Implicit Objects in Computer Graphics

Luiz Velho
Jonas Gomes
Luiz Henrique de Figueiredo

Springer

Implicit Objects in Computer Graphics

Springer
New York
Berlin
Heidelberg
Hong Kong
London
Milan
Paris
Tokyo

Luiz Velho Jonas Gomes
Luiz Henrique de Figueiredo

Implicit Objects in Computer Graphics

With 65 Figures

Springer

Luiz Velho
Jonas Gomes
Luiz Henrique de Figueiredo
Instituto de Matematica Pura e Aplicada
Rio de Janeiro, RJ 22460-320
Brazil

Library of Congress Cataloging-in-Publication Data
Velho, Luiz.
Implicit objects in computer graphics / Luiz Velho, Jonas Gomes, Luiz Henrique de Figueiredo.
p. cm.
Includes bibliographical references and index.
ISBN 0-387-98424-0 (alk. paper)
1. Computer graphics. I. Gomes, Jonas. II. Figueiredo, Luiz Henrique de.
III. Title.
TS385 .V45 2002
006.6—dc21 2002016005

ISBN 0-387-98424-0 Printed on acid-free paper.

Printed in the United States of America.

9 8 7 6 5 4 3 2 1 SPIN 10658083

Typesetting: Pages created by the authors using a Springer LaTeX2e macro package.

www.springer-ny.com

Springer-Verlag New York Berlin Heidelberg
A member of BertelsmannSpringer Science+Business Media GmbH

Preface

Overview

In this book we discuss the role of implicitly defined objects in computer graphics, within a coherent conceptual framework. We are concerned with the mathematical definition of shapes using an implicit form, as well as with its applications to geometric modeling, visualization and animation.

Organization and Features

The book is divided into five parts: mathematical foundations of geometric models, implicit formulations for the specification of shapes, implicit primitives, techniques for constructing and manipulating implicit objects, modeling, rendering and animation implicit objects.

Audience

The book is useful to researchers and graduate students in computer graphics and geometric modeling, and also to professionals in the fields of CAD/CAM and special effects.

Acknowledgments

We wish to thank our colleagues of the VISGRAF Laboratory for providing a great research enviroment that stimulated our work in the area of computer graphics. We also thank our students from IMPA, PUC-Rio, and UFRJ, that participated in many activities related to the development of this book.

Rio de Janeiro, November 2001

Luiz Velho
Jonas Gomes
Luiz Henrique Figureiredo

Contents

List of Figures

This page intentionally left blank

1. Introduction

The use of implicit formulations to describe the geometry of objects is not new in Computer Graphics. In fact, one of the first visualization systems developed in the sixties by MAGI [Mat68] was a ray tracing program based on quadric surfaces, which are specified implicitly by algebraic equations. Nonetheless, it was only recently that the graphics community began to recognize the specific aspects of this form of representation and to fully realize its potential applications [BBB+97].

1.1 Parametric versus Implicit

The geometry of an object can be formulated in two different ways: parametrically or implicitly. In the *parametric form*, the points belonging the object are given directly by a collection of mappings or parameterizations. These mappings relate a space of parameters to the object surface such that there is a correspondence between points in these two spaces. In the *implicit form*, the points belonging to the object are given indirectly through a point-membership classification function. This function defines the relation of points in the ambient space with the object. These two forms are in a sense complementary. It is important to notice that the parametric form is a *direct* description while the implicit form is an *indirect* one.

Intuitive understanding of the differences between the parametric and implicit forms is gained by means of a simple example:

Consider the unit circle in the plane (Figure 1.1). It can be described by the parametric equation $(x, y) = f(\theta)$, where:

$$f(\theta) = (\cos\theta, \sin\theta), \quad \theta \in [0, 2\pi].$$

This parameterization maps the interval $[0, 2\pi]$ of the real line onto the unit circle S^1. It also allows us to directly enumerate all the points of S^1, by varying the parameter θ from 0 to 2π.

