradients. The methods compare the value of the bivariate quadratic at a specified position to the isovalue to determine the connection polarity. The specified position is the face center for the center-pointing method and the function saddle point for the quadratic fit method. The quadratic fit method is actually a generalization of the AD [161] since the bilinear interpolant used by the AD is a special case of the bivariate quadratic function. The gradient consistency heuristics have also been used to disambiguate interval volumes [175].

The type merging [181] extends the AD [161] to use information from neighboring cubes. The approach first simplifies the triangulation produced by the standard MC by merging adjacent active cells to form rectangular parallelepipeds. Then, inconsistencies are prevented by applying the AD to the ambiguous faces of the parallelepipeds.

5.5. Ambiguity and speed

When fast isosurfacing is needed, saving time by avoiding disambiguation may be reasonable, especially when there are few ambiguities [175]. One aid to determining ambiguity frequency is the metric of Fujishiro and Takeshima [175]. The metric measures grid point value cooccurrences which appear to be correlated with ambiguity frequency for both MC-based isosurfaces and interval volumes. The metric might also be useful in hierarchical geometric approaches as a means to detect the regions in which disambiguation overhead can be avoided.

6. The mesh's facet count

For large data sets, the MC can generate an isosurface with very many facets. In addition, the triangulation can include undesirable details (e.g., from sampling noise) and many small triangles [183]. As the number of facets increases, frame rendering rates are reduced and memory consumption increases. Approaches to limit this problem's impact, which is especially germane in address of inter-CPU communication latency for multi-CPU computation of MC, include use of more efficient, compressed isosurface representations (e.g., [184]) and simplification of the MC mesh. Discussion of compression and simplification is beyond the scope of this paper.

Approaches that adapt the MC (or its extensions) basic processing in order to support multi-resolution viewing have also been developed. Such approaches can produce a compact, simplified description of the isosurface mesh for low-resolution viewing or a fully detailed description of the mesh when high-resolution viewing is needed. In this section, such strategies are discussed.

6.1. Multi-resolution based approaches

A number of multi-resolution representations have been presented for use with MC, such as the octree and pyramidal approaches that were discussed earlier. Hierarchical approaches to MT have also been demonstrated, some of which are also multi-resolution (e.g., [100,106]). The use of hierarchy can allow faster isosurfacing. Multi-resolution representations can enable rendering less facets than in conventional MT. In the work of Gerstner et al. [103,106,185], a multi-resolution isosurface representation is constructed based on tetrahedral bisection. The technique first decomposes a volume into six tetrahedra and builds a binary tetrahedra hierarchy tree through recursive bisection of each tetrahedron. The MT is then applied on the tree to extract isosurfaces at a desired (or at multiple) resolution(s). Gerstner [99,100] has also utilized this tree for a hierarchical back-to-front sorting of the tetrahedra and isosurface triangles and for a hierarchical computation of data gradients to enable fast extraction and rendering of multiple transparent isosurfaces.

Other hierarchical data structures have also been used to support efficient level-of-detail (LOD) renderings for extensions of the MC. For example, an efficient method for view-dependent MT has been reported by Gregorski et al. [164]. The method employs a hierarchical data structure that well-supports adaptive refinement based on longest-edge bisection. These features are coupled with careful layout of data, which altogether enable efficient processing, especially for LOD operations. Recently, Gregorski et al. [186] have described a way to hasten the method by exploiting desktop graphics hardware.

7. Computing quantities from an isosurface

Methods for determining quantitative information about the isosurface mesh produced by the MC has also been a topic that some have investigated. For example, methods to incorporate computation of isosurface area, volume, etc., have been described (e.g., [187,188]). In addition, a DMC isosurface's volume can be easily found via table look-up [189]. These approaches enable simultaneous isosurface extraction and quantitative information computation.

Histogram-based approaches for determining an isosurface's volume, area, mean gradient, and surface curvature prior to an isosurface extraction have also been described [96]. Nielson [190] has described a quite different approach to determining surface measures, such as curvature. His approach requires use of several variant facetizations for MC which ensure that each portion of the isosurface is a function with a single value. The isosurface can then be locally represented by approximating functions, such as radial basis functions of thin plate splines, for

which surface measures, such as surface curvature, are well-defined. The universe of facetization patterns, when applied with the 23-case rotational symmetry-exploitive MC, forms what Nielson has termed the MC* isosurfacing. Use of MC* involves variant faceting topologies for only four of the 23 configurations shown in Fig. 6.

Quantitative information about isosurfaces is useful to determine properties related to data as well as to evaluate implementations of the MC. For example, isosurface area has been used to determine isovalues of interest [102] and to evaluate MC versus MT [191]. Joseph et al. [192] have described a set of tools that include mechanisms for display of qualitative information (e.g., the relative divergence of two isosurfaces) and of quantitative information (e.g., an isosurface's surface area). The tool set can aid isosurface evaluations, such as analysis of the topological consistency of the isosurface generated by any implementation of the MC.

8. Conclusions and future directions

The MC is probably the most popular method at present for isosurface extraction and rendering. Since the MC was proposed, many researchers have focused on extending the basic approach, resolving its ambiguities, and improving its performance. In this paper, an exhaustive survey of the development and extensions of MC has been presented. The enhancements, applications, and spin-offs reported in the graphics and visualization literature have also been described.

