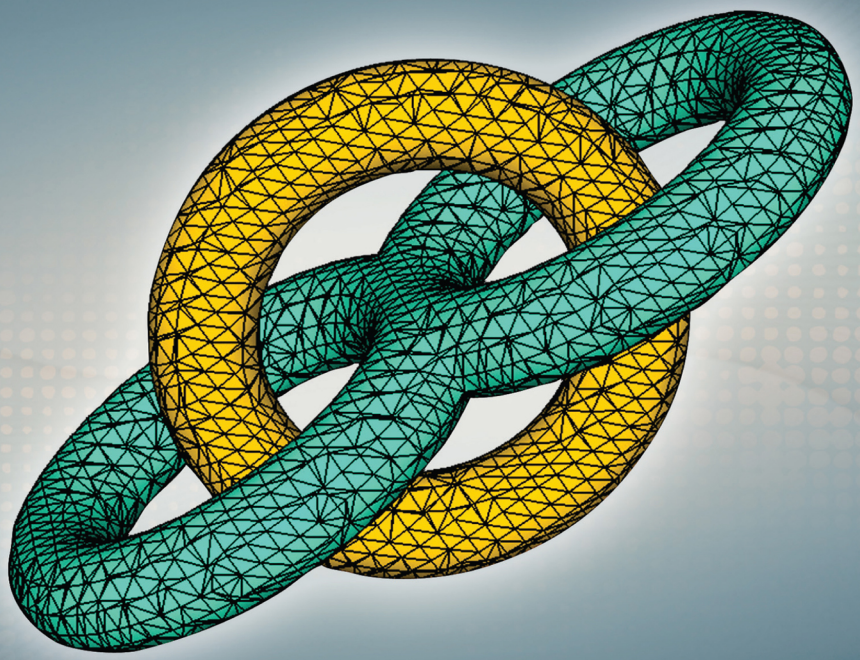
CRC Press
Taylor & Francis Group
AN A K PETERS BOOK

ISOSURFACES

GEOMETRY, TOPOLOGY & ALGORITHMS



REPHAEL WENGER

ISOSURFACES

This page intentionally left blank

ISOSURFACES

GEOMETRY, TOPOLOGY, AND ALGORITHMS

REPHAEL WENGER

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
AN A K PETERS BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20121214

International Standard Book Number-13: 978-1-4665-7102-0 (eBook - PDF)

To my wife, Shifra,
for her love, companionship, and support.

This page intentionally left blank

CONTENTS

PREFACE

Ever since Lorensen and Cline published their 1987 paper on the MARCHING CUBES algorithm, isosurfaces have been a standard technique for visualization of three-dimensional volumetric data. Nevertheless, there is no book specifically devoted to isosurfaces. Part of this is because of the elegance and simplicity of the MARCHING CUBES algorithm, which can easily be described in a few pages. Yet, extensive work has been done since 1987 on extensions and variations of the MARCHING CUBES algorithm, on other algorithms for isosurface construction, on isosurface simplification, and on isosurface topology.

This book is my attempt to give a clear presentation of the basic algorithms for isosurface construction. It is also an attempt at a more rigorous, mathematical perspective for some of the algorithms and results. My targeted audience is designers of visualization software who would like an organized overview of the various algorithms associated with isosurfaces; graduate students pursuing research in visualization who need a solid introduction to research in the areas; and visualization researchers for whom this can serve as a reference for the vast amount of literature on isosurfaces.

The mathematical proofs in this book are more rigorous and challenging than one might see in a typical graphics or visualization text. Despite the many readers who will skip the proofs, I have included them because they are "guarantors" of the correctness of the claims about the various algorithms. Starting with the paper by Lorensen and Cline, numerous papers on isosurfaces contain erroneous, obscure, or unsubstantiated claims. The proofs in this book are an attempt to remedy this deficiency. I have tried to place the proofs in separate sections so that readers who wish to avoid them can easily do so.

Of course, it is possible (and probable) that some of the claims and/or proofs in this book are incorrect. Providing the proofs will hopefully help others uncover and correct any erroneous claims.

Because some readers will be interested only in a subset of the topics in this book, I have tried to make the chapters as self-contained as possible. Unfortunately, this resulted in some redundancy in the text, for which I apologize.

Everyone should read Chapters 1 and 2, the introduction and the MARCHING CUBES algorithm. Chapters 5, 6, and 7 on isosurface patch construction, four-

dimensional isosurfaces, and interval volumes are related and should be read in order. Chapter 9 on multiresolution tetrahedral meshes should be read before Section 10.1 on multiresolution convex polyhedral meshes. Chapter 3 on dual contouring should be read before Section 10.2 on multiresolution surface nets. The other chapters are relatively independent and can be read independently.

ACKNOWLEDGMENTS

I am indebted to many people for their support and assistance. First and foremost is my colleague Tamal Dey, who continually challenged me to apply rigorous methods to geometric modeling. He is a source of inspiration and a role model for research excellence. Roger Crawfis initiated my interest in isosurfaces during a graduate seminar on isosurfaces. That seminar led to a joint paper on generalizing the Marching Cubes algorithm to four dimensions and my ongoing interest and research in isosurfaces. Colleagues and collaborators, Han-Wei Shen and Yusu Wang, were also sources of encouragement and inspiration.

A special thanks to Josh Levine who carefully reviewed this book and suggested numerous corrections and improvements. Thanks also to my former and current students, Ramakrishnan Khaziyur-Mannar, Marc Khoury, and Arindam Bhattacharya. Thanks to Hamish Carr at University of Leeds and Carlos Scheidegger from ATT Labs for numerous conversations about isosurfaces and scalar data sets. Thanks also for support from the National Science Foundation.

Many of the images in this text were produced from data sets compiled by Michael Meissner at www.volvis.org and the data sets at the Volume Library, www.stereofx.org, compiled by Stefan Roettger. These data sets are an invaluable resource for research in volume graphics.

Finally, thanks to my wife, Shifra. Without her encouragement and support, I would never have completed this book.

This page intentionally left blank

CHAPTER 1

INTRODUCTION

1.1 What Are Isosurfaces?

A scalar field is a function ϕ which assigns a scalar value (a real number) to each point in $\mathbb{R}^d$. The value d is known as the dimension of the scalar field. Examples of three-dimensional scalar fields are densities, pressures, or temperatures associated with points in $\mathbb{R}^3$. If these values change with time, then the addition of time as a fourth dimension gives a four-dimensional scalar field.

Given a scalar field $\phi : \mathbb{R}^d \to \mathbb{R}$ and a constant $\sigma \in \mathbb{R}$, the set $\{x : \phi(x) = \sigma\}$ is called a level set[1] of ϕ. We use the notation $\phi^{-1}(\sigma)$ to represent the level set $\{x : \phi(x) = \sigma\}$. If ϕ is a continuous function, then the level set $\phi^{-1}(\sigma)$ separates $\mathbb{R}^d$ into two sets of points, those with scalar value above σ and those with scalar value below σ.

In two dimensions ($d = 2$), level sets are called isocontours or contour lines. Contour lines in topographic maps are a familiar example of isocontours. Each contour line on a topographic map represents a specific elevation. Walking along the contour line means walking along a level path that does not change elevation. Crossing contour lines means climbing up or down and changing elevations.

In three dimensions ($d = 3$), level sets are also called implicit surfaces or isosurfaces. In computer graphics, the term *implicit surface* is generally used to refer to surfaces defined by explicitly providing a function ϕ. Problems include rendering such a surface, converting the implicit representation to a parameterized one, and computing intersections of implicit surfaces.

[1] This mathematical formulation of level sets should not be confused with the level set method for segmentation. The level set method defines a continuous, smooth function g based on the input data and then uses the level sets from this function to segment the data.

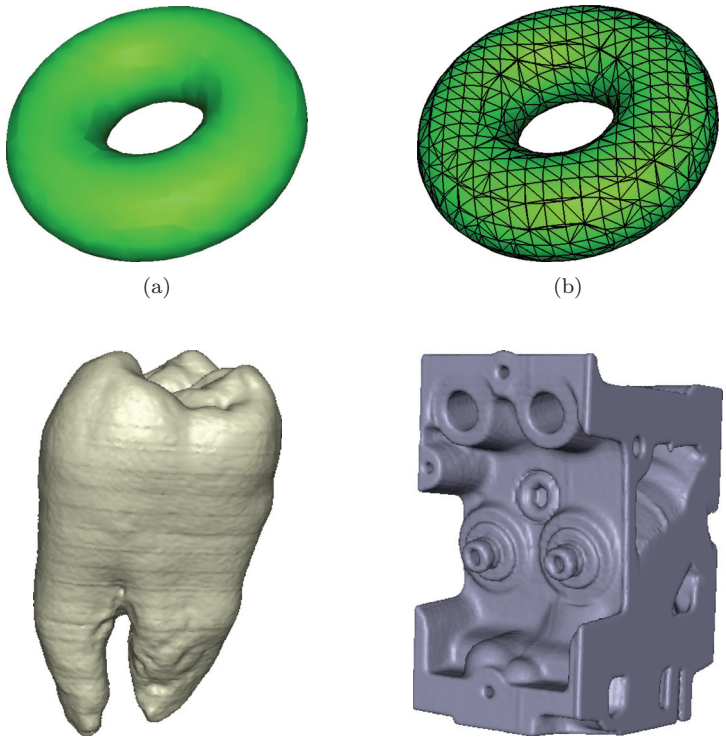


Figure 1.1. (a) Isosurface (isovalue 3) forming a torus. Scalar data set is a $20 \times 20 \times 20$ regular grid with origin $(0, 0, 0)$ measuring the distance to a circle with radius 6 centered at $(9.5, 9.5, 9.5)$. (b) Torus isosurface edges. (c) Isosurface (isovalue 600) of a micro CT scan of a tooth using a GE Industrial Micro CT scanner. Data set created by GE Aircraft Engines. (d) Isosurface (isovalue 80) of CT scan of an engine block. Data set created by General Electric.

Isosurface is the term for level sets used in volume visualization. Generally, it refers to a surface constructed from a finite set of input points, each associated with a scalar value. (See Figure 1.1.) This set of input points is a sampling of some continuous function ϕ and the isosurface is an approximation of the level set of ϕ. Of course, numerous functions take on the same value at a finite set of sample points, so the function ϕ and the isosurface are not uniquely defined. In addition, sample data often contains noise and so is not even a precise representation of ϕ at the sample points. Finally, the very idea that the input data represents a sampling of some continuous function ϕ is itself a modeling assumption and may be misleading.

Unfortunately, the term *isosurface* is sometimes used to represent the level set of a function ϕ and at other times is used to represent an approximation to a level set. In this book, we will always use level set to refer to the mathematically defined set $\phi^{-1}(\sigma)$. We use the term isosurface to refer to an approximation to a level set $\phi^{-1}(\sigma)$ where function ϕ is represented by a finite set of sample points. The value σ is called the isovalue of the isosurface.

1.2 Applications of Isosurfaces

Two well-established procedures in medical imaging produce extensive scalar field data. Computerized tomography (CT) scanners send beams of radiation through a person and measure the amount of radiation that arrives at various detectors. The radiation measurements are processed to produce a (radiation) density at various sample points within the person. Magnetic resonance imaging (MRI) scanners measure changes in a magnetic field caused by excited hydrogen nuclei in water. Mathematical transformations map these measurements to water density values at sample points within the person. The CT and MRI density measurements implicitly represent a scalar density field on the scanned person with each point associated with a density. Since CT and MRI scans are measuring different material properties, they have different relative strengths and weaknesses. CT scans excel at imaging solid organs while MRI scans are better at imaging subtle differences in soft tissue.