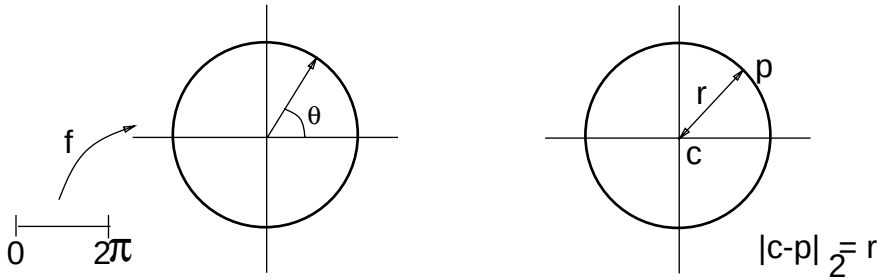


Fig. 1.1. Unit circle in parametric and implicit form

The unit circle can also be described by the implicit equation $F(x, y) = 0$, where:

$$F(x, y) = x^2 + y^2 - 1, \quad x, y \in \mathbb{R}.$$

The set of points (x, y) that satisfy $F(x, y) = 0$ is the circle S^1. The function F classifies the points in the plane with respect to the unit disk D^1, which is the region delimited by S^1. When we substitute the coordinates of a point $p = (x, y)$ in the equation $F(x, y) = x^2 + y^2 - 1$, the sign of the value of F at p indicates whether p is inside, outside, or on the circle, as follows: If $F(p)$ is negative, then p is in the interior of D^1; if $F(p)$ is positive, then p is in the exterior of D^1; and if $F(p)$ is zero, then p belongs to the boundary of D^1, that is, p is on S^1.

It is apparent that one form is better suited to some types of operations than the other, and vice-versa. For instance, to draw an approximation of the circle we need to connect by straight lines an ordered set of samples lying on S^1. This can be done easily using the parametric form stepping θ through $[0, 2\pi]$. On the other hand, to detect interferences between objects we need to test whether points from one object are inside the other. This is a straightforward operation using the implicit form: it amounts to simply checking the sign of $F(p)$.

1.2 Motivation

The parametric form has been by far the most popular geometric representation used in computer graphics and CAD to date. Parametric

objects and the techniques associated with them have been exhaustively investigated and developed. As a consequence, parametric objects are very well understood and are now part of the common vocabulary of most graphics practitioners. This can be verified when one notices that all graphics workstations have built-in engines to render parametric surfaces. Another indication is the large number of different types of spline surfaces, as well as the developments in the area of subdivision surfaces.

In contrast, the implicit form has been used only as a complementary geometric representation, mainly in the restricted context of specific applications. Recent developments in graphics are causing this situation to change and the community is beginning to draw its attention to implicit objects. This is reflected in the current research of aspects related to this subject.

There is another aspect of parametric and implicit representations that should be stressed. The equation that describes a point set $\mathcal{O}$ of the space defines the constraints that the points of the space must undergo in order to belong to the set $\mathcal{O}$. This is true both for implicit and parametric equations. Implicit equations arise naturally connected to several important problems in computer graphics, in particular in geometric modeling. Besides writing the equation that formulates the problem we must be able to develop "interrogation techniques" in order to solve it. Robust algorithms to deal with implicit surface interrogations have appeared in the literature.

The main motivation for writing this book is to review what has been done so far in this area and put it into perspective under a coherent conceptual framework.

1.3 Scope and Overview

The structure of the document is logically divided into five parts. The first part reviews the mathematical foundations of geometric models, including chapters on manifold models, parametric and implicit manifolds, space decompositions, and the realtion of shape and space. The second part concentrates on the specification of shapes using implicit formulations. including chapter on definition of implicit objects, as well as, operations to manipulate and combine implicit objects. The third part studies the main techniques used to compute with implicit

objects, including chapters on methods for approximating implicit objects. The fourth part describes the different types of implicit primitives, including chapters on analytical, procedural sample-based implicit objects, as well as, skeleton and multiscale representations. The fifth part considers the computer graphics techniques to model, render, and animate implicit objects. The book ends with a chapter discussing a high-dimensional implicit formulation of some graphics problems and algorithms to solve them.

2. Manifolds

This chapter reviews some of the main concepts involved in the description and computation of the geometric aspects of objects. These concepts are related to the mathematical study of shape and form, requiring results from Differential Geometry and Topology.

In order to simulate real objects using computers it is necessary to develop an abstract model that captures the relevant properties of the object. From this model, a concrete representation can be constructed describing particular instances of the model. This representation is made of symbols that are actually processed when we manipulate the object being simulated.