Although some applications, such as those involving "quick-and-dirty" data set investigation, do use the standard MC and simply ignore the possibility of it producing a representation with defects, a variety of resolutions to its ambiguity-induced defects are available. Applications concerned primarily with eliminating face ambiguity should either (1) avoid exploiting reflective symmetry, for example, by using the rotationally symmetric topological base cases explained in detail by Nielson et al. [28] or (2) couple standard MC with any one of the suitable corrective measures, such as the historically popular AD [161], described in Section 5. An application that is interested in the highest fidelity should (1) utilize one of the ambiguity corrections that resolves both internal and face ambiguity or (2) couple an internal ambiguity correction with a basic topology determination that does not exploit reflective symmetry.

Many researchers continue to study MC's behavior, methods to perform it efficiently, and ways to improve the quality of its renderings. For example, study of ambiguity, methods for disambiguation, and methods to achieve a more high-quality facetization—such as how to display interior structures [193] and how noisy edges/corners can be eliminated—is ongoing. Work toward performance improvements, such as (1) to take advantage of the power of multiple processors housed locally or distributed over a wide-area network and (2) to allow efficient out-of-core computation for very large data sets, such as time-varying data, is continuing as well. Expansion of the MC to broader application areas (e.g., to high-dimensional (4 + D) data, especially irregular, unusual, or time-varying 4 + D data) is also a research interest.

Hybrid approaches that combine direct volume rendering (such as ray casting) with IVR (such as MC) also continue to be subjects of ongoing investigation. For example, Kreeger and Kaufman [194] have mixed rendering of polygonal surfaces (extracted by MC or other means) with rendering of solid volume structures. Also, Hauser et al.'s [195] two-level volume rendering approach allows a separate rendering technique to be applied for each object of a volume data set. Development of additional mechanisms and study of those mechanisms' properties are likely in the future.

Many users of MC desire localization of structures or features of interest from a volumetric data set. However, in many data sets, inhomogeneities in the items of interest make isosurfacing of limited utility, especially for tasks requiring analysis of shape. Development of new methods that extend the MC or which postprocess its output to achieve more accurate localization is likely in the future.

The MC is continually being applied to new problems, and such application is likely to produce many more spin-off findings in coming years.

Acknowledgments

We acknowledge and appreciate the National Science Foundation's support under Grants ASC-9702401 and ACI-0222819. We also are grateful for the helpful suggestions of the reviewers and other colleagues, including the ideas that have arisen from discussions with J. Brad Byrd, Suk Seo, and Huijuan Zhang.

References

[1] Jones M, Leu A, Satherley R, Treavett S. Glossary. In: Chen M, Kaufman A, Yagel R, editors. Volume graphics. London: Springer; 2000. p. 395–406.
[2] Lorensen W, Cline H. Marching cubes: a high resolution 3D surface construction algorithm. Computer Graphics 1987;21(4):163–9.
[3] Weber G, Scheuermann G, Hagen H, Hamann B. Exploring scalar fields using critical isovalues. In: Proceedings of visualization '02, Boston, 2002. p. 171–8.
[4] Wyvill G, McPheeters C, Wyvill B. Data structures for soft objects. Visual Computer 1986;2(4):227–34.
[5] Watt A, Watt M. Advanced animation and rendering techniques: theory and practice. New York: Addison-Wesley; 1992.
[6] Heiden W, Goetze T, Brickmann J. Fast generation of molecular surfaces from 3D data fields with an enhanced marching cube algorithm. Journal of Computational Chemistry 1993;14(2):246–50.
[7] Yim P, Vasbinder G, Ho V, Choyke P. Isosurfaces as deformable models for magnetic resonance angiography. IEEE Transactions on Medical Imaging 2003;22(7):875–81.
[8] Lin F, Seah HS, Lee YT. Deformable volumetric model and isosurface: exploring a new approach for surface boundary construction. Computers & Graphics 1996;20(1):33–40.