The output of a CT or MRI scan is simply a set of values associated with sample points, usually on a regular grid. Regions within this data represent individual objects such as skin, muscle, or bones or pathologies such as tumors, hemorrhages, or bowel obstructions. There are two approaches to visualizing objects within this data. One, called direct volume rendering, is to cast imaginary rays from a specified eye location through the data and integrate a color along the rays based on the density values. A transfer function determines how the color is constructed from the density values. By varying the transfer function, the user can view or highlight different objects within the data. Direct volume rendering can produce excellent images, but it is computationally expensive and produces only a visual image of a specific view of the data. In addition, the transfer function is difficult to set and adjust.

The other approach to visualizing data is to produce surfaces representing the boundaries of objects within the data. This approach is called surface reconstruction. Once such surfaces are produced, they can be rendered from any viewpoint using standard computer graphics techniques. Moreover, the surfaces model the object boundaries and can be used to measure object volume and surface area. The most direct way to produce surfaces from volumetric data is to construct an isosurface that approximates the level set of a scalar field implicitly represented by the data.

Computational fluid dynamics also produces extensive scalar field data. In computational fluid dynamics, the flow space is partitioned into small elements (polyhedra). Each element has a flow density that is derived by solving a set of finite difference equations. The flow density of an element can be thought of as the density of some point within the element, perhaps the barycenter. Usually this density varies with time. The objects of interest in fluid flow are high or low pressure regions, perhaps representing shock waves or turbulence. Again, either direct volume rendering or surface reconstruction can be used to visualize such regions at a fixed time.

1.3 Isosurface Properties

As previously mentioned, we use the term *isosurface* to refer to an approximation of a level set. There are infinitely many approximations to a level set. What properties are required or desired in such an approximation?

One obvious property is that the isosurface should be a surface. However, this is not as simple as it seems. For example, the level set is not necessarily a surface: the level set of the constant function, $\phi(x, y, z) = \sigma$, is all of $\mathbb{R}^3$ for isovalue σ and the empty set for all other isovalues. If $g : \mathbb{R}^3 \to \mathbb{R}$ is the distance from (x, y, z) to the origin, then the level set of isovalue 0 is a point.

Another problem is what exactly is meant by surface. Consider the union of two unit spheres in $\mathbb{R}^3$, one lying above and one below the $x-y$ plane, such that the two spheres touch at the origin. Is the union of these two spheres a surface? The union of two spheres tangent at the origin separates points inside the spheres from points outside the spheres. On the other hand, in the neighborhood of the origin this set of points looks like two surfaces glued together at a single point. In technical terms, the union of two spheres is not a 2-manifold. Should the isosurface be a 2-manifold?

An isosurface is an approximation to a level set of a continuous scalar field ϕ. However, only a finite sampling P of ϕ is given. There are numerous scalar fields ϕ with drastically different geometry and topology that have the same scalar values on P. These different scalar fields can give rise to very different isosurfaces. How do we choose among such isosurfaces?

One assumption we will make is that function ϕ is continuous. Under this assumption, it is possible to at least identify some line segments that are intersected by the level set $\phi^{-1}(\sigma)$.

Let p and p' be points in P where p has scalar values above $\sigma \in \mathbb{R}$ and p' has scalar values below σ. For any continuous field ϕ, the level set $\phi^{-1}(\sigma)$ intersects line segment (p, p'). Thus, any isosurface approximation of such a level set should intersect line segment (p, p').

On the other hand, if p and p' both have scalar values above or both have scalar values below σ, then the level set $\phi^{-1}(\sigma)$ may or may not intersect line segment (p, p'). In general, the isosurface should not intersect such an edge.

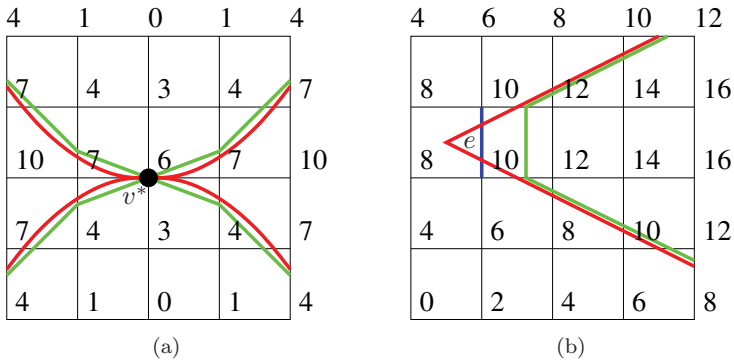


Figure 1.2. (a) Scalar grid sampling the function $\phi_a(x, y) = (x - 2)^2 - 3|y - 2| + 6$, the red level set $\phi_a^{-1}(6)$ and the green isocontour with isovalue 6. Four branches of the level set and four edges of the isocontour meet at the grid center, v^*. (b) Scalar grid sampling the function $\phi_b(x, y) = 2x - |4y - 10| + 10$, the red level set $\phi_b^{-1}(11)$ and the green isocontour with isovalue 11. The red level set intersects the blue grid edge e twice.

The boundary of many objects, particularly manufactured ones, is often piecewise smooth but with sharp edges or corners connecting the pieces. An isosurface representing such a boundary should not smooth over such edges or corners.

We summarize some of the desirable properties of an isosurface:

1. It separates sample points with scalar value above isovalue from scalar points with value below isovalue.

2. It does not intersect a grid edge more than once.

3. It does not intersect grid edges with both endpoint scalar values above or both endpoint scalar values below the isovalue.

4. It is a manifold.

5. It represents sharp edges and corners.

Not all of these properties are always desirable. For instance, Figure 1.2(a) displays a scalar grid sampling the function $\phi_a(x, y) = (x - 2)^2 - 3|y - 2| + 6$. The vertex v^* at the grid center has scalar value 6, so any isocontour with isovalue 6 should pass through this vertex. Each of the four squares surrounding v^* contains a grid edge with scalar values of 4 and 7 so the isocontour passes through each such grid edge. The result is that four isocontour edges meet at v^*, and the isocontour is not a manifold. However, the isocontour does faithfully represent the topology of $\phi_a^{-1}(6)$ which has four curves meeting at v^*.

Some of the properties listed above are mutually exclusive. Figure 1.2(b) displays a scalar grid sampling the function $\phi_b(x, y) = 2x - |4y - 10| + 10$. The red level set $\phi_b^{-1}(11)$ has a sharp corner at (0.5,2.5) and intersects the blue grid edge e twice. The green isocontour does not properly represent the sharp corner at (0.5,2.5) and does not intersect grid edge e. Any isocontour that reproduces the sharp corner, satisfying Property 5, would intersect grid edge e twice, violating Properties 2 and 3.

1.4 Isosurface Construction

There are four basic approaches to isosurface construction. The first and earliest approach is to partition volumetric data into two-dimensional (2D) slices, construct isocontours in each slice, and then "stitch" together the slices using triangles. This approach mimics the way early radiologists used CT and MRI data by examining slices of the data. The difficulty is in the stitching, which is both time-consuming and error-prone. This approach has been superseded by volumetric methods, which construct the isosurface directly in 3-space.

The second approach is to partition space into cubes and associate each cube with a scalar value. The isosurface is the boundary of all cubes with scalar values below a given value. This approach was motivated by pixel graphics, which represents images as a collection of square pixels. The obvious drawback is that the boundary of a set of cubes is extremely nonsmooth, with faces meeting at ninety-degree angles. In visualization, this problem can be mitigated by rendering the surface using "phony" surface normals constructed from the original data. Alternatively, smoothing techniques can be applied to the choppy surface but with potential loss of some detail.

The third and most popular approach is the MARCHING CUBES algorithm and its variants introduced by Lorensen and Cline [Lorensen and Cline, 1987a] in 1987. The MARCHING CUBES algorithm partitions the volume into cubes and then independently constructs surface patches within each cube. Each patch is a small triangulated surface with a boundary on the cube. Based on a comparison of the scalar values of the cube corners and the isovalue, a cube is classified into one of 256 cases. The surface patches are constructed using a precomputed table based on these 256 cases.

The original MARCHING CUBES algorithm sometimes created cube patches that did not properly meet the patches of adjacent cubes. A number of solutions were proposed, the simplest being a change to the precomputed table of 256 cases.

Variants of the MARCHING CUBES algorithm include using tetrahedra instead of cubes and extending the algorithm to higher dimensions.

The last and most recent approach is called dual contouring. The volume is partitioned into cubes and each cube is replaced by a single point. Points in

adjacent cubes are connected to form a surface using quadrilaterals that are the dual of cube edges. Dual contouring has the nice property of producing surfaces that are tiled by quadrilaterals, not triangles. It can also be easily used with multiresolution techniques where the volume partitioning may not be uniform. On the other hand, the surfaces produced by dual contouring are usually not manifolds.

1.5 Limitations of Isosurfaces

Using isosurfaces to model object boundaries from volumetric data has some significant advantages. Isosurfaces encode basic, simple structures of the scalar field sampled in the input data. They are easy to define and understand. They correspond to a formal mathematical object, the level set of a scalar field, and so lend themselves to rigorous mathematical analysis. They can be constructed in time proportional to the size of the input data (linear time).

Unfortunately, isosurfaces have some significant deficiencies and limitations as models for object boundaries. These deficiencies are caused by problems of sampling and noise and by the lack of any global criterion in the isosurface definition. We list some below:

1. undersampling of the spatial domain,
2. high-frequency noise,
3. low-frequency noise,
4. overspecification of the scalar values,
5. lack of smoothness criterion,
6. choice of isosurface,
7. lack of global information,
8. lack of a priori information.

Undersampling and high-frequency noise generate adjacent samples with large variations in scalar value. These scalar variations create surfaces with complicated geometric and topological features that are not representative of the object. In regions where scalar values are constant or near-constant, using scalar values with precision beyond the range of the scanner creates isosurfaces which wind arbitrarily through the regions. Without any smoothness criterion, isosurfaces have no restrictions on their susceptibility to undersampling and noise, even though most objects are best represented by some smooth or piecewise smooth boundaries.

Applying smoothing and noise reduction filters to the raw data helps mitigate some of the problems described above but at the expense of losing some of the fine isosurface features and nonsmooth features that may be present in the data. On the other hand, one of the benefits of isosurfaces is their faithfulness to the data, including all the irregularities and noise in the data. The trade-off between smooth filtering versus exact data representation is data- and application-dependent and is best left to the individual researcher or clinician.

Low-frequency noise produces shifts in scalar values in different regions of the data. The boundary of the object or objects of interest may have different scalar values in different regions of the data. One isosurface will capture the objects in one region while a different isosurface with a different isovalue will bound the objects in the other region. Between the two regions, an isosurface may give object fragments, representing portions of the object. Normalizing the data across regions by adjusting scalar values may help, but it creates the danger of introducing normalization errors.

Isosurfaces depend upon a single parameter, the isovalue of the points on the isosurface. Choosing this parameter is itself a challenging task. Both visualization and data analysis tools exist to help in finding interesting or relevant isovalues.

Isosurfaces are intrinsically local with no global criteria about their shape or structure. In almost all applications such global criteria do exist and are known to researchers or clinicians. On the other hand, because isosurfaces make no application or data-specific assumptions, they are versatile structures that can be used in almost any geometric application. They are a basic tool for anyone visualizing or modeling data but only as the building blocks for more sophisticated data-specific tools.

1.6 Multivalued Functions and Vector Fields

Many applications produce more than a single scalar value at each point. The simplest example is color images that have an RGB (red, green, blue) value associated with each pixel. In fluid flow simulation, both a pressure and temperature could be associated with sample points in the flow. Combinations of scans from different instruments, such as a CT scan and an MRI scan of the same individual, can produce a radiation density and a water density at each sample point.