The concept of a manifold is an appropriate abstraction for describing the shape of a large class of objects. This mathematical entity makes it possible to study the geometric and topological properties of objects.[1] Most importantly, when we work with manifolds, all results are stated in a *coordinate-independent* manner, and most geometric notions are defined *intrinsically*.

2.1 Definition

Intuitively, a manifold is a topological space that is locally Euclidean. More precisely, a *n-dimensional manifold* is a separable topological space having at each point a neighborhood homeomorphic to an open subset of $\mathbb{R}^n$. The separability requirement means that the manifold is a Hausdorff space. Furthermore, we suppose that this space admits a countable base of open sets, that is, any open set on the manifold may be expressed as a countable union of open sets.

[1] The need for more generality motivated the development of non-manifold models. These are supported by the CSG and B-Rep representation schemes [Wei86] [RO89]. Nevertheless, these models generalize the previous one and, in general, can be defined by a collection of manifold structures.

Therefore, for each point $p \in M$, there exists an open set $U \ni p$, and a homeomorphism $\varphi : U \subset M \to \varphi(U) \subset \mathbb{R}^n$. This means that we have an open covering $\{U_i\}$ of M and each set in this covering is mapped onto an open subset of $\mathbb{R}^n$ by the map φ_i.

Figure 2.1 shows examples of subsets of the Euclidean space that are not manifolds.

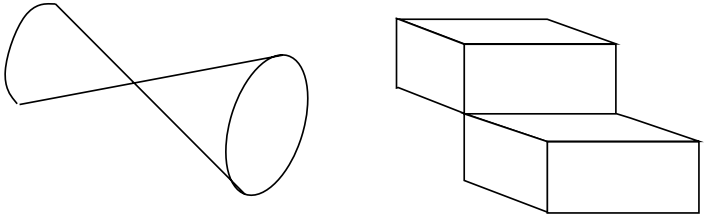


Fig. 2.1. Examples of non-manifold 3D objects.

2.2 Local Charts

The pair (U, φ) in the definition of a manifold is called a *local coordinate system*, or a *local chart*, where U is the domain of the chart. To each point $p \in U$, we assign the coordinates $(x_1, x_2, \ldots, x_n)$ of $\varphi(p) \in \mathbb{R}^n$.

Given two local coordinate systems $\varphi_1 : U_1 \ni p \to \mathbb{R}^n$ and $\varphi_2 : U_2 \ni p \to \mathbb{R}^n$, the map $h : \varphi_2^{-1} \circ \varphi_1$ is a homeomorphism with inverse $h^{-1} = \varphi_1^{-1} \circ \varphi_2$. (See Figure 2.2.) The homeomorphism h is called a *local change of coordinates*.

The local change of coordinates guarantees that the local pieces are nicely glued together. As a consequence, no matter how complex the manifold may be globally, we can regard the vicinity of each point as being a piece of an Euclidean space.

2.3 Atlases and Structures

Usually, it is not possible to find a single chart covering M entirely, in which case we use a collection of charts whose domains taken together cover M.

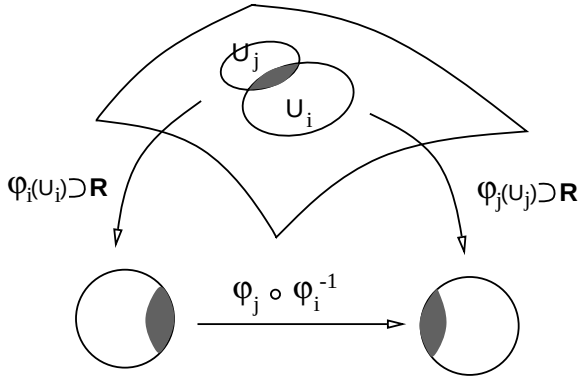


Fig. 2.2. Local change of coordinates

A collection $\{(U_i, \varphi_i)\}_{i \in I}$ of charts that covers M (where I is an index set) is called an *atlas*.

By imposing restrictions on the nature of the homeomorphism that defines the change of local coordinates we get a hierarchy of different types of atlases on the manifold:

h homeomorphism	Topological Manifold.
h C^k, $k \geq 1$, diffeomorphism	Differentiable C^k Manifold.
h analytical	Analytical Manifold.
$n = 2$, h holomorph	Riemann Surface.
h piecewise linear	Combinatorial Manifold.
h piecewise differentiable	Piecewise Differentiable Manifold.