[9] Ferley E, Cani M-P, Gascuel J-D. Practical volumetric sculpting. Visual Computer 2000;16(8):469–80.
[10] Stein R, Shih A, Baker M, Cerco C, Noel M. Scientific visualization of water quality in the chesapeake bay. In: Proceedings of visualization '00, Salt Lake City, 2000. p. 509–12.
[11] Matsuda H, Cingoski V, Kaneda K, Yamashita H, Takehara J, Tatewaki I. Extraction and visualization of semitransparent isosurfaces for 3D finite element analysis. IEEE Transactions on Magnetics 1999;35(3):1365–74.
[12] Trembilski A. Two methods for cloud visualization from weather simulation data. Visual Computer 2001;17:179–84.
[13] Kim K, Wittenbrink C, Pang A. Data level comparison of surface classification and gradient filters. In: Proceedings of joint international workshop on volume graphics 2001, Stony Brook, New York, 2001. p. 249–63.
[14] Savarese S, Rushmeier H, Rernardini F, Perona P. Shadow carving. In: Proceedings of the eighth international conference on computer vision, Vancouver, 2001. p. (I)190–7.
[15] Frisken S, Perry R, Rockwood A, Jones T. Adaptively sampled distance fields: a general representation of shape for computer graphics. In: Proceedings of SIGGRAPH 2000, New Orleans, 2000. p. 249–54.
[16] Hall M. Defining surfaces from sampled data. In: Glassner A, editor. Graphics gems. New York: Academic Press; 1990. p. 552–7.
[17] Bajaj C, Pascucci V, Schikore D. Accelerated isocontouring of scalar fields. In: Bajaj C, editor. Data visualization techniques. New York: Wiley; 1999. p. 31–47.
[18] Brodlie K, Wood J. Recent advances in volume visualization. Computer Graphics Forum 2001;20(2):125–48.
[19] Livnat Y. Accelerated isosurface extraction approaches. In: Hansen C, Johnson C, editors. The visualization handbook. New York: Elsevier; 2005. p. 39–55.
[20] Banks D, Linton S, Stockmeyer P. Counting cases in substitope algorithms. IEEE Transactions on Visualization and Computer Graphics 2004;10(4):371–84.
[21] Bhaniramka P, Wenger R, Crawfis R. Isosurface construction in any dimension using convex hulls. IEEE Transactions on Visualization and Computer Graphics 2004;10(2):130–41.
[22] Chan S, Purisima E. A new tetrahedral tesselation scheme for isosurface generation. Computers & Graphics 1998;22(1):83–90.
[23] Miguet S, Nicod J-M. A load-balanced parallel implementation of the marching-cube algorithm. In: Proceedings of high performance computing symposium '95, Montreal, 1995. p. 229–39.
[24] van Gelder A, Wilhelms J. Topological considerations in isosurface generation. ACM Transactions on Graphics 1994;13(4): 337–75.
[25] Roberts JC, Hill S. Piecewise linear hypersurfaces using the marching cubes algorithm. In: Proceedings of visual data exploration and analysis VI, San Jose, 1999. p. 170–81.
[26] Banks D, Linton S. Counting cases in marching cubes: Toward a generic algorithm for producing substitopes. In: Proceedings of visualization '03, Seattle, 2003. p. 51–8.
[27] Nagae T, Agui T, Nagahashi H. Surface construction and contour generation from volume data. In: Proceedings of medical imaging '93 conference on image processing (SPIE vol. 1898), Newport Beach, CA, 1993. p. 74–84.
[28] Nielson G, Huang A, Sylvester S. Approximating normals for marching cubes applied to locally supported isosurfaces. In: Proceedings of visualization '02, Boston, 2002. p. 459–66.
[29] Nielson G. On marching cubes. IEEE Transactions on Visualization and Computer Graphics 2003;9(3):283–97.
[30] Delibasis K, Matsopoulos G, Mouravliansky N, Nikita K. A novel and efficient implementation of the marching cubes algorithm. Computerized Medical Imaging and Graphics 2001;25:343–52.
[31] Weber G, Kreylos O, Ligocki T, Shalf J, Hagen H, Hamann B. Extraction of crack-free isosurfaces from adaptive mesh refinement data. In: Proceedings of VisSym'01, Ascona, Switzerland, 2001.
[32] Chiang, Y-J. Out-of-core isosurface extraction of time-varying fields over irregular grids. In: Proceedings of visualization '03, Seattle, 2003. p. 217–24.
[33] Shen H-W. Isosurface extraction in time-varying fields using a temporal hierarchical index tree. In: Proceedings of visualization '98, Research Triangle Park, 1998. p. 159–66.
[34] Sutton P, Hansen C. Isosurface extraction in time-varying fields using a temporal branch-on-need tree. In: Proceedings of visualization '99, San Francisco, 1999. p. 147–53.
[35] Sutton P, Hansen C. Accelerated isosurface extraction in time-varying fields. IEEE Transactions on Visualization and Computer Graphics 2000;6(2):98–107.
[36] Tory M, Röber N, Möller T, Celler A, Atkins M. 4d space-time techniques: a medical imaging case study. In: Proceedings of visualization '01, San Diego, 2001. p. 473–6.
[37] Weigle C, Banks D. Complex-valued contour meshing. In: Proceedings of visualization '96, San Francisco, 1996. p. 173–80.
[38] Weigle C, Banks D. Extracting isovalued features in 4-dimensional scalar fields. In: Proceedings of 1998 symposium on volume visualization, Research Triangle Park, 1998. p. 103–10.
[39] Bhaniramka P, Wenger R, Crawfis R. Isosurfacing in higher dimensions. In: Proceedings of visualization '00, Salt Lake City, 2000. p. 267–73.
[40] G. Ji, H.-W. Shen, R. Wenger, Volume tracking using higher dimensional isosurfacing. In: Proceedings of visualization '03, Seattle, 2003. p. 209–16.
[41] Goldsmith J, Jacobson A. Marching cubes in cylindrical and spherical coordinates. Journal of Graphics Tools 1996;1(1): 21–31.
[42] Carr H, Theussl T, Möller T. Isosurfaces on optimal regular samples. In: Proceedings of VisSym '03, Grenoble, 2003. p. 39–48.
[43] He M, Xiong B, Yu H. Multi-resolution surface reconstruction. In: Proceedings of international conference on image processing '04, vol. 3, Singapore, 2004. p. 1971–4.
[44] Nielson G, Foley TA, Hamann B, Lane D. Visualizing and modeling scattered multivariate data. IEEE Computer Graphics and Applications 1991;11(3):47–55.
[45] Payne B, Toga A. Surface mapping brain function on 3D models. IEEE Computer Graphics and Applications 1990;10(5):33–41.
[46] Shirley P, Tuchman A. A polygonal approximation to direct scalar volume rendering. Computer Graphics 1990;24(5):63–70.
[47] Takahashi T, Mekada Y, Murase H, Yonekura T. High quality isosurface generation from volumetric data and its application to visualization of medical ct data. In: Proceedings of the 17th international conference on pattern recognition, vol. 3, Cambridge, UK, 2004. p. 734–7.
[48] Gallagher R, Nagtegaal J. An efficient 3-d visualization technique for finite element models and other coarse volumes. Computer Graphics 1989;23(3):185–94.
[49] Gallagher R. Computer visualization: graphics techniques for scientific and engineering analysis. Boca Raton, FL: CRC Press; 1995.
[50] He T, Hong L, Varshney A, Wang S. Controlled topology simplification. IEEE Transactions on Visualization and Computer Graphics 1996;2(2):50–4.
[51] S. Schaefer, J. Warren, Dual marching cubes: primal contouring of dual grids. In: Proceedings of Pacific graphics 2004, Seoul, 2004. p. 70–6.
[52] Shekhar R, Fayyad E, Yagel R, Cornhill J. Octree-based decimation of marching cubes surfaces. In: Proceedings of visualization '96, San Francisco, 1996. p. 335–44.
[53] Shu R, Zhou C, Kankanhalli M. Adaptive marching cubes. Visual Computer 1995;11(4):202–17.
[54] Westermann R, Kobbelt L, Ertl T. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. Visual Computer 1999;15(2):100–11.
[55] Carneiro B, Silva C, Kaufman A. Tetra-cubes: an algorithm to generate 3D isosurfaces based upon tetrahedra. In: Proceedings of