Visualizing and modeling multivalued data is much more difficult than analyzing scalar fields. Sometimes multiple values are combined into a single scalar value at each point producing a single scalar field. Isosurfaces can then be used to visualize and model objects in that scalar field. The resulting surface is highly sensitive to the function used to create the scalar field from the multivalued functions.

Vector fields are multivalued functions that map $\mathbb{R}^d$ to $\mathbb{R}^d$. In fluid flow simulation, they can represent direction and speed of the flow. Critical points in a vector field are points that are assigned the zero vector, $(0, 0, \ldots, 0)$. Visualization and modeling of vector fields usually relies upon identification of critical points and representation of the flow between critical points.

Various transformations can be used to transform a vector field into a scalar one—for instance, by replacing each vector by its length. Such transformations are usually too crude to extract all but the most rudimentary information.

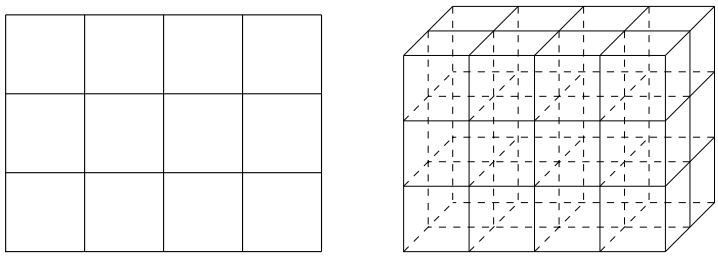
1.7 Definitions and Basic Techniques

Before discussing isosurface construction, we need to review some basic definitions and techniques that are used throughout this book.

1.7.1 Definitions

Regular scalar grid. Isosurface construction algorithms take as input a sample set of points. This sample set is often represented by a regular grid.

In two dimensions, a regular grid is a partition of a large rectangle into small congruent rectangles. More generally, a regular grid in $\mathbb{R}^d$ is a partition of a large hyperrectangle into small congruent hyperrectangles. (See Figure 1.3.) The vertices and edges of the regular grid are the vertices and edges of the small hyperrectangles. A typical example of a regular grid is the partition of the region $[0, m_1] \times [0, m_2] \times [0, m_3]$ into $m_1 \times m_2 \times m_3$ cubes. Note that along each axis d this regular grid has m_d edges and $(m_d + 1)$ vertices. The grid has $(m_1 + 1) \times (m_2 + 1) \times (m_3 + 1)$ vertices.



(a) A 2D regular grid. (b) A 3D regular grid.

Figure 1.3. (a) A 2D regular grid with vertex dimensions 5×4 and cube dimensions 4×3. (b) A 3D regular grid with vertex dimensions $5 \times 4 \times 3$ and cube dimensions $4 \times 3 \times 2$.

The vertex dimensions of a regular grid is the number of vertices along each axis. A regular grid of cubes with vertex dimensions $n_1 \times n_2 \times n_3$ has n_d vertices along each axis, $n_1 \times n_2 \times n_3$ vertices, and $(n_1 - 1) \times (n_2 - 1) \times (n_3 - 1)$ cubes.

The cube dimensions of a regular grid is the number of edges along each axis. A regular grid of cubes with cube dimensions $m_1 \times m_2 \times m_3$ has m_d edges along each axis, $(m_1 + 1) \times (m_2 + 1) \times (m_3 + 1)$ vertices, and $m_1 \times m_2 \times m_3$ cubes. A regular grid with cube dimensions $m_1 \times m_2 \times m_3$ has vertex dimensions $(m_1 + 1) \times (m_2 + 1) \times (m_3 + 1)$.

Unless otherwise noted, the dimensions of a grid refers to its vertex dimensions. Thus, an $n_1 \times n_2 \times n_3$ regular grid has vertex dimensions $n_1 \times n_2 \times n_3$ and cube dimensions $(n_1 - 1) \times (n_2 - 1) \times (n_3 - 1)$.

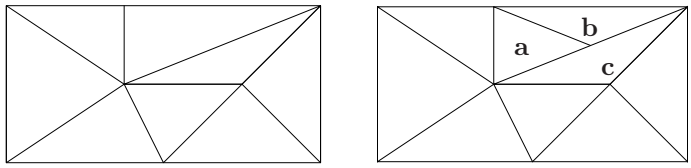
A regular scalar grid is a regular grid where each grid vertex v_i is associated with a scalar value $s_i \in \mathbb{R}$. A simple example is a grayscale image—for instance, a black-and-white picture.[2] The sample points are the pixel centers. The scalar value at each point is the grayscale value of the pixel containing the point.

Triangulation. Isosurfaces are often triangulations, sets of triangles or simplices with appropriate intersection conditions.

Definition 1.1. A triangulation τ is a set of simplices such that for every pair of simplices $\mathbf{t}, \mathbf{t}' \in \tau$, the intersection $\mathbf{t} \cap \mathbf{t}'$ is either empty or a face of each simplex.

For instance, if triangulation τ is a set of triangles, then the intersection $\mathbf{t} \cap \mathbf{t}'$ is either empty, a common vertex of $\mathbf{t}$ and $\mathbf{t}'$, or a common edge of $\mathbf{t}$ and $\mathbf{t}'$. (See Figure 1.4.)

Mathematics texts usually add a formal requirement that if simplex $\mathbf{t}$ is in τ, then every face of $\mathbf{t}$ is in τ. See Appendix B.4 for further discussion and definitions.



(a) (b)

Figure 1.4. (a) A triangulation of a rectangle. (b) A partition of a rectangle into triangles, which is not a triangulation. The intersection of triangles **a** and **c** is a line segment that is not an edge of **c**. The intersection of triangles **b** and **c** is also not an edge of **c**.

[2] The term *black-and-white* is a bit misleading since black-and-white pictures generally contain all different shades of gray.

The notation $|\tau|$ represents the set of all points in all triangles of τ, i.e., $|\tau| = \bigcup_{\mathbf{t}\in\tau} \mathbf{t}$.

Definition 1.2. A set $\mathbb{X} \subseteq \mathbb{R}^d$ is piecewise linear if $\mathbb{X}$ equals $|\tau|$ for some triangulation τ.

Convex Polyhedral Mesh. In many instances, a scalar field is represented not by a regular scalar grid but by a mesh composed of triangles or convex polyhedra.

Definition 1.3. A convex polyhedral mesh Γ is a set of convex polyhedra in $\mathbb{R}^3$ such that for every pair of convex polyhedra $\mathbf{c}, \mathbf{c}' \in \Gamma$, the intersection $\mathbf{c} \cap \mathbf{c}'$ is either empty or a face of each convex polyhedron.

Mathematics texts usually add a formal requirement that if convex polyhedron $\mathbf{c}$ is in Γ, then every face of $\mathbf{c}$ is in Γ.

The notation $|\Gamma|$ represents the set of all points in all elements of Γ, i.e., $|\Gamma| = \bigcup_{\mathbf{c}\in\Gamma} \mathbf{c}$.

A tetrahedral mesh is a convex polyhedral mesh where every mesh element is a tetrahedron. A scalar mesh is a mesh where each mesh vertex v_i is associated with a scalar value $s_i \in \mathbb{R}$.

The generalization of a convex polyhedral mesh to $\mathbb{R}^d$ is called a convex polytopal mesh. The definition is given in Appendix B.5. A convex polytopal mesh where every mesh element is a simplex is a triangulation. It is also sometimes called a simplicial mesh.

Orientation. Let L be a line segment L with vertices $\{v_0, v_1\}$. The orientation of L is an ordering of the vertices of L, either (v_0, v_1) or (v_1, v_0).

Let $\mathbf{t}$ be a triangle with vertices $\{v_0, v_1, v_2\}$. The orientation of $\mathbf{t}$ is a cyclic order of the vertices of $\mathbf{t}$. (See Figure 1.5(a).) There are two possible cyclic

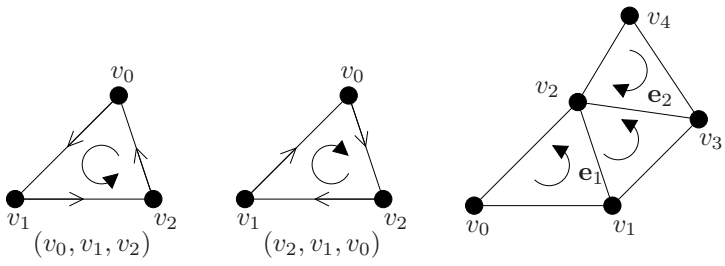


Figure 1.5. (a) Triangle orientation (v_0, v_1, v_2). (b) Triangle orientation (v_2, v_1, v_0). (c) Triangle orientations (v_0, v_1, v_2), (v_1, v_3, v_2), and (v_4, v_3, v_2). Orientations (v_0, v_1, v_2) and (v_1, v_3, v_2) are consistent. Orientation (v_0, v_1, v_2) induces the orientation (v_1, v_2) on edge $\mathbf{e}_1$ while (v_1, v_3, v_2) induces the opposite orientation (v_2, v_1) on edge $\mathbf{e}_1$. Orientations (v_1, v_3, v_2) and (v_4, v_3, v_2) are not consistent. Both orientations induce the same orientation (v_3, v_2) on edge $\mathbf{e}_2$.

orders, either (v_0, v_1, v_2) or (v_2, v_1, v_0). The sequences (v_1, v_2, v_0) and (v_2, v_0, v_1) represent the same cyclic order as (v_0, v_1, v_2). Only the starting vertex has changed. Similarly, the sequences (v_1, v_0, v_2) and (v_0, v_2, v_1) represent the same cyclic order as (v_2, v_1, v_0).

The cyclic order (v_0, v_1, v_2) induces orientations, (v_0, v_1), (v_1, v_2), and (v_2, v_0) of the edges of $\mathbf{t}$. The reverse cyclic order (v_2, v_1, v_0) induces opposite orientations (v_1, v_0), (v_2, v_1), and (v_0, v_2) of the edges of $\mathbf{t}$. Two oriented triangles, $\mathbf{t}_1$ and $\mathbf{t}_2$, which share an edge $\mathbf{e}$ have consistent orientations if the orientation of $\mathbf{e}$ induced by $\mathbf{t}_1$ is the opposite of the orientation of $\mathbf{e}$ induced by $\mathbf{t}_2$. (See Figure 1.5(c).)

The orientation (v_0, v_1, v_2) of a triangle $\mathbf{t}$ in $\mathbb{R}^3$ determines the vector

$$\begin{aligned} u &= (v_1 - v_0) \times (v_2 - v_0) \\ &= v_1 \times v_2 - v_0 \times v_2 - v_1 \times v_0 \\ &= v_0 \times v_1 + v_1 \times v_2 + v_2 \times v_0, \end{aligned}$$

where $\times$ is the cross product. Vector u is orthogonal to $\mathbf{t}$. The sequence (v_1, v_2, v_0) determines the vector $(v_2 - v_1) \times (v_0 - v_1) = v_0 \times v_1 + v_1 \times v_2 + v_2 \times v_0$ that is u. Similarly, (v_2, v_0, v_1) determines the vector $(v_0 - v_2) \times (v_1 - v_2)$, which equals u. Thus, the vector u is independent of the representation of the cycle (v_0, v_1, v_2). The orientation (v_2, v_1, v_0) determines the vector

$$\begin{aligned} (v_1 - v_2) \times (v_0 - v_2) &= v_1 \times v_0 - v_1 \times v_2 - v_2 \times v_0 \\ &= v_1 \times v_0 + v_2 \times v_1 + v_0 \times v_2 \\ &= -u. \end{aligned}$$

Thus, the two orientations of $\mathbf{t}$ determine two opposite vectors, u and $-u$, which are both orthogonal to $\mathbf{t}$.

In computer graphics, triangle orientations are used to determine the front and back faces of triangles. Triangle shading is often dependent on whether the viewer is seeing a front or back face. Thus, it is important that any two triangles that share a common edge have consistent orientations.