An atlas $\mathcal{A} = \{\varphi_i, U_i\}$ is called *maximal* if it contains all local charts whose change of coordinates with φ_i are of the same type of $\mathcal{A}$. A maximal atlas on M defines a *structure* of the manifold. Note that for a given structure on a manifold M, there is no privileged coordinate system.

Each type of structure is suitable to study some class of problems, and is related to different fields of Mathematics. In computer graphics, we use mainly differentiable, piecewise differentiable or piecewise linear manifolds. This notion of piecewise linear manifolds will be used in Chapters 10 and 14. In this book, a differentiable manifold of class C^k, $k \geq 1$, will be called *smooth manifold*, or simply a *differentiable manifold*.

2.4 Calculus on Manifolds

If M is a C^k differentiable manifold it is possible to extend to M concepts from analysis on Euclidean spaces. The most delicate part is the definition of tangent space, since there is no ambient space involved. The definition of differentiability and derivative of a map are done straightforwardly using local coordinates. In other words, we can perform local computations on an n-dimensional manifold as if we were working in $\mathbb{R}^n$. Any function f defined on M can be expressed locally with the aid of the coordinates $x_1, x_2, \ldots, x_n$ defined by φ. At a point $p \in M$, we have

$$f(p) = f(x_1(p), x_2(p), \ldots, x_n(p)).$$

Since we can transport the differential calculus to manifolds, they become the natural habitat for problems from different areas of Pure and Applied Mathematics. As an example consider the study of motions of a double pendulum on a plane, composed of two articulated segments linked by a rotational joint and connected to a base by another rotational joint. Figure 2.3-a contains a diagram of this configuration. The position of the pendulum can be completely characterized by the rotation angles ϕ and θ of the two joints. This problem is properly formulated as a dynamical system defined on a manifold M, parameterized by θ and ϕ. The topology of M is the topology of a torus because the angles 0 and 2π are identified. This is shown in Figure 2.3-b.

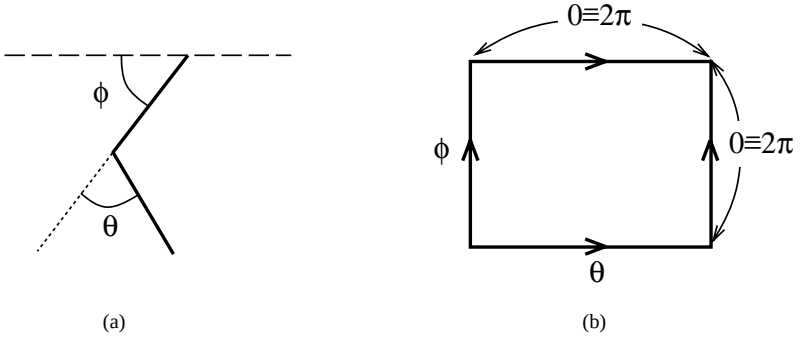


Fig. 2.3. Phase space of a double pendulum

It is also possible to define a metric on any differentiable manifold, if the metric is defined at each tangent space and varies in a differentiable way. This enables us to introduce geometric concepts on these spaces, generalizing the Euclidean Geometry of $\mathbb{R}^n$. The study of this Geometry begun with Riemann in the 19th century, and it is called *Riemannian Geometry*.

2.5 Immersions and Embeddings

Given two smooth manifolds M and N, a differentiable map $f : M \to N$ is an *immersion* if the derivative $f'(p)$ is $1-1$ for every $p \in M$. If $f : M \to N$ is an immersion and f is a homeomorphism $f : M \to f(M)$, $f(M)$ with the topology induced from N, then f is called an *embedding*.

Two conditions can prevent an immersion of being an embedding:

- f is not $1-1$ (See Figure 2.4-a).
- f is not a homeomorphism (See Figure 2.4-b).