IX Brazilian symposium on computer, graphics, image processing and vision (SIBGRAPI '96), 1996. p. 205–10.
[56] Treece G, Prager R, Gee A. Regularised marching tetrahedra: improved iso-surface extraction. Computers Graphics 1999;23: 583–98.
[57] Carr H, Möller T, Snoeyink J. Simplicial subdivisions and sampling artifacts. In: Proceedings of visualization '01, San Diego, 2001. p. 99–106.
[58] Ning P, Bloomenthal J. An evaluation of implicit surface tiler. IEEE Computer Graphics and Applications 1993;13(6):33–4.
[59] Guéziec A, Hummel R. Exploiting triangulated surface extraction using tetrahedral decomposition. IEEE Transactions on Visualization and Computer Graphics 1995;1(4):328–42.
[60] Conkey J, Joy K, Crawford C, Teeter B. Using isosurface methods for visualizing the envelope of a swept trivariate solid. In: Proceedings of Pacific graphics 2000, Hong Kong, 2000. p. 272–80.
[61] Bonnell KS, Duchaineau MA, Schikore DR, Hamann B, Joy K. Material interface reconstruction. IEEE Transactions on Visualization and Computer Graphics 2003;9(4):500–11.
[62] Elvins T. A survey of algorithms for volume visualization. Computer Graphics 1992;26(3):194–201.
[63] Wilhelms J, van Gelder A. Topological considerations in isosurface generation—extended abstract. Computers Graphics 1990;24(5): 79–86.
[64] Criscione P, Montani C, Scateni R, Scopigno R. DiscMC: An interactive system for fast fitting isosurfaces on volume data. In: Proceedings of virtual environments and scientific visualization '96, Monaco, 1996. p. 178–90.
[65] Globus A. Octree optimization. In: Proceedings of SPIE/SPSE symposium on electronic imaging science and technology, San Jose, 1991. p. 2–10.
[66] Livnat Y, Hansen C. View dependent isosurface extraction. In: Proceedings of visualization '98, Research Triangle Park, 1998. p. 175–80.
[67] Montani C, Scateni R, Scopigno R. Decreasing isosurface complexity via discrete fitting. Computer-Aided Geometric Design 2000;17(3):207–32.
[68] Velasco F, Torres JC. Cells octree: a new data structure for volume modeling and visualization. In: Proceedings of vision, modeling, and visualization, Stuttgart, 2001. p. 151–8.
[69] Wilhelms J, van Gelder A. Octrees for faster isosurface generation. ACM Transactions on Graphics 1992;11(3):201–27.
[70] Bordoloi U, Shen H-W. Space efficient fast isosurface extraction for large data sets. In: Proceedings of visualization '03, Seattle, 2003. p. 201–8.
[71] Chiang Y-J, Silva C. I/o optimal isosurface extraction. In: Proceedings of visualization '97, Phoenix, 1997. p. 293–300.
[72] Chiang Y-J, Silva C, Schroeder W. Interactive out-of-core isosurface extraction. In: Proceedings of visualization '98, Research Triangle Park, 1998. p. 167–74.
[73] Cignoni P, Marino P, Montani C, Puppo E, Scopigno R. Speeding up isosurface extraction using interval trees. IEEE Transactions on Visualization and Computer Graphics 1997;3(2):158–70.
[74] Cignoni P, Montani C, Puppo E, Scopigno R. Optimal isosurface extraction from irregular volume data. In: Proceedings of 1996 volume visualization symposium, San Francisco, 1996. p. 31–8.
[75] Gallagher R. Span filter: An optimization scheme for volume visualization of large finite element models. In: Proceedings of visualization '91, San Diego, 1991. p. 68–75.
[76] Giles M, Haimes R. Advanced interactive visualization for cfd. Computing Systems in Engineering 1990;1(1):51–62.
[77] Livnat Y, Shen H-W, Johnson C. A near optimal isosurface extraction algorithm using span space. IEEE Transactions on Visualization and Computer Graphics 1996;2(1):73–84.
[78] Shen H, Hansen C, Livnat Y, Johnson C. Isosurfacing in span space with utmost efficiency (issue). In: Proceedings of visualization '96, San Francisco, 1996. p. 287–94.