Orientations are defined for higher dimensional simplices, where they are also represented by sequences of simplex vertices. The orientation of a $(d-1)$-simplex in $\mathbb{R}^d$ determines a unique vector u orthogonal to the simplex. The opposite orientation determines the vector $-u$. Appendix B.6 contains the definition and discussion of orientations in higher dimensional simplices.

Separation. An important property of isosurfaces is that they "separate" those points with scalar value above the isovalue from those points with scalar value below the isovalue [Nielson et al., 2003]. We give the following formal definition of this concept.

Let $\mathbb{X}$ and $\mathbb{Y}$ be sets of points in $\mathbb{R}^d$. We first define what it means for $\mathbb{X}$ to separate two points in $\mathbb{Y}$.

Definition 1.4.

- Set $\mathbb{X}$ separates point $p \in \mathbb{Y}$ from point $q \in \mathbb{Y}$ if every path in $\mathbb{Y}$ connecting p to q intersects $\mathbb{X}$.
- Set $\mathbb{X}$ strictly separates p from q if $\mathbb{X}$ separates p from q and neither p nor q is in $\mathbb{X}$.

We next define what it means for $\mathbb{X}$ to separate two subsets of $\mathbb{Y}$.

Definition 1.5.

- Set $\mathbb{X}$ separates $\mathbb{Y}_1 \subseteq \mathbb{Y}$ from $\mathbb{Y}_2 \subseteq \mathbb{Y}$ if $\mathbb{X}$ separates every $p \in \mathbb{Y}_2$ from every $q \in \mathbb{Y}_2$.
- Set $\mathbb{X}$ strictly separates $\mathbb{Y}_1 \subseteq \mathbb{Y}$ from $\mathbb{Y}_2 \subseteq \mathbb{Y}$ if $\mathbb{X}$ separates $\mathbb{Y}_1$ from $\mathbb{Y}_2$ and $\mathbb{X}$ does not intersect $\mathbb{Y}_1$ or $\mathbb{Y}_2$.

(See Appendix B.9 for further discussion of separation and its properties.)

Manifolds. A manifold is a mathematical formalization of the intuitive concept of a surface.

Let $\mathbb{B}^k$ be the k-dimensional open ball with radius one centered at the origin. Ball $\mathbb{B}^1$ is an open line segment and $\mathbb{B}^2$ is an open disk. A k-dimensional manifold (k-manifold) is a set of points that locally resembles $\mathbb{B}^k$. Examples of 1-manifolds are circles or simple, closed curves. Every point of a 1-manifold has a small neighborhood that is topologically equivalent to an open line segment ($\mathbb{B}^1$). Examples of 2-manifolds are spheres, tori, or double tori. Every point of a 2-manifold has a small neighborhood that is topologically equivalent to an open disk ($\mathbb{B}^2$).

Let $\mathbb{B}^{k+}$ be the intersection of the open ball $\mathbb{B}^k$ and the closed half-space $\{(x_1, \ldots, x_k) : x_k \geq 0\}$. Note that $\mathbb{B}^{k+}$ is neither closed nor open. $\mathbb{B}^{1+}$ is a line segment open at one endpoint and closed at the other. $\mathbb{B}^{2+}$ is a half-disk, open along the disk and closed at the bounding edge. $\mathbb{B}^{3+}$ is a half-sphere, open along the sphere and closed at the bounding disk. A k-dimensional manifold with boundary (k-manifold with boundary) is a set of points which locally resembles either $\mathbb{B}^k$ or $\mathbb{B}^{k+}$. Examples of 1-manifolds with boundary are line segments or simple curves with two endpoints. Examples of 2-manifolds with boundary are disks or convex polygons (including the polygon interior.) Examples of 3-manifolds with boundary are closed balls or cubes (including the cube interior.) For more precise definitions of k-manifold and k-manifold with boundary, see Appendix B.3.

Piecewise linear manifold. A k-manifold (possibly with boundary) is piecewise linear if it is the union of a set of k-simplices that form a triangulation of the manifold. A piecewise linear manifold is orientable if every simplex in the manifold can be assigned an orientation and these orientations are consistent. The orientation of the manifold is the orientation of all its simplices. If a piecewise

linear manifold is connected and orientable, then assigning an orientation to one simplex fixes the orientations of all the other manifold simplices.

1.7.2 Linear Interpolation

A basic step in MARCHING CUBES and its variants is approximating the intersection of a level set and a line segment. These algorithms use linear interpolation to find a point on the line segment that approximates the intersection.

Let $\phi : \mathbb{R}^d \to \mathbb{R}$ be a scalar field and let $\sigma \in \mathbb{R}$ be an isovalue defining the level set $\phi^{-1}(\sigma)$. Given two grid vertices p and q where $\phi(p) \neq \phi(q)$, if σ is between $\phi(p)$ and $\phi(q)$, then the level set intersects line segment $[p, q]$. We wish to approximate the intersection of $\phi^{-1}(\sigma)$ and line segment $[p, q]$. We do so by defining a linear function $\widehat{\phi}$ based on the two scalar values $\phi(p)$ and $\phi(q)$ and calculating the point $r \in [p, q]$ where $\widehat{\phi}(r) = \sigma$.

Every point on line segment $[p, q]$ can be described as a linear combination of p and q. More specifically, every point on line segment $[p, q]$ equals $(1-\alpha)p+\alpha q$ for some α where $0 \leq \alpha \leq 1$. For example, in $\mathbb{R}^3$ where p equals (p_x, p_y, p_z) and q equals (q_x, q_y, q_z), the linear combination is

$$((1-\alpha)p_x + \alpha q_x,\ (1-\alpha)p_y + \alpha q_y,\ (1-\alpha)p_z + \alpha q_z).$$

Define $\widehat{\phi} : [p, q] \to \mathbb{R}$ by

$$\widehat{\phi}((1-\alpha)p + \alpha q) = (1-\alpha)\phi(p) + \alpha\phi(q).$$

Note that $\widehat{\phi}(p) = \phi(p)$ $(\alpha = 0)$ and $\widehat{\phi}(q) = \phi(q)$ $(\alpha = 1)$. Values of $\widehat{\phi}$ vary linearly with α.

We approximate the intersection of $\phi^{-1}(\sigma)$ with $[p, q]$ as the point r where $\widehat{\phi}(r)$ equals σ. Since r is on line segment $[p, q]$, point r equals $(1-\alpha_r)p + \alpha_r q$ for some α_r. Thus,

$$\sigma = \widehat{\phi}(r) = \widehat{\phi}((1-\alpha_r)p + \alpha_r q) = (1-\alpha_r)\phi(p) + \alpha_r\phi(q).$$

Solving for α_r gives

$$\alpha_r = \frac{\sigma - \phi(p)}{\phi(q) - \phi(p)}.$$

Note that since $\phi(p) \neq \phi(q)$, the denominator $\phi(q) - \phi(p)$ is nonzero.

In $\mathbb{R}^3$, the equations for the coordinates of $r = (r_x, r_y, r_z)$ are

$$\begin{aligned} r_x &= (1-\alpha_r)p_x + \alpha_r q_x, \\ r_y &= (1-\alpha_r)p_y + \alpha_r q_y, \\ r_z &= (1-\alpha_r)p_z + \alpha_r q_z. \end{aligned}$$

```
Input     : Points p, q ∈ R^d, scalar values s_p, s_q, and an isovalue σ.
Requires  : s_p ≠ s_q and either s_p ≤ σ ≤ s_q or s_p ≥ σ ≥ s_q.
Output    : Point r lying on [p, q].

LinearInterpolation(p, s_p, q, s_q, σ)
1 α ← (σ − s_p)/(s_q − s_p);
2 for i = 1 to d do
3 |   r_i ← (1 − α)p_i + αq_i;
4 end
5 return (r);
```

Algorithm 1.1. Linear interpolation.

More generally, in $\mathbb{R}^d$ the equations for the coordinates of $r = (r_1, r_2, \ldots, r_d)$ are

$$\begin{aligned} r_1 &= (1-\alpha_r)p_1 + \alpha_r q_1, \\ r_2 &= (1-\alpha_r)p_2 + \alpha_r q_2, \\ &\cdots \\ r_d &= (1-\alpha_r)p_d + \alpha_r q_d. \end{aligned}$$

Pseudocode is given in Algorithm 1.1.

The assumption for all these algorithms is that $\phi(p)$ does not equal $\phi(q)$. If both $\phi(p)$ and $\phi(q)$ equal σ, then the level set contains both p and q and there is no way to approximate the intersection of $\phi^{-1}(\sigma)$ and $[p, q]$ by a single point. Where or whether the isosurface approximation intersects line segment $[p, q]$ is dependent upon the specific isosurface construction algorithm.

1.7.3 Mesh Representation

The output of surface reconstruction algorithms is a mesh consisting of a set of small, simple surface elements. Typical surface elements are triangles or quadrilaterals. In curve reconstruction, the elements are line segments, while in higher dimensions the elements are simplices, cubes or hypercubes.

A mesh is represented by a list of mesh vertices, $\mathcal{L}_1$, followed by a list, $\mathcal{L}_2$, of surface elements. The list $\mathcal{L}_1$ of mesh vertices contains the mesh vertex coordinates, the location of each mesh vertex in $\mathbb{R}^d$. This representation is called an indexed mesh or a face-vertex mesh.

The list $\mathcal{L}_2$ of surface elements contains the element vertices, the mesh vertices determining the element. For instance, triangles are specified by three vertices, while quadrilaterals are specified by four vertices in order around the

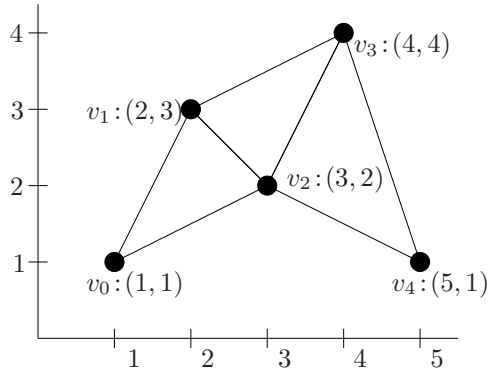


Figure 1.6. Triangle mesh. List of vertices (specified by vertex coordinates): $((1, 1), (2, 3), (3, 2), (4, 4), (5, 1))$. List of triangles (specified by triangle vertices): $((v_0, v_2, v_1), (v_1, v_2, v_3), (v_2, v_4, v_3))$.

quadrilateral. Each mesh vertex stored in $\mathcal{L}_2$ is actually a reference to a mesh vertex in the list $\mathcal{L}_1$.

Figure 1.6 contains an example of a triangle mesh. The list $\mathcal{L}_1$ of mesh vertices for this mesh is $((1, 1), (2, 3), (3, 2), (4, 4), (5, 1))$. The list $\mathcal{L}_2$ of mesh triangles is $((v_0, v_2, v_1), (v_1, v_2, v_3), (v_2, v_4, v_3))$.

CHAPTER 2

MARCHING CUBES AND VARIANTS

In the introduction, we mentioned four different approaches to isosurface construction. In this chapter, we describe one of those approaches to isosurface construction, the widely used MARCHING CUBES algorithm by Lorensen and Cline [Lorensen and Cline, 1987a].

The MARCHING CUBES algorithm is based on two ideas. First, the isosurface can be constructed piecewise within each cube of the grid without reference to other grid cubes. Second, the combinatorial structure of each isosurface patch in a grid cube can be retrieved from a lookup table. Since the main operation is retrieving this structure from the lookup table, the algorithm runs in time proportional to the number of grid cubes.

We first present a two-dimensional version of the algorithm, called MARCHING SQUARES, for constructing two-dimensional isocontours. Before discussing the MARCHING SQUARES algorithm, we define some terminology that will be used by the algorithms in this chapter.