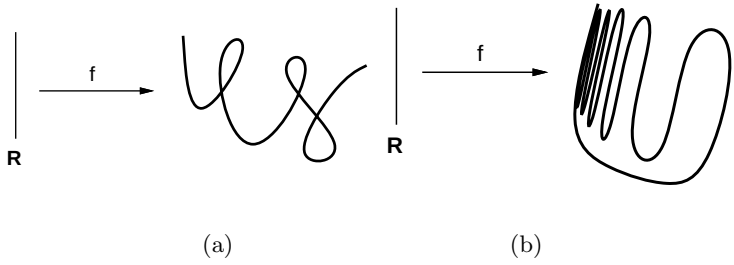


Fig. 2.4. Non-embedding conditions

An immersion is called in the literature *parametric manifold*. In dimensions 1 and 2, their use is widespread in computer graphics under the name of "parametric curves and surfaces". We will come to this subject in Chapter 3.

2.6 Submanifolds

As we have seen, a manifold is a very general and abstract mathematical concept. It has a structure of its own and exists independently of any other space.

For the purposes of modeling geometric objects we need a more restrictive concept. Real objects are part of the surrounding environment. This means that we have to regard them as residing in an n-dimensional ambient space. The proper mathematical concept in this case is that of a submanifold. This more specialized notion deals with a manifold S required to lie on another manifold M.

A *submanifold* S of another manifold M is a subset $S \subset M$ with a manifold structure, such that the inclusion map $i : S \to M$, $i(p) = p$, is an embedding. Most of the geometric objects used in computer graphics are submanifolds of the Euclidean space $\mathbb{R}^2$ or $\mathbb{R}^3$. Usually these submanifolds are called *regular curves* in the 1-dimensional case and *regular surfaces* in the 2-dimensional case [dC74]. If $S^n \subset M^m$ is an n-dimensional submanifold of an m-dimensional manifold M, with $n \leq m$, then the difference $k = m - n$ is called the *codimension* of S in M.

The definition of submanifold given here do not coincide exactly with the usual definition of submanifolds of $\mathbb{R}^n$ found in the literature. According to the traditional definition, a subset $M \subset \mathbb{R}^m$ is a n-dimensional submanifold of $\mathbb{R}^n$ if for each point $p \in M$ there exists an open neighborhood $U \ni p$ in $\mathbb{R}^m$, a neighborhood $V \subset \mathbb{R}^n$ and an embedding $\varphi : V \to U \cap M$. Note that the local charts take values in M. They are called a *parameterization* of the neighborhood $M \cap U$.

2.7 Realizations of a Manifold

It can be shown that any manifold can be embedded in a Euclidean space $\mathbb{R}^m$ of sufficiently high dimension m. For example, the product manifold $N = S^1 \times S^1$ can be embedded in $\mathbb{R}^3$ with a "doughnut" shape. Note however that this embedding does not inherit from $\mathbb{R}^3$ the "flat metric" naturally defined on N. In order to get the induced "flat metric" we must to embed it in $\mathbb{R}^4 = \mathbb{R}^2 \times \mathbb{R}^2$. This is depicted in Figure 2.5.

For Riemannian manifolds, i.e., manifolds with a metric, the embedding result stated above still holds, but the dimension of the embedding space is much higher.

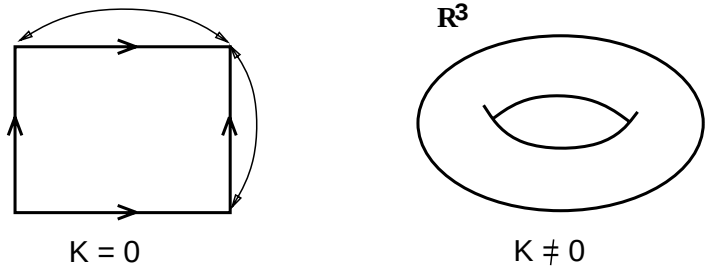


Fig. 2.5. Embedding $S^1 \times S^1$ in $\mathbb{R}^3$

2.8 Manifolds with Boundary

With the definition given above, the disk $\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\}$ is not a manifold: no neighborhood of a point (x, y) on its boundary, i.e., such that $x^2 + y^2 = 1$, is homeomorphic to an open set of $\mathbb{R}^2$. The class of objects represented by the disk, i.e., a surface plus its "boundary", is very common in computer graphics models. In order to consider them in a unified framework we have to extend the notion of a manifold to include boundaries.