[79] Shen H, Johnson C. Sweeping simplices: a fast isosurface extraction algorithm for unstructured grids. In: Proceedings of visualization '95, Atlanta, 1995. p. 143–50.
[80] Bajaj C, Pascucci V, Schikore D. Fast isocontouring for improved interactivity. In: Proceedings of 1996 IEEE symposium on volume visualization, San Francisco, 1996. p. 39–46.
[81] Itoh T, Koyamada K. Automatic isosurface propagation using an extrema graph and sorted boundary cell lists. IEEE Transactions on Visualization and Computer Graphics 1995;1(4):319–27.
[82] Itoh T, Yamaguchi Y, Koyamada K. Fast isosurface generation using the volume thinning algorithm. IEEE Transactions on Visualization and Computer Graphics 2001;7(1):32–46.
[83] van Kreveld M, van Oostrum R, Bajaj C. Contour trees and small seed sets for isosurface traversal. In: Proceedings of the 13th ACM symposium on computational geometry, Nice, 1997. p. 212–9.
[84] Bartz D. Optimizing memory synchronization for the parallel construction of recursive tree hierarchies. In: Proceedings of EG workshop on parallel graphics and visualization, Girona, 2000. p. 53–60.
[85] Livnat Y, Tricoche X. Interactive point-based isosurface extraction. In: Proceedings of visualization '04, Austin, 2004. p. 457–64.
[86] Pesco S, Lindstrom P, Pascucci V, Silva C. Implicit occluders. In: Proceedings of symposium on volume visualization and graphics '04, Austin, 2004. p. 47–54.
[87] Gao J, Shen H-W. Hardware-assisted view-dependent isosurface extraction using spherical partition. In: Proceedings of VisSym '03, Grenoble, 2003. p. 267–76.
[88] Sulatycke P, Ghose K. Multithreaded isosurface rendering on smps using span-space buckets. In: Proceedings of international conference on parallel processing (ICPP '02), Vancouver, 2002. p. 572–80.
[89] von Rymon-Lipinski B, Hanssen N, Jansen T, Ritter L, Keeve E. Efficient point-based isosurface exploration using the span-triangle. In: Proceedings of visualization '04, Austin, 2004. p. 441–8.
[90] Zhang H, Newman T, Zhang X. Case study of multithreaded in-core isosurface extraction algorithms. In: Proceedings of EG symposium on parallel graphics and visualization '04, Grenoble, 2004. p. 83–92.
[91] Mascarenhas A, Isenburg M, Pascucci V, Snoeyink J. Encoding volumetric grids for streaming isosurface extraction. In: Proceedings of the second international symposium on 3D data processing, visualization and transmission (3DPVT '04), 2004. p. 665–72.
[92] Chhugani J, Vishwanath S, Cohen J, Kumar S. Isoslider: a system for interactive exploration of isosurfaces. In: Proceedings of VisSym '03, Grenoble, 2003. p. 259–66.
[93] Sutton P, Hansen C, Shen H-W, Schikore D. A case study of isosurface extraction algorithm performance. In: Data visualization 2000, Proceedings of the joint symposium on visualization, Amsterdam, 2000. p. 259–68.
[94] Saupe D, Toelke J. Optimal memory constrained isosurface extraction. In: Proceedings of vision, modeling, and visualization, Stuttgart, 2001. p. 351–8.
[95] Bajaj C, Pascucci V, Schikore D. The contour spectrum. In: Proceedings of visualization '97, Phoenix, 1997. p. 167–73.
[96] Pekar V, Wiemker R, Hempel D. Fast detection of meaningful isosurfaces for volume data visualization. In: Proceedings of visualization '01, San Diego, 2001. p. 223–30.
[97] Tenginakai S, Lee J, Machiraju R. Salient iso-surface detection with model-independent statistical signatures. In: Proceedings of visualization '01, San Diego, 2001. p. 231–8.
[98] Tenginakai S, Machiraju R. Statistical computation of salient isovalues. In: Proceedings of VisSym '02, Barcelona, 2001. p. 19–24.
[99] Gerstner T. Fast multiresolution extraction of multiple transparent isosurfaces. In: Proceedings of VisSym'01, Ascona, Switzerland, 2001.
[100] Gerstner T. Multiresolution extraction and rendering of transparent isosurfaces. Computers & Graphics 2002;26(2):219–28.