2.1 Definitions

Given a regular scalar grid and an isovalue σ, it is convenient to assign "+" and "−" labels to each grid vertex based on the relationship between its scalar value and σ.

Definition 2.1.

- A grid vertex is positive, "+", if its scalar value is greater than or equal to σ.
- A grid vertex is negative, "−", if its scalar value is less than σ.
- A positive vertex is strictly positive if its scalar value does not equal σ.

Since the scalar value of a negative vertex never equals the isovalue, there is no point in defining a similar "strictly negative" term.

Grid edges can be characterized by the labels at their endpoints.

Definition 2.2.

- A grid edge is positive if both its endpoints are positive.
- A grid edge is negative if both its endpoints are negative.
- A positive grid edge is strictly positive if both its endpoints are strictly positive.
- A grid edge is bipolar if one endpoint is positive and one endpoint is negative.

Note that a grid vertex or edge is only positive or negative in relationship to some isovalue.

The definitions given above apply not just to regular scalar grids but also to curvilinear grids. They also apply to the vertices and edges of polyhedral meshes such as tetrahedral and simplicial meshes.

2.2 Marching Squares

2.2.1 Algorithm

Input to the MARCHING SQUARES algorithm is an isovalue and a set of scalar values at the vertices of a two-dimensional regular grid. The algorithm has three steps. (See Figure 2.1.) Read in the isocontour lookup table from a pre-constructed data file. For each square, retrieve from the lookup table a set of

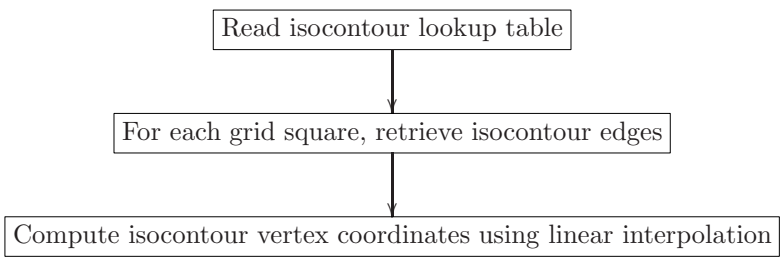


Figure 2.1. MARCHING SQUARES.

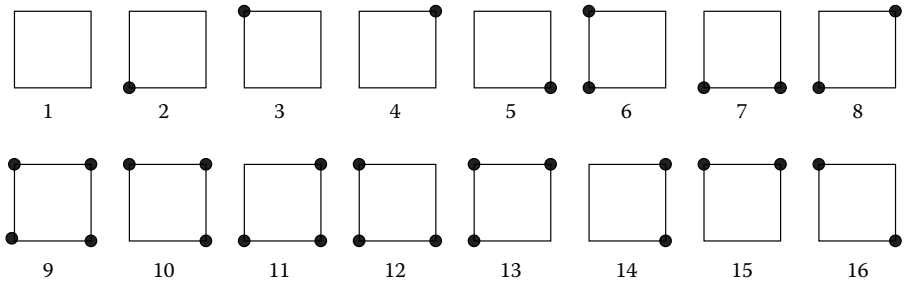


Figure 2.2. Square configurations. Black vertices are positive.

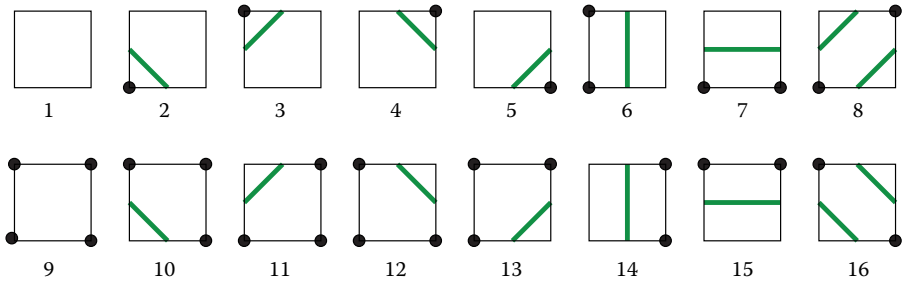


Figure 2.3. Square isocontours. Configurations 1 and 9 have no isocontour. Isocontours for configurations 2–7 and 10–15 are single line segments. Isocontours for configurations 8 and 16 are two line segments.

isocontour edges representing the combinatorial structure of the isocontour. The endpoints of these edges form the isocontour vertices. Assign geometric locations to the isocontour vertices based on the scalar values at the square edge endpoints. We explain the last two steps of the algorithm next.

Each grid vertex is labeled positive or negative as described in Section 2.1. (See Figure 2.4(b) for an example.) Since a square has four vertices, there are $2^4 = 16$ different configurations of square vertex labels. These configurations are listed in Figure 2.2.

The combinatorial structure of the isocontour within each square is determined from the configuration of the square's vertex labels. In order to separate the positive vertices from the negative ones, the isocontour must intersect any square edge that has one positive and one negative endpoint. An isocontour that intersects a minimal number of grid edges will not intersect any square edge whose endpoints are both strictly positive or whose endpoints are both negative.

For each square configuration κ, let $E_\kappa^{+/-}$ be the set of bipolar edges. Note that the size of $E_\kappa^{+/-}$ is either zero, two, or four. Pair the edges of $E_\kappa^{+/-}$. Each such pair represents an isocontour edge with endpoints on the two elements of the pair. Figure 2.3 contains the sixteen square configurations and their

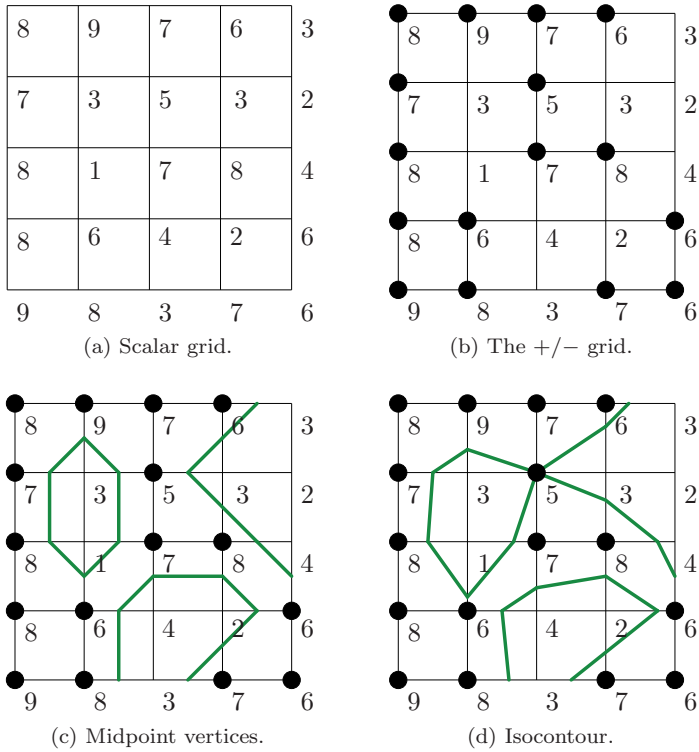


Figure 2.4. (a) 2D scalar grid. (b) Black vertices are positive. Vertex v with scalar value s_v is positive if $s_v >= 5$ and negative if $s_v < 5$. Note that $s_v = 5$ for one grid vertex v. (c) Isocontour with vertices at edge midpoints (before linear interpolation). (d) Isocontour with isovalue 5.

isocontours. The isocontour lookup table, Table, contains sixteen entries, one for each configuration. Each entry, Table$[\kappa]$ is a list of the $E_\kappa^{+/-}$ pairs.

In Figure 2.3 the isocontour edges are drawn connecting the midpoints of each square edge. This is for illustration purposes only. The geometric locations of the isocontour vertices are not defined by the lookup table.

The isocontour lookup table is constructed on the unit square with vertices $(0,0),(1,0),(0,1),(1,1)$. To construct the isocontour in grid square (i,j), we have to map pairs of unit square edges to pairs of square (i,j) edges. Each vertex $v = (v_x, v_y)$ of the unit square maps to $v + (i,j) = (v_x, v_y) + (i,j) = (v_x + i, v_y + j)$. Each edge $\mathbf{e}$ of the unit square with endpoints (v, v') maps to edge $\mathbf{e} + (i,j) = (v + (i,j), v' + (i,j))$. Finally, each edge pair $(\mathbf{e}_1, \mathbf{e}_2)$ maps to $(\mathbf{e}_1 + (i,j), \mathbf{e}_2 + (i,j))$.

The endpoints of the isocontour edges are the isocontour vertices. To map each isocontour edge to a geometric line segment, we use linear interpolation to

```
Input  : F is a 2D array of scalar values.
         Coord is a 2D array of (x, y) coordinates.
         σ is an isovalue.
Result : A set ϒ of isocontour line segments.

MarchingSquares(F, Coord, σ, ϒ)
Read Marching Squares lookup table into Table;
/* Assign "+" or "−" signs to each vertex                          */
foreach grid vertex (i, j) do
    if F[i, j] < σ then Sign[i, j] ← "−";
    else Sign[i, j] ← "+";                          /* F[i, j] ≥ σ */
end
S ← ∅;
/* For each grid square, retrieve isocontour edges                 */
foreach grid square (i, j) do
    /* Grid square vertices are (i, j), (i+1, j), (i, j+1), (i+1, j+1) */
    κ ← (Sign[i, j], Sign[i+1, j], Sign[i, j+1], Sign[i+1, j+1]);
    foreach edge pair (e1, e2) ∈ Table[κ] do
        Insert edge pair (e1 + (i, j), e2 + (i, j)) into S;
    end
end
/* Compute isocontour vertex coordinates using linear interpolation */
foreach bipolar grid edge e with endpoints (i1, j1) and (i2, j2) do
    /* Compute the isosurface vertex we on edge e                  */
    we ← LinearInterpolation
              (Coord[i1, j1], F[i1, j1], Coord[i2, j2], F[i2, j2], σ);
end
/* Convert S to set of line segments                               */
ϒ ← ∅;
foreach pair of edges (e1, e2) ∈ S do
    ϒ ← ϒ ∪ {(we1, we2)};
end
```

Algorithm 2.1. MARCHING SQUARES.

position the isocontour vertices as described in Section 1.7.2. Each isocontour vertex v lies on a grid edge $[p, q]$. If s_p and s_q are the scalar values at p and q and σ is the isovalue, then map v to $(1-\alpha)p + \alpha q$ where $\alpha = (\sigma - s_p)/(s_q - s_p)$. Note that since p and q have different signs, scalar s_p does not equal s_q and the denominator $(s_q - s_p)$ is never zero.

The MARCHING SQUARES algorithm is presented in Algorithm 2.1. Function `LinearInterpolation`, called by this algorithm, is defined in Algorithm 1.1 in Section 1.7.2.

Figure 2.4 contains an example of a scalar grid, an assignment of positive and negative labels to the grid vertices, the isocontour before linear interpolation, and the final isocontour after linear interpolation.

2.2.2 Running Time

The MARCHING SQUARES algorithm runs in linear time.

Proposition 2.3. *Let N be the total number of vertices of a 2D scalar grid. The running time of the* MARCHING SQUARES *algorithm on the scalar grid is $\Theta(N)$.*

Proof: Reading the MARCHING SQUARE lookup table takes constant time. Each grid square is processed once. At each grid square, at most two isocontour edges are retrieved from the lookup table. Since the number of grid squares is bounded by the number of grid vertices, determining the isocontour edges takes $O(N)$ time.

Computing the isocontour vertex on each grid edge takes time proportional to the number of isocontour vertices. Since each grid edge has at most one isocontour edge, the time to compute isocontour vertices is proportional the number of grid edges. The number of grid edges is less than twice the number of grid vertices, so the number of grid edges is at most $2N$. Thus computing the isocontour vertices takes $O(N)$ time.