In a manifold with boundary we must distinguish between interior and boundary points. This is done by parameterizing the manifold on the positive halfspace

$$\mathbb{R}^n_+ = \{(x_1, \ldots, x_n) \in \mathbb{R}^n : x_1 \geq 0\}$$

More precisely, a separable topological space M is a *manifold with boundary* if each point $p \in M$ has a neighborhood homeomorphic to an open subset of $\mathbb{R}^n_+$. A point $p \in M$ is called a *boundary point* if for any local chart X on M, $x(p)$ is on the boundary of $\mathbb{R}^n_+$ (i.e., its first coordinate is zero). The boundary of the manifold is the collection of all its boundary points (See Figure 2.6).

The definitions of charts, atlases, differentiability of a map, and the notion of submanifold can be extended to manifolds with boundary.

There is an important concept related to submanifolds, when the manifolds involved have boundary. A submanifold A of a submanifold with boundary M is called *neat* if $\partial A = A \cap \partial M$.

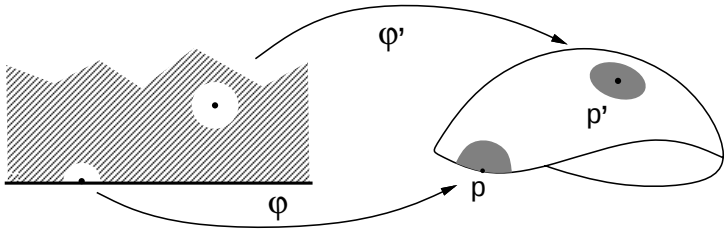


Fig. 2.6. Manifold with boundary

2.9 Orientability

A differentiable manifold is *orientable* if it admits an atlas $\mathcal{A}$ such that for every pair of overlapping charts (U, φ) and (V, ϑ), $U \cap V \neq \emptyset$, the associated coordinate systems (x^i) and (y^i) are consistently oriented (i.e., the Jacobian $\partial(y^1, \ldots, y^n)/\partial(x^1, \ldots, x^n)$ of the change of coordinates is positive). The atlas $\mathcal{A}$ is called an *orientation* of M.

The above definition means that for every $p \in M$, it is possible to orient the tangent plane T_pM in such a way that it is compatible with the canonical orientation of R^m under any local coordinate system of the atlas $\mathcal{A}$. This is illustrated in Figure 2.7.

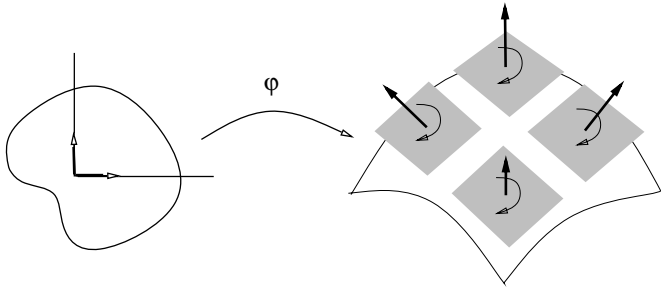


Fig. 2.7. Orientation on a manifold

That is the most natural and intuitive way to think about the orientability of a manifold. It is important to realize that orientability is a global concept. In an orientable manifold there exists an orientation of the tangent plane at each point and this orientation is globally compatible.

This definition holds for manifolds with or without boundary. If M is an orientable manifold with boundary, ∂M, and $\mathcal{A}$ is an orientation of M, then $\mathcal{A}$ defines in a natural way an orientation of ∂M, called the *induced* orientation.

This general definition of orientability is very hard to verify in practice. It is possible, under certain restrictions, to get better conditions for orientability. One such case is the study of orientability of submanifolds of the Euclidean space $\mathbb{R}^m$. If $M^n \subset \mathbb{R}^m$ is a submanifold of codimension $k = m - n > 0$, and $v_1, \ldots, v_k$ is a family of k normal, continuous, vector fields linearly independent at each point $p \in M$, then M is an orientable submanifold. When $k = 1$, the converse of the statement above is also true. Therefore, a surface in $\mathbb{R}^3$ is orientable if and only if there exists a non-null normal, continuous, vector field to it.

With the above result relating orientability and non-vanishing normal vector fields, it is very easy to prove that the Moebius band is not orientable.

We defined here the concept of orientability for differentiable manifolds. It is possible to extend this notion even to topological manifolds, but this needs results from algebraic topology.

2.10 Classification of Manifolds

A connected manifold without boundary of dimension 1 is homeomorphic either to the unit interval $(0, 1)$ or to the circle S^1. Therefore 1-dimensional manifolds are orientable.