[101] Weber G, Scheuermann G, Hamann B. Detecting critical regions in scalar fields. In: Proceedings of VisSym '03, Grenoble, 2003. p. 85–93.
[102] Carr H, Snoeyink J, van de Panne M. Simplifying flexible isosurfaces using local geometric measures. In: Proceedings of visualization '04, Austin, 2004. p. 497–504.
[103] Gerstner T, Pajarola R. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In: Proceedings of visualization '00, Salt Lake City, 2000. pp. 259–66.
[104] Crockett T. An introduction to parallel rendering. Parallel Computing 1997;23:819–43.
[105] Chiang Y-J, Farias R, Silva C, Wei B. A unified infrastructure for parallel out-of-core isosurface extraction and volume rendering of unstructured grids. In: Proceedings of IEEE parallel and large data visualization and graphics, San Diego, 2001. p. 59–66.
[106] Gerstner T, Rumpf M. Multiresolutional parallel isosurface extraction based on tetrahedral bisection. In: Chen M, Kaufman A, Yagel R, editors. Volume graphics. London: Springer; 2000. p. 267–78.
[107] Sulatycke P, Ghose K. A fast multithreaded out-of-core visualization technique. In: Proceedings of international symposium on parallel and distributed processing (IPPS/SPDP) '99, San Juan, Puerto Rico, 1999. p. 569–75.
[108] Hansen C, Hinker P. Massively parallel isosurface extraction. In: Proceedings of visualization '92, Boston, 1992. pp. 77–83.
[109] Adams P, Dommermuth D. Visualization of steep breaking waves and thin spray sheets around a ship. In: Proceedings of visualization '03, Seattle, 2003. p. 555–9.
[110] Bajaj C, Pascucci V, Thompson D, Zhang X. Parallel accelerated isocontouring for out-of-core visualization. In: Proceedings of 1999 IEEE parallel visualization and graphics symposium, San Francisco, 1999. p. 97–104.
[111] Gao J, Shen H-W. Parallel view dependent isosurface extraction using multi-pass occlusion culling. In: Proceedings of IEEE parallel and large-data visualization and graphics, San Diego, 2001. p. 67–74.
[112] Gao J, Shen H-W, Garcia A. Parallel view dependent isosurface extraction for large scale data visualization. In: Proceedings of the 10th SIAM conference on parallel processing for scientific computing (CD-ROM), Portsmouth, Virginia, 2001.
[113] Kurc T, Gatalyürek Ü, Chang C, Sussman A, Saltz J. Visualization of large data sets with the active data repository. IEEE Computer Graphics and Applications 2001;21(4):24–33.
[114] Mackerras P. A fast parallel marching-cubes implementation on the fujitsu api1000. In Computer Science Technical Report TR-CS-92-10, The Australian National University, 1992.
[115] Zhang H, Newman T. High performance mimd computation for out-of-core volume visualization. In: Proceedings of supercomputing 2001, Denver, 2001 [Poster Presentation Abstracts].
[116] Zhang H, Newman T. Efficient parallel out-of-core isosurface extraction. In: Proceedings of symposium on parallel and large-data visualization and graphics, Seattle, 2003. p. 9–16.
[117] Zhang X, Bajaj C, Blanke W. Scalable isosurface visualization of massive data sets on cots clusters. In: Proceedings of IEEE parallel and large-data visualization and graphics, San Diego, 2001. p. 51–8.
[118] Zhang X, Bajaj C, Ramachandran V. Parallel and out-of-core view-dependent isocontour visualization using random data distribution. In: Proceedings of VisSym '02, Barcelona, 2002. p. 9–18.
[119] Neeman A, Sulatycke P, Ghose K. Fast remote isosurface visualization with chessboarding. In: Proceedings of EG symposium on parallel graphics and visualization '04, Grenoble, 2004. p. 75–82.
[120] Newman T, Tang N. Approaches that exploit vector-parallelism for three rendering and volume visualization techniques. Computers & Graphics 2000;24(5):755–74.
[121] Newman T, Byrd J, Emani P, Narayanan A, Dastmalchi A. High performance simd marching cubes isosurface extraction on commodity computers. Computers & Graphics 2004;28(2):213–33.
[122] Eldridge M, Igehy H, Hanrahan P. Pomegranate: a fully scalable graphics architecture. In: Proceedings of SIGGRAPH 2000, New Orleans, 2000. p. 443–54.
[123] Engel K, Grosso R, Ertl T. Progressive iso-surfaces on the web. In: Proceedings of visualization '98, Research Triangle Park, 1998. p. 37–40.
[124] Engel K, Westermann R, Ertl T. Isosurface extraction techniques for web-based volume visualization. In: Proceedings of visualization '99, San Francisco, 1999. p. 139–46.
[125] Newman T, Narayanan A. Towards a system for web-based medical visualization and simulation. In: Proceedings of 1999 south eastern simulation conference, Huntsville, AL, 1999. p. 143–8.
[126] Bender M, Hagen H, Seck A. A client-side approach towards platform independent molecular visualization over the world wide web. In: Data visualization '99, Proceedings of joint symposium on visualization, Vienna, 1999. p. 167–76.
[127] Clematis A, d'Agostinod D, de Marco W, Gianuzzi V. A web-based isosurface extraction system for heterogeneous clients. In: Proceedings of the 29th Euromicro conference, Belek-Antalya, Turkey, 2003. p. 148–56.
[128] Michaels C, Bailey M. Vizwiz: a java applet for interactive 3D scientific visualization on the web. In: Proceedings of visualization '97, Phoenix, 1997. p. 261–7.
[129] Wallin Å. Constructing isosurfaces from CT data. IEEE Computer Graphics and Applications 1991;11(6):28–33.
[130] Lin C-F, Yang D-L, Chung Y-C. A marching voxels method for surface rendering of volume data. In: Proceedings of computer graphics international, Hong Kong, 2001. p. 306–13.
[131] Crossno P, Angel E. Isosurface extraction using particle systems. In: Proceedings of visualization '97, Phoenix, 1997. p. 495–8.
[132] Lachaud J-O, Montanvert A. Continuous analogs of digital boundaries: a topological approach to iso-surfaces. Graphical Models 2000;62(3):129–64.
[133] Ernst F, van Overveld C, Wilinski P. Efficient generation of 3D models out of depth maps. In: Proceedings of vision, modeling, and visualization, Stuttgart, 2001. p. 203–10.
[134] Kaneko T, Yamamoto Y. Volume-preserving surface reconstruction from volume data. In: Proceedings of IEEE international conference on image processing (ICIP '97), Santa Barbara, 1997. p. 145–8.
[135] Kobbelt LP, Botsch M, Schwanecke U, Seidel H-P. Feature sensitive surface extraction from volume data. In: Proceedings of SIGGRAPH '01, Los Angeles, 2001. p. 57–66.
[136] Montani C, Scateni R, Scopigno R. Discretized marching cubes. In: Proceedings of visualization '94, Washington, 1994. p. 281–7.
[137] Nagae T, Agui T, Nagahashi H. Isosurface construction from volume data. Machine Graphics & Vision 1994;3(1/2):51–60.
[138] Varadhan G, Krishnan S, Kim YJ, Manocha D. Feature-sensitive subdivision and isosurface reconstruction. In: Proceedings of visualization '03, Seattle, 2003. p. 99–106.
[139] Nielson G. Dual marching cubes. In: Proceedings of visualization '04, Austin, 2004. p. 489–96.
[140] Hamann B, Trotts I, Farin G. On approximating contours of the piecewise trilinear interpolant using triangular rational-quadratic bezier patches. IEEE Transactions on Visualization and Computer Graphics 1997;3(3):215–27.
[141] Theisel H. Exact isosurfaces for marching cubes. Computer Graphics Forum 2002;21(1):19–31.
[142] Bærentzen J, Christensen N. Hardware accelerated point rendering of isosurfaces. Journal of WSCG 2003;11(1):41–8.
[143] Cline H, Lorensen W, Ludke S. Two algorithms for the three-dimensional reconstruction of tomograms. Medical Physics 1988;15(3):320–7.
[144] Co C, Hamann B, Joy K. Iso-splatting: A point-based alternative to isosurface visualization. In: Proceedings of Pacific graphics 2003, Canmore, Alberta, 2003. p. 325–34.
[145] Vrolijk B, Botha C, Post F. Fast time-dependent isosurface extraction and rendering. In: Proceedings of the 20th spring