The algorithm examines every grid square, so its running time has an $\Omega(N)$ lower bound. Thus, the running time of the MARCHING SQUARES algorithm is $\Theta(N)$. □

2.2.3 Isocontour Properties

To properly discuss the output produced by the MARCHING SQUARES algorithm, we need to differentiate between two cases based on the isovalue. In the first case, the isovalue does not equal the scalar value of any grid vertex. In this case, the MARCHING SQUARES algorithm produces a piecewise linear 1-manifold with boundary. The boundary of the 1-manifold lies on the boundary of the grid. In the second case, the isovalue equals the scalar value of one or more grid vertices. In this case, the MARCHING SQUARES algorithm may not produce a 1-manifold with boundary or the boundary may not lie on the boundary of the grid. For instance, the MARCHING SQUARES algorithm applied to the 3×3 grids in Figures 2.5 and 2.6 produces non-manifold isocontours or isocontours with boundary not on the scalar grid. In Figure 2.5(a), four isocontour line segments intersect at a single point; in Figure 2.5(b), the isocontour is a single point, and in Figure 2.6, the boundary of the isocontour lies inside the grid.

The two cases also differ in the nature of the line segments produced by the algorithm. The isocontour produced by the MARCHING SQUARES algorithm is

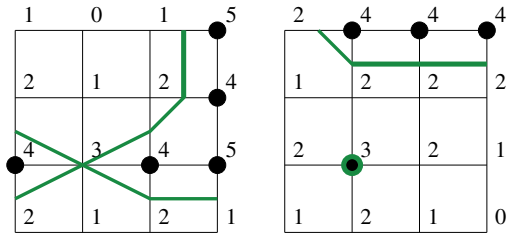


Figure 2.5. Examples of non-manifolds produced by MARCHING SQUARES (isovalue 3). Black vertices are positive. (a) Four curves joining at the grid vertex with isovalue 3. (b) Isosurface includes an isolated point at the grid vertex with isovalue 3.

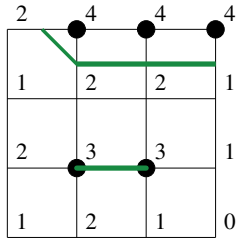


Figure 2.6. Examples of a manifold produced by MARCHING SQUARES whose boundary does not lie on the grid boundary (isovalue 3). Black vertices are positive.

a set of line segments whose vertices lie on the grid edges. If the isovalue does not equal the scalar value of any grid vertex, then these line segments all have positive length. If the isovalue equals the scalar value of one or more grid vertices, then the isocontour may have zero-length edges. For instance, the MARCHING SQUARES algorithm applied to the three grids in Figure 2.7 produces isocontours for isovalue 3 with zero-length edges.

In Figure 2.7(a), the lower-left grid square has configuration 4, producing a single isocontour edge, but both endpoints of that edge map to the vertex in the middle of the grid. In Figure 2.7(b), each grid square produces an isocontour edge, but all four edges have zero length and collapse to a single point. In Figure 2.7(c), leftmost and rightmost grid squares produce zero-length isocontour edges and two middle grid squares produce two duplicate isocontour edges on a grid edge.

MARCHING SQUARES returns a finite set, Υ, of line segments. The isocontour is the union of those line segments. The vertices of the isocontour are the endpoints of the line segments.

The following properties apply to all isocontours produced by the MARCHING SQUARES algorithm.

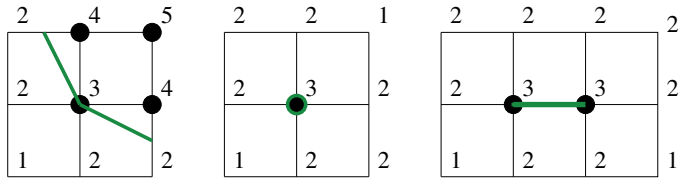


Figure 2.7. Examples of zero-length contour edges produced by MARCHING SQUARES (isovalue 3). Black vertices are positive. (a) Isocontour with one zero-length isocontour edge (from lower-left grid square). (b) Isocontour with four zero-length isocontour edges. (c) Another isocontour with four zero-length isocontour edges. Isocontour also has two duplicate nonzero isocontour edges (from the two middle grid squares).

Property 1. *The isocontour is piecewise linear.*

Property 2. *The vertices of the isocontour lie on grid edges.*

Property 3. *The isocontour intersects every bipolar grid edge at exactly one point.*

Property 4. *The isocontour does not intersect any negative or strictly positive grid edges.*

Property 5. *The isocontour separates positive grid vertices from negative grid vertices and strictly separates strictly positive grid vertices from negative grid vertices.*

Set $\mathbb{Y} \subseteq \mathbb{X}$ separates point $p \in \mathbb{X}$ from point $q \in \mathbb{X}$ if every path in $\mathbb{X}$ connecting p to q intersects $\mathbb{Y}$. Set $\mathbb{Y}$ strictly separates p from q if $\mathbb{Y}$ separates p from q and neither p nor q is on $\mathbb{Y}$. (See Section 1.7.1 and Appendix B.9.)

Properties 3 and 4 imply that the isocontour intersects a minimum number of grid edges. If both endpoints of a grid edge have scalar value equal to the isovalue, then the isocontour may intersect the grid edge zero, one, or two times or may contain the grid edge. (See Figure 2.8.)

A grid vertex may have scalar value equal to the isovalue and yet no isocontour passes through any edge containing that grid vertex. For instance, the MARCHING SQUARES algorithm returns the empty set when run on the scalar grid in Figure 2.9 with isovalue 3. Each vertex, including the center vertex, is positive, so each grid square has configuration 9 (Figure 2.2) and has no isocontour edges.

By Property 3, the isocontour intersects every bipolar grid edge. However, the bipolar grid edge may be intersected by zero-length isocontour edges as in Figure 2.7(b).

The following properties apply to MARCHING SQUARES isocontours whose isovalues do not equal the scalar value of any grid vertex.

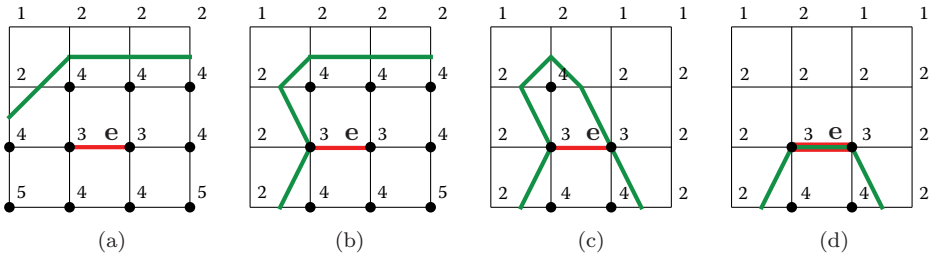


Figure 2.8. Examples of grid edges with both endpoint scalar values equal to the isovalue (3). Black vertices are positive. (a) Red grid edge **e** does not intersect the isocontour. (b) Red grid edge **e** intersects the isocontour at one endpoint. (c) Red grid edge **e** intersects the isocontour at both endpoints. (d) Red grid edge **e** is contained in the isocontour.

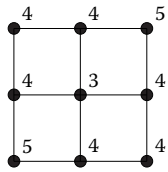


Figure 2.9. Example of a scalar grid whose MARCHING SQUARES isocontour is the empty set, even though the center grid vertex has scalar value equal to the isovalue 3. All vertices are positive.

Property 6. *The isocontour is a piecewise linear 1-manifold with boundary.*

Property 7. *The boundary of the isocontour lies on the boundary of the grid.*

Property 8. *Set Υ does not contain any zero-length line segments or duplicate line segments, and the line segments in Υ form a "triangulation" of the isocontour.*

The triangulation in Property 8 simply means that line segments in Υ intersect at their endpoints. The isocontour is one-dimensional and does not contain any triangles.

2.2.4 Proof of Isocontour Properties

We give a proof of each of the properties listed in the previous section.

Property 1. *The isocontour is piecewise linear.*

Property 2. *The vertices of the isocontour lie on grid edges.*

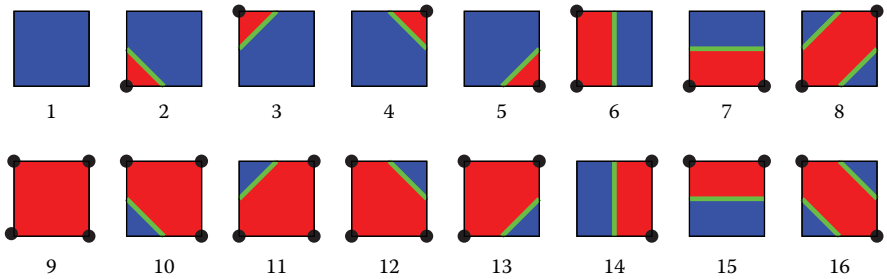


Figure 2.10. Red, positive regions and blue, negative regions for each square configuration. The green isocontour is part of the positive region. Black vertices are positive.

Proof of Properties 1 & 2: The MARCHING SQUARES isocontour consists of a finite set of line segments, so it is piecewise linear. These line segments intersect only at their endpoints and thus form a triangulation of the isocontour. The endpoints of these line segments lie on the grid edges, confirming Property 2. □

Property 3. *The isocontour intersects every bipolar grid edge at exactly one point.*

Property 4. *The isocontour does not intersect any negative or strictly positive grid edges.*

Proof of Properties 3 & 4: Each isocontour edge is contained in a grid square. Since the grid squares are convex, only isocontour edges with endpoints (vertices) on the grid edge intersect the grid edge. If the grid edge has one positive and one negative endpoint, the unique location of the isocontour vertex on the grid edge is determined by linear interpolation. Thus the isocontour intersects a bipolar grid edge at only one point.

If the grid edge is negative or strictly positive, then no isocontour vertex lies on the grid edge. Thus the isocontour does not intersect negative or strictly positive grid edges. □

Within each grid square the isocontour partitions the grid square into two regions. Let the positive region for a grid square $\mathbf{c}$ be the set of points which can be reached by a path ζ from a positive vertex. More precisely, a point p is in the positive region of $\mathbf{c}$ if there is some path $\zeta \subset \mathbf{c}$ connecting p to a positive vertex of $\mathbf{c}$ such that the interior of ζ does not intersect the isocontour. A point p is in the negative region of $\mathbf{c}$ if there is some path $\zeta \subset \mathbf{c}$ connecting p to a negative vertex of $\mathbf{c}$ such that ζ does not intersect the isocontour. Since any path $\zeta \subset \mathbf{c}$ from a positive to a negative vertex must intersect the isocontour, the positive and negative regions form a partition of the square $\mathbf{c}$. Figure 2.10 illustrates the positive and negative regions, colored red and blue, respectively, for each square configuration.

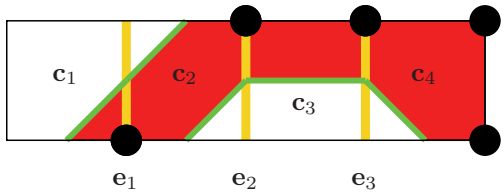


Figure 2.11. Adjacent grid squares, $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$, and $\mathbf{c}_4$, and their positive (red) regions, $R^+_{\mathbf{c}_1}$, $R^+_{\mathbf{c}_2}$, $R^+_{\mathbf{c}_3}$ and $R^+_{\mathbf{c}_4}$, respectively. Yellow edges $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ separate the squares. Positive regions agree on the grid square boundaries, i.e., $R^+_{\mathbf{c}_1} \cap \mathbf{e}_1 = R^+_{\mathbf{c}_2} \cap \mathbf{e}_1$ and $R^+_{\mathbf{c}_2} \cap \mathbf{e}_2 = R^+_{\mathbf{c}_3} \cap \mathbf{e}_2$ and $R^+_{\mathbf{c}_3} \cap \mathbf{e}_3 = R^+_{\mathbf{c}_4} \cap \mathbf{e}_3$.