The family of connected compact orientable manifolds of dimension 2, without boundary, consists of the sphere S^2 and the surfaces obtained by "attaching handles" to it. See Figure 2.8.

The boundary of a solid is an orientable manifold of dimension 2. Therefore, it is a disjoint union of the surfaces described above.

For manifolds of dimension greater than 2, there is no complete theory of classification.

2.11 A Suitable Model

We finally arrived at a suitable mathematical model to describe a large class of geometric objects in computer graphics: *curves* are one-dimensional submanifolds of $\mathbb{R}^2$ or $\mathbb{R}^3$, *surfaces* are two-dimensional

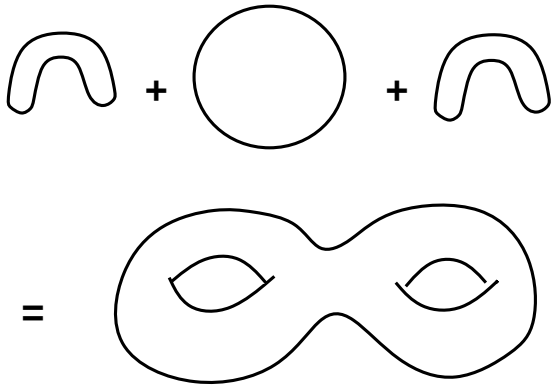


Fig. 2.8. Two-dimensional manifold of genus 2

orientable submanifolds of $\mathbb{R}^3$, and *solids* are three-dimensional orientable, compact, submanifolds of $\mathbb{R}^3$ with boundary. This allows us to represent in the computer many types of solids: solids with holes, solids with multiple shells, etc.

3. Parametric and Implicit Manifolds

A manifold is defined in parametric form if it is given by an atlas. This specification is not very attractive from a computational point of view because each chart is defined on an open set and they overlap. In this chapter we investigate different alternatives to represent a manifold.

3.1 Parametrizations

In Chapter 2 we showed that the natural way to describe a submanifold in $\mathbb{R}^n$ is by parameterizing a neighborhood of each of its points. The parameterization defines a coordinate system, the differentiability of the change of coordinates guarantees that adjacent pieces are glued nicely together. For each point on a surface in three dimensions, There exists an open set $V \ni p$ and an embedding $f : U \subset \mathbb{R}^2 \to f(U) = V$.

For each $(u, v) \in U$, $f(u, v) = (x, y, z)$ where

$$x = f_1(u, v),$$
$$y = f_2(u, v),$$
$$z = f_3(u, v).$$

(See Figure 3.1).

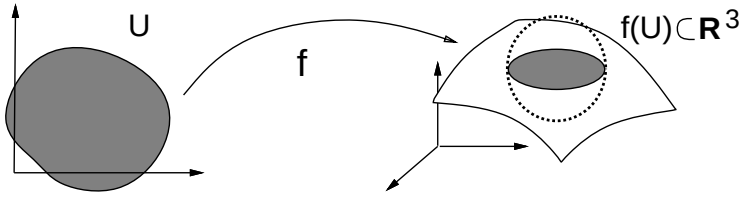


Fig. 3.1. Parametric Manifold

In general, it is impossible to cover the whole surface with a single parametrization. For compact manifolds without boundary, such as the sphere, we need at least two parametrizations.

A parametrization of the unit sphere by longitude and latitude is given by

$$f(\theta, \phi) = (\cos\theta \sin\phi, \sin\theta \sin\phi, \cos\phi),$$

where $f : U \to \mathbb{R}^3$, and $U = \{(\theta, \phi) \in \mathbb{R}^2 : 0 < \theta, \phi < \pi\}$. Note that the north and south poles are missing. If the points 0 and π are included in the domain to cover the poles, then f is no longer an immersion.

Of course, different parametrizations of the same surface always exist. As an example, the sphere can be also parameterized using the stereographic projection:

$$f(u, v) = (2u, 2v, u^2 + v^2 - 1)/(u^2 + v^2 + 1).$$

The parametrization f is a one-to-one mapping of $\mathbb{R}^2$ onto $S^2 - \{q\}$. Only the north pole q is missing, therefore two charts suffice to get an atlas of the sphere using these projections.

Definition: A manifold $M \subset \mathbb{R}^n$ is called *parametric* if it is the image of an immersion $f : U \subset \mathbb{R}^m \to \mathbb{R}^n$.