conference on computer graphics, Budmerice, Slovakia, 2004. p. 45–54.
[146] Fujishiro I, Maeda Y, Sato H. Interval volume: a solid fitting technique for volumetric data display and analysis. In: Proceedings of visualization '95, Atlanta, 1995. p. 151–8.
[147] Fujishiro I, Maeda Y, Sato H, Takeshima Y. Volumetric data exploration using interval volume. IEEE Transactions on Visualization and Computer Graphics 1996;2(2):144–55.
[148] Guo B. Interval sets: a volume rendering technique generalizing isosurface extraction. In: Proceedings of visualization '95, Atlanta, 1995. p. 3–10.
[149] Nielson G, Sung J. Interval volume tetrahedrization. In: Proceedings of visualization '97, Phoenix, 1997. p. 221–8.
[150] Bhaniramka P, Zhang C, Xue D, Crawfis R, Wenger R. Volume interval segmentation and rendering. In: Proceedings of symposium on volume visualization and graphics '04, Austin, 2004. p. 55–62.
[151] T. Ju, F. Losasso, S. Schaefer, J. Warren, Dual contouring of hermite data. In: Proceedings of SIGGRAPH '02, San Antonio, 2002. p. 339–46.
[152] Tiede U, Hoehne K, Bomans M, Pommert A, Riemer M, Wiebecke G. Investigation of medical 3D-rendering algorithms. IEEE Computer Graphics and Applications 1990;10(2): 41–53.
[153] Li J, Agathoklis P. A case study of isosurface generation in 3D visualization. In: Proceedings of IEEE Pacific rim conference on communications, computers and signal processing, Victoria, BC, 1993. p. 622–5.
[154] Li J, Agathoklis P. An efficiency enhanced isosurface generation algorithm for volume visualization. Visual Computer 1997;13(9–10):391–400.
[155] Galin E, Akkouche S. Incremental polygonization of implicit surfaces. Graphical Models 2000;62:19–39.
[156] Frühauf T. Raycasting with opaque isosurfaces in nonregularly gridded CFD data. In: EG workshop proceedings of visualization in scientific computing, Chia, Italy, 1995. p. 45–57.
[157] Dürst M. Letters: additional reference to marching cubes. Computer Graphics 1988;22(2):72–3.
[158] Chernyaev E. Marching cubes 33: Construction of topologically correct isosurfaces. In: CERN Report, CN/95-17, 1995 [Available at ⟨http://wwwinfo.cern.ch/asdoc/psdir⟩].
[159] Natarajan BK. On generating topologically consistent isosurfaces from uniform samples. Visual Computer 1994;10(2):52–62.
[160] Lopes A, Brodlie K. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. IEEE Transactions on Visualization and Computer Graphics 2003;9(1):16–29.
[161] Nielson G, Hamann B. The asymptotic decider: resolving the ambiguity in marching cubes. In: Proceedings of visualization '91, San Diego, 1991. p. 83–91.
[162] Bartsch M, Weiland T, Witting M. Generation of 3D isosurfaces by means of the marching cube algorithm. IEEE Transactions on Magnetics 1996;32(3):1469–72.
[163] Bloomenthal J. Polygonization of implicit surfaces. Computer Aided Geometric Design 1988;5(4):341–55.
[164] Gregorski B, Duchaineau M, Lindstrom P, Pascucci V, Joy K. Interactive view-dependent rendering of large isosurfaces. In: Proceedings of visualization '02, Boston, 2002. p. 475–82.
[165] Klein T, Stegmaier S, Ertl T. Hardware-accelerated reconstruction of polygonal isosurface representations on unstructured grids. In: Proceedings of Pacific graphics 2004, Seoul, 2004. p. 186–95.
[166] Montani C, Scateni R, Scopigno R. A modified look-up table for implicit disambiguation of marching cubes. Visual Computer 1994;10(6):353–5.
[167] Schroeder W, Martin K, Lorensen B. The visualization toolkit. Upper Saddle River, NJ: Prentice-Hall; 1998.
[168] Zhou C, Shu R, Kankanhalli M. Handling small features in isosurface generation using marching cubes. Computers Graphics 1994;18(6):845–8.
[169] Zhou Y, Chen W, Tang Z. An elaborate ambiguity detection method for constructing isosurfaces within tetrahedral meshes. Computers Graphics 1995;19(3):355–64.
[170] Andújar C, Brunet P, Chica A, Navazo I, Rossignac J, Vinacua À. Optimizing the topological and combinatorial complexity of isosurfaces. Computer-Aided Design 2005;37(8):847–57.
[171] Kenmochi Y, Kotani K, Imiya A. Marching cubes method with connectivity. In: Proceedings of international conference on image processing '99. vol. 4, Kobe, Japan, 1999. p. 361–5.