Note the asymmetry in the definitions of the positive and negative regions. For the positive region the *interior* of ζ does not intersect the isocontour, while for the negative region the entire path ζ must not intersect the isocontour. Thus, the positive region contains the isocontour while the negative region does not. The positive region is also closed. Any point within the positive region that does not lie in the isocontour has a neighborhood contained in the positive region.

Every negative vertex is contained in the negative region since the zero-length path connects the vertex to itself. Similarly, every positive vertex is contained in the positive region.

Let $R^+_{\mathbf{c}}$ be the positive region for a grid square $\mathbf{c}$. We claim that positive and negative regions agree on the grid square boundaries. For instance, in Figure 2.11 $R^+_{\mathbf{c}_1} \cap \mathbf{e}_1$ equals $R^+_{\mathbf{c}_2} \cap \mathbf{e}_1$ where $R^+_{\mathbf{c}_1}$ and $R^+_{\mathbf{c}_2}$ are the positive regions for grid squares $\mathbf{c}_1$ and $\mathbf{c}_2$, respectively, and $\mathbf{e}_1$ is the edge between $\mathbf{c}_1$ and $\mathbf{c}_2$. Similarly, $R^+_{\mathbf{c}_2} \cap \mathbf{e}_2$ equals $R^+_{\mathbf{c}_3} \cap \mathbf{e}_2$ and $R^+_{\mathbf{c}_3} \cap \mathbf{e}_3$ equals $R^+_{\mathbf{c}_4} \cap \mathbf{e}_3$.

Lemma 2.4. *Let $\mathbf{c}_1$ and $\mathbf{c}_2$ be adjacent grid squares where each vertex of $\mathbf{c}_1$ and $\mathbf{c}_2$ has a positive or a negative label. Let p be a point in $\mathbf{c}_1 \cap \mathbf{c}_2$. Point p is in $R^+_{\mathbf{c}_1}$ if and only if p is in $R^+_{\mathbf{c}_2}$.*

Proof: If p is a grid vertex, then p is in $R^+_{\mathbf{c}_1}$ and $R^+_{\mathbf{c}_2}$ if it is positive and not in $R^+_{\mathbf{c}_1}$ or $R^+_{\mathbf{c}_2}$ if it is negative. Otherwise, p must be in the interior of some grid edge $\mathbf{e}$. If edge $\mathbf{e}$ is positive, then p is in $R^+_{\mathbf{c}_1}$ and $R^+_{\mathbf{c}_2}$. If edge $\mathbf{e}$ is negative, then p is not in $R^+_{\mathbf{c}_1}$ or $R^+_{\mathbf{c}_2}$. If one endpoint, v_1, is positive and the other endpoint, v_2, is negative, then the isocontour in both grid squares intersects the grid edge in the same interpolated point q. The closed segment $[v_1, q]$ is in both $R^+_{\mathbf{c}_1}$ and $R^+_{\mathbf{c}_2}$ while the segment $(q, v_2]$ (open at q and closed at v_2) is in neither. Thus if p is in $[v_1, q]$, then p is in both $R^+_{\mathbf{c}_1}$ and $R^+_{\mathbf{c}_2}$ and if p is in $(q, v_2]$, then p is in neither. □

Using Lemma 2.4, we prove that the isocontour separates positive vertices from negative ones.

Property 5. *The isocontour separates positive grid vertices from negative grid vertices and strictly separates strictly positive grid vertices from negative grid vertices.*

Proof: For all the possible configurations, a path from a positive vertex to a negative one in a grid square must intersect the isocontour. We must show that this also holds true for paths through many grid squares.

Let R^+ be the union of the positive regions over all the grid squares. Consider a path ζ in the grid from a positive grid vertex to a negative one. The positive grid vertex lies in R^+ while the negative one does not. Thus ζ must intersect some point p on the boundary of R^+ where it crosses out of R^+. Every neighborhood of p must contain points that are not in R^+.

Since R^+ is closed, point p lies in R^+. Thus point p lies in $R^+_{\mathbf{c}'}$ for some grid square $\mathbf{c}'$. By Lemma 2.4, point p lies in $R^+_{\mathbf{c}}$ for every grid square $\mathbf{c}$ containing p. Assume p is not on the isocontour. Within each grid square containing p, some neighborhood of p is contained in the positive region for that grid square. The union of those neighborhoods is a neighborhood of p within the grid and is contained in R^+. Thus ζ does not cross out of R^+ at p. We conclude that p must lie on the isocontour and that ζ intersects the isocontour. Thus the isocontour separates positive from negative grid vertices.

If the scalar value of a grid vertex does not equal the isovalue, then the grid vertex does not lie on the isocontour. Thus the isocontour strictly separates strictly positive grid vertices from negative ones. (By definition, the scalar value of a negative vertex never equals the isovalue.) □

To prove properties 6 and 7, we prove something slightly more general.

Proposition 2.5. *Let p be any point on the* Marching Squares *isocontour that is not a grid vertex with scalar value equal to the isovalue.*

1. *If p is in the interior of the grid, then the isocontour restricted to some sufficiently small neighborhood of p is a 1-manifold.*
2. *If p is on the boundary of the grid, then the isocontour restricted to some sufficiently small neighborhood of p is a 1-manifold with boundary.*

Proof: Let v be a grid vertex with scalar value s_v. If s_v is not the isovalue, then the isocontour does not contain v, so point p is not v. If s_v equals the isovalue, then, by assumption, point p is not v. Therefore, point p is not a grid vertex.

If p lies in the interior of a grid square, then it lies in the interior of some isocontour edge. The interior of this edge is a 1-manifold containing p.

Assume p lies on the boundary of a grid square but not on the boundary of the grid. Since p is not a grid vertex, point p must lie in the interior of some grid edge $\mathbf{e}$ with one positive and one negative vertex. The two grid squares containing $\mathbf{e}$ each contain a single contour edge with endpoint at p. The interior of these two contour edges and the point p form a 1-manifold containing p.

Finally, assume p lies on the boundary of the grid. Since p is not a grid vertex, point p is contained in a single grid square. This grid square contains a single contour edge with endpoint at p. This contour edge is a manifold with boundary containing p. □

Properties 6 and 7 apply to MARCHING SQUARES isocontours whose isovalues do not equal the scalar value of any grid vertex.

Property 6. *The isocontour is a piecewise linear 1-manifold with boundary.*

Property 7. *The boundary of the isocontour lies on the boundary of the grid.*

Proof of Properties 6 & 7: Consider a point p on the isocontour. Since the isovalue does not equal the scalar value of any grid vertex, point p is not a grid vertex. By Proposition 2.5, the isocontour restricted to some suitably small neighborhood of point p is either a 1-manifold or a 1-manifold with boundary. Thus the isocontour is a 1-manifold with boundary. Since the restricted isocontour is a 1-manifold whenever p is in the interior of the grid, the boundary of the isocontour must lie on the grid boundary. □

The last property is that Υ does not contain any zero-length or duplicate edges and forms a triangulation of the isocontour.

Property 8. *Set Υ does not contain any zero-length line segments or duplicate line segments, and the line segments in Υ form a "triangulation" of the isocontour.*

Proof: Since no grid vertex has scalar value equal to the isovalue, no isocontour vertex lies on a grid vertex. By Property 4, each bipolar grid edge contains only one isocontour vertex. Thus, the linear interpolation on isocontour vertices does not create any zero-length or duplicate isocontour edges. Since isocontour edges intersect only at their endpoints, Υ forms a triangulation of the isocontour. □

2.2.5 2D Ambiguity

Set $E_\kappa^{+/-}$ is the set of bipolar square edges for configuration κ. The combinatorial structure of the isocontour depends upon the matching of the elements of $E_\kappa^{+/-}$. If $E_\kappa^{+/-}$ has two elements, then there is no choice. However, if $E_\kappa^{+/-}$ has four bipolar edges, then there are two possible pairings and two possible isocontours that could be constructed for configuration κ. Configurations 8 and 16 from Figure 2.2 have four bipolar edges. They are called ambiguous configurations. These two ambiguous configurations are shown in Figure 2.12 along with the two combinatorially distinct isocontours for each ambiguous configuration.

Choosing different isocontours for the ambiguous configurations will change the topology of the overall isocontour. For instance, Figure 2.13 shows the same

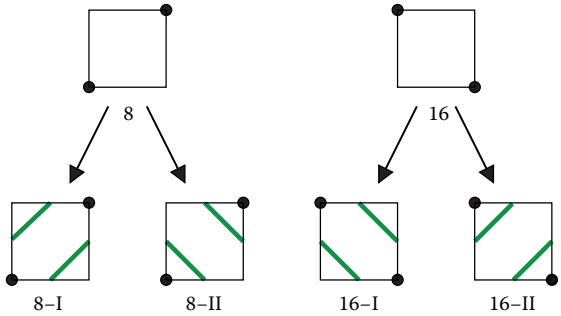


Figure 2.12. Ambiguous square configurations.

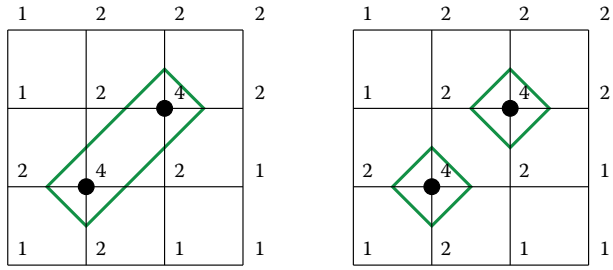


Figure 2.13. Topologically distinct isocontours created by using different isocontours for the ambiguous configuration in the central grid square.

scalar grid with two topologically distinct isocontours created by different resolutions of the ambiguous configurations. The first isocontour has two components while the second has one.

While the choice of isocontours for the ambiguous configurations changes the isocontour topology, any of the choices will produce isocontours that are 1-manifolds and strictly separate strictly positive vertices from negative vertices. As we shall see, this is not true in three dimensions.

2.3 Marching Cubes

2.3.1 Algorithm

The three-dimensional MARCHING CUBES algorithm follows precisely the steps in the two-dimensional MARCHING SQUARES algorithm. Input to the MARCH-

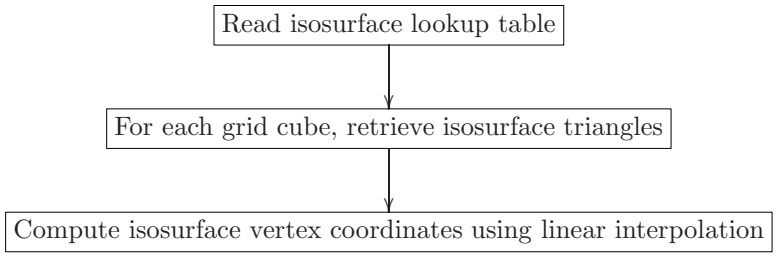


Figure 2.14. MARCHING CUBES.

ING CUBES algorithm is an isovalue and a set of scalar values at the vertices of a three-dimensional regular grid. The algorithm has three steps. (See Figure 2.14.) Read the isosurface lookup table from a preconstructed data file. For each cube, retrieve from the lookup table a set of isosurface triangles representing the combinatorial structure of the isosurface. The vertices of these triangles form the isosurface vertices. Assign geometric locations to the isosurface vertices based on the scalar values at the cube edge endpoints. We explain the last two steps below.

Grid vertices are labeled positive or negative as described in Section 2.1. Grid edges are labeled positive, negative, or bipolar.

The combinatorial structure of the isosurface within each cube is determined from the configuration of the cube's vertex labels. In order to separate the positive vertices from the negative ones, the isosurface must intersect any cube edge that has one positive and one negative endpoint. An isosurface that intersects a minimal number of grid edges will not intersect any edge whose endpoints are both strictly positive or whose endpoints are both negative.

Since each vertex is either positive or negative and a cube has eight vertices, there are $2^8 = 256$ different configurations of cube vertex labels. Many of these configurations are rotations or reflections of one another. By exploiting this symmetry, the number of distinct configurations can be reduced to twenty-two.[1] These distinct configurations are listed in Figure 2.15. All other configurations are rotations or reflections of these twenty-two.

For each cube configuration κ, let $E_\kappa^{+/-}$ be the set of edges with one positive and one negative endpoint. The isosurface lookup table contains 256 entries, one for each configuration κ. Each entry is a list of triples of edges of $E_\kappa^{+/-}$. Each triple $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ represents a triangle whose vertices lie on $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$. The list of triples define the combinatorial structure of the isosurface patch for

[1] Lorensen and Cline's original paper on MARCHING CUBES [Lorensen and Cline, 1987a] listed only fifteen configurations. For reasons discussed in Section 2.3.5, twenty-two configurations are preferable.

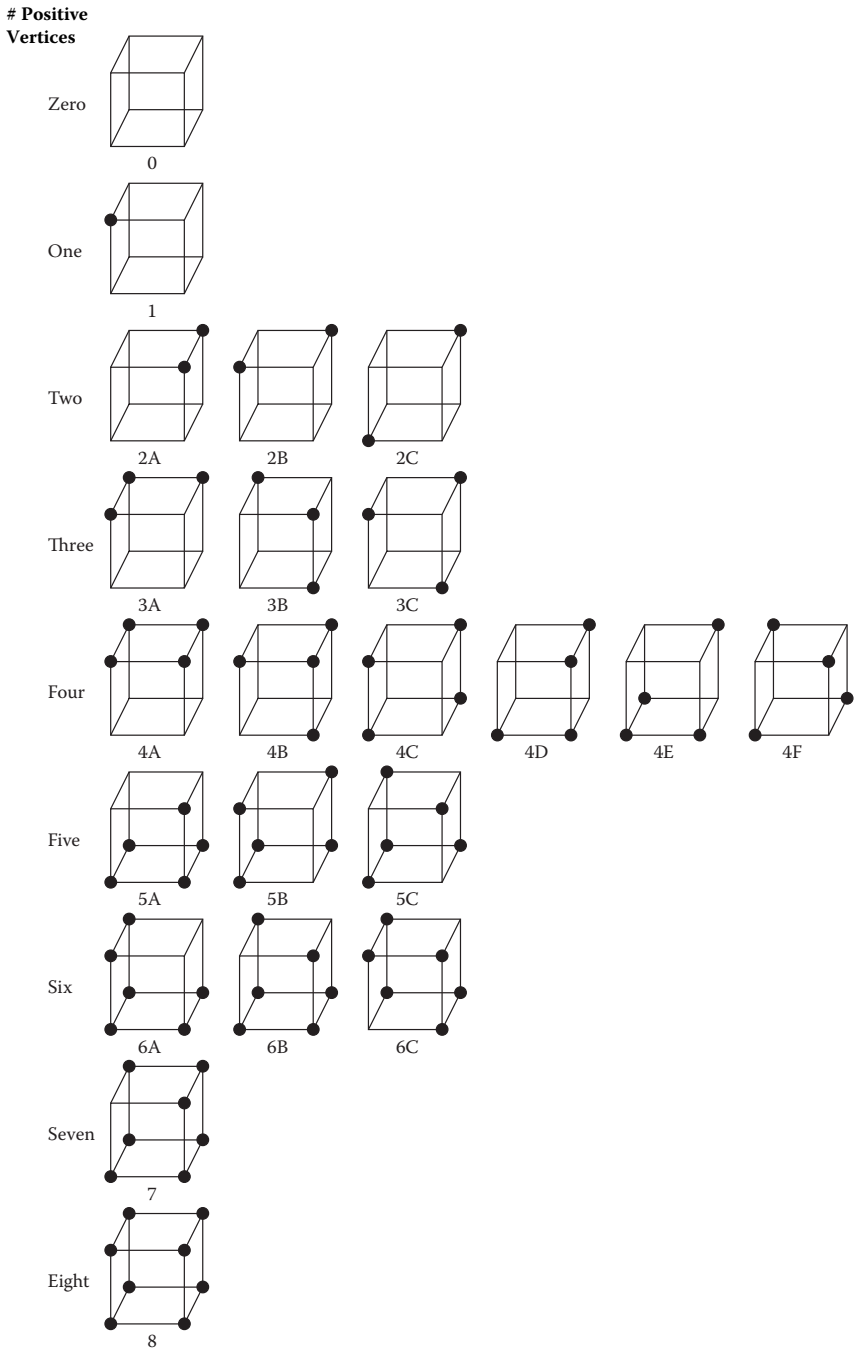


Figure 2.15. Twenty-two distinct cube configurations. Black vertices are positive.

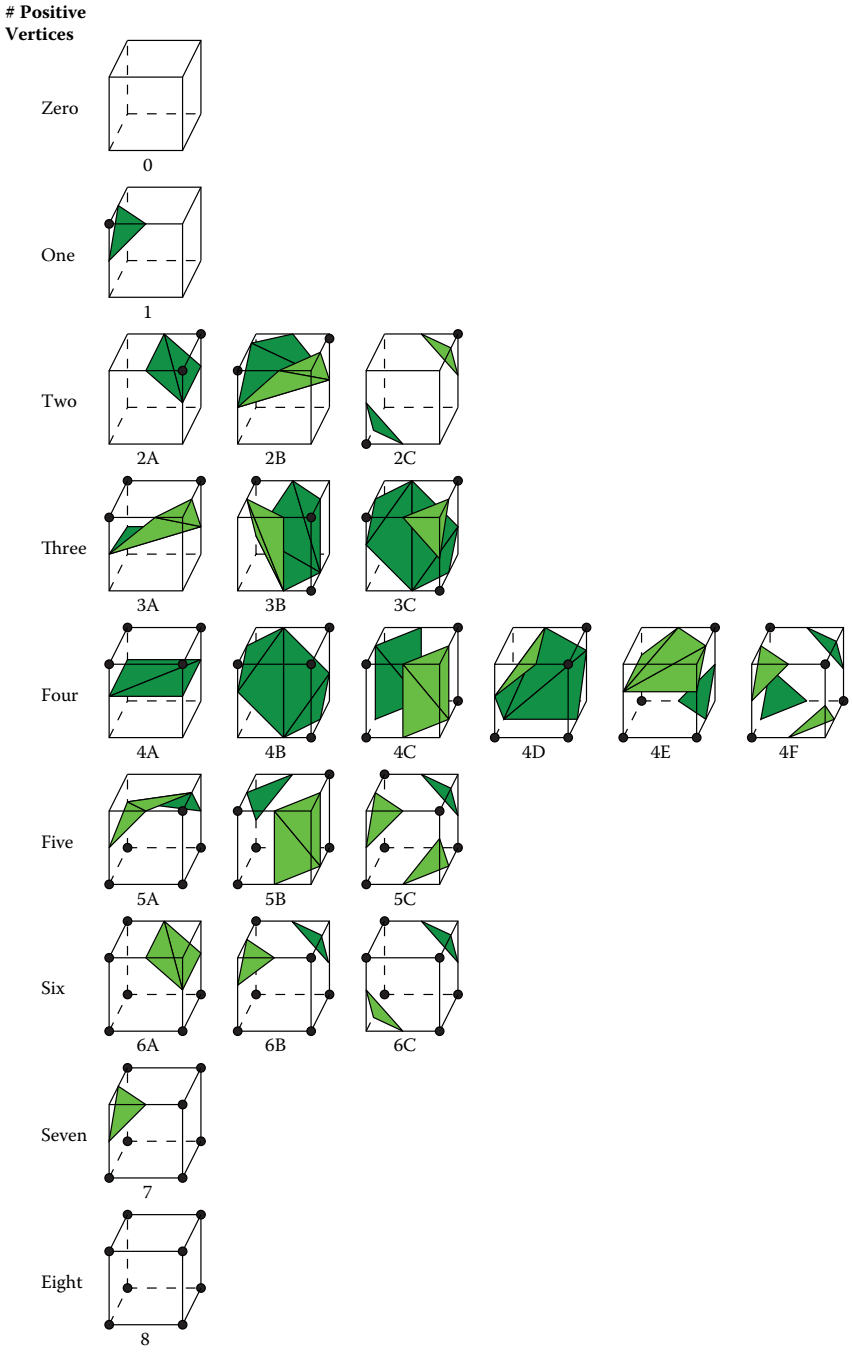


Figure 2.16. Isosurfaces for twenty-two distinct cube configurations.

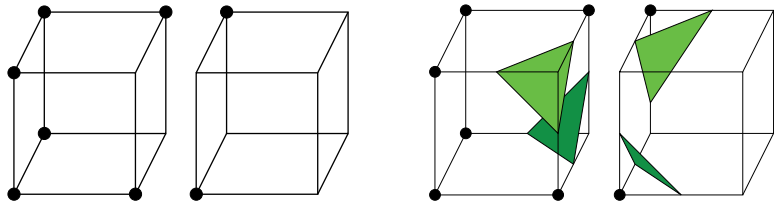


Figure 2.17. (a) Adjacent configurations sharing a common face. (b) Incompatible isosurface patches for the adjacent configurations.

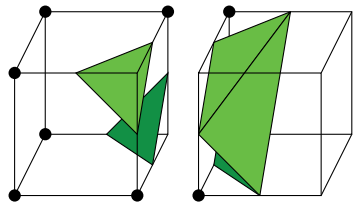


Figure 2.18. Compatible isosurface patches for adjacent configurations in Figure 2.17(a).

configuration κ. The isosurface patch intersects every edge of $E_\kappa^{+/-}$ exactly once and does not intersect any other grid cube edges.

To define the 256 entries in the table, it is only necessary to determine the table entries for the twenty-two distinct configurations. The table entries for the other configurations can be derived using rotation and reflection symmetry. Figure 2.16 contains the twenty-two distinct cube configurations and their isosurfaces.

The isosurface lookup table is constructed on the unit cube with vertices $(0,0,0),(1,0,0),(0,1,0),\ldots,(0,1,1),(1,1,1)$. To construct the isosurface in grid cube (i,j,k), we have to map unit cube edges to edges of cube (i,j,k). Each vertex $v=(v_x,v_y,v_z)$ of the unit cube maps to $v+(i,j,k)=(v_x,v_y,v_z)+(i,j,k)=(v_x+i,v_y+j,v_z+k)$. Each edge $\mathbf{e}$ of the unit square with endpoints (v,v') maps to edge $\mathbf{e}+(i,j,k)=(v+(i,j,k),v'+(i,j,k))$. Finally, each edge triple $(\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3)$ maps to $(\mathbf{e}_1+(i,j,k),\mathbf{e}_2+(i,j,k),\mathbf{e}_3+(i,j,k))$.

In Figure 2.16, the isosurface vertices lie on the midpoints of the grid edges. This is for illustration purposes only. The geometric locations of the isosurface vertices are not defined by the lookup table.

The vertices of the isosurface triangles are the isosurface vertices. To map each isosurface triangle to a geometric triangle, we use linear interpolation to position the isosurface vertices as described in Section 1.7.2. Each isosurface vertex v lies on a grid edge $[p,q]$. If s_p and s_q are the scalar values at p and q and σ is the isovalue, then map v to $(1-\alpha)p+q$ where $\alpha=(\sigma-s_p)/(s_q-s_p)$.