[172] Röll S, Haase A, Kienlin M. Fast generation of leakproof surfaces from well-defined objects by a modified marching cubes algorithm. Computer Graphics Forum 1995;14(2):127–38.
[173] Allamandri F, Cignoni P, Montani C, Scopigno R. Adaptively adjusting marching cubes output to fit a trilinear reconstruction filter. In: Proceedings of EG workshop on scientific visualization '98, Wien, 1998. p. 25–34.
[174] Cignoni P, Ganovelli F, Montani C, Scopigno R. Reconstruction of topologically correct and adaptive trilinear isosurfaces. Computers Graphics 2000;24(3):399–418.
[175] Fujishiro I, Takeshima Y. Coherence-sensitive solid fitting. Computers Graphics 2002;26(5):417–27.
[176] Lewiner T, Lopes H, Vieira A, Tavares G. Efficient implementation of marching cubes' cases with topological guarantees. Journal of Graphics Tools 2003;8(2):1–16.
[177] Lin C-C, Ching Y-T. A note on computing the saddle values in isosurface polygonization. Visual Computer 1997;13(7): 342–4.
[178] Matveyev S. Approximation of isosurface in the marching cube: Ambiguity problem. In: Proceedings of visualization '94, Washington, 1994. p. 288–92.
[179] Matveyev S. Resolving the topological ambiguity in approximating the isosurface of scalar function. In: Proceedings of IEEE workshop on visualization and machine vision, Seattle, 1994. p. 18–21.
[180] Matveyev S. Marching cubes: surface complexity measure. In: Proceedings of visual data exploration and analysis VI, San Jose, 1999. p. 220–5.
[181] Oh K-M, Park K. A type-merging algorithm for extracting an isosurface from volumetric data. Visual Computer 1996;12(8): 406–19.
[182] Nielson G. Personal communication July 12, 2005.
[183] Duan Y, Qin H. Extracting boundary surface of arbitrary topology from volumetric data sets. In: Proceedings of joint international workshop on volume graphics 2001, Stony Brook, New York, 2001. p. 235–48.
[184] Lakshmipathy J, Nowinski W, Wernert E. A novel approach to extract triangle strips for iso-surfaces in volumes. In: Proceedings of ACM international conference on VR continuum and its applications in industry '04, Singapore, 2004. p. 239–45.
[185] Gerstner T. Multiresolution visualization and compression of global topographic data. In: Proceedings of the ninth international symposium on spatial data handling, Beijing, 2000. p. 14–27.
[186] Gregorski B, Senecal J, Duchaineau M, Joy K. Adaptive extraction of time-varying isosurfaces. IEEE Transactions on Visualization and Computer Graphics 2004;10(6):683–94.
[187] Hare J, Grosh J, Schmitt C. Volumetric measurements from an isosurface algorithm. In: Proceedings of visual data exploration and analysis VI, San Jose, 1999. p. 206–10.
[188] Oliván J, Bosma M, Smit J. Accurate measurements in volume data. In: Proceedings of medical imaging conference on visualization display, and image-guided procedures, San Diego, 2001. p. 634–41.
[189] Maple C. Geometric desing[sic] and space planning using the marching squares and marching cube algorithms. In: Proceedings of international conference on geometric modeling and graphics (GMAG '03), London, 2003. p. 90–5.

[190] Nielson G. Mc*: Star functions for marching cubes. In: Proceedings of visualization '03, Seattle, 2003. p. 59–66.
[191] Patera J, Skala V. A comparison of fundamental methods for iso-surface extraction. Machine Graphics & Vision 2004;13(4):329–43.
[192] Joseph A, Lodha S, Renteria J, Pang A. Uisurf: Visualizing uncertainty in isosurfaces. In: Proceedings of 4th international conference on computer graphics and imaging, Palm Springs, CA, 1999. p. 184–91.
[193] Fischer J, Bartz D, Straßer W. Illustrative display of hidden iso-surface structures. In: Proceedings of visualization '05, Minneapolis, 2005. p. 663–70.
[194] Kreeger K, Kaufman A. Mixing translucent polygons with volumes. In: Proceedings of visualization '99, San Francisco, 1999. p. 191–8.
[195] Hauser H, Mroz L, Bischi G, Gröller M. Two-level volume rendering. IEEE Transactions on Visualization and Computer Graphics 2001;7(3):242–51.