3.2 Stratifications

There is a need to introduce some decomposition scheme based on manifolds, such that each region has nice properties but its topology is not so restrictive. Also, the different pieces should have nice gluing properties in order to enable global constructions based on the decomposition. The intuitive idea of a stratification of a given set is a partition of the set into a number of manifolds that have nice gluing properties. Stratified sets constitute the natural habitat for the concept of manifolds with singularities.

A *stratification* of a subset V of $\mathbb{R}^n$ is a partition $\mathcal{P}$ of V into a family of smooth submanifolds of $\mathbb{R}^n$. Each connected decomposing submanifold is called an *strata* of the decomposition.

A triangulated manifold is an example of a cell decomposition that defines a stratification. The different faces of the simplices in the triangulation are the strata of the decomposition. (Chapter 4 discusses space decompositions in greater detail.)

In general the following additional assumptions are required for a stratification:

- local finiteness;
- boundary condition: the frontier of each strata is the union of lower dimensional strata;
- Whitney regularity. This guarantees topological homogeneity in the neighborhood of each point. The exact definition is more involved and is given in [CGV92].

In computer graphics we are mostly interested in finite stratifications. In this case, it has been shown that the regularity condition implies the boundary condition [Wal75].

There is a natural way to define a partial order on the family of stratifications associated to a given set V. For each stratification $\mathcal{P}$ we define the filtration (V^i)

$$\emptyset = V^0 \subset V^1 \subset \cdots \subset V^n = V$$

by taking V^i to be the union of the strata in $\mathcal{P}$ of dimension at most i.

Now, if $\mathcal{P}$, $\overline{\mathcal{P}}$ are two stratifications of V with filtrations (V^i) and $(\overline{V}^i)$, we define $\mathcal{P} < \overline{\mathcal{P}}$ if there exists an integer i for which $V^i \subset \overline{V}^i$, and $V^j = \overline{V}^j$, for all $j > i$. For finite stratifications this partial order has a least element. This element is called *minimal stratification* of the set V. This notion of minimality eliminates unnecessary subdivisions in the stratification, producing in this way a decomposition with the minimum number of strata.

Since each stratum is a smooth manifold, it is triangulable. Also, since different strata are glued in a well behaved way, it seems reasonable that we can glue the different triangulations in each stratum to obtain a triangulation of the whole stratified set. This result is true and was proved in [Joh83].

3.3 Piecewise Descriptions

We said that in general more than one parametrization is needed to cover a manifold. Since the parametrizations are defined on open subsets, given two local charts $\varphi_i : U_i \to V_i \subset M$, $i = 1, 2$, either V_1 and V_2 are disjoint or they overlap (i.e., the intersection is open). Computationally, this situation is not attractive. Ideally, we should cover M

by parametrized sets that do not overlap. The concept of stratification is adequate to define precisely good piecewise descriptions of a set.

Example 3.1. The decomposition of the sphere in Figure 3.2-a is a stratification, but not a cell decomposition. Figure 3.2-b shows a cell decomposition of the sphere.

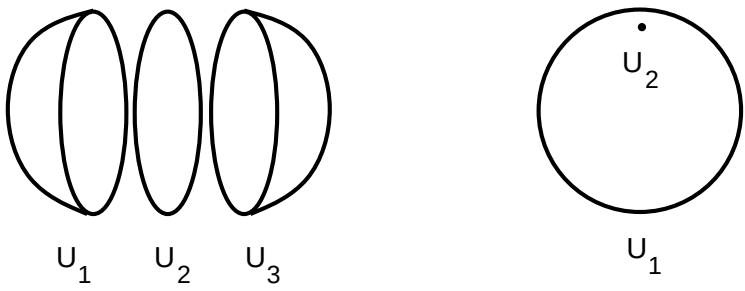


Fig. 3.2. Stratification and cell decomposition

A submanifold $M \subset \mathbb{R}^m$ of dimension n is called *piecewise parametric manifold* if the following conditions are satisfied:

- There exists a finite stratification, $M = \cup_{i=0}^{k} M_i$ of M;
- There exists a partition of a set $V \subset \mathbb{R}^n$, $V = \cup_{i=0}^{k} V_i$;
- There exists a family of parametrizations $\varphi_i : V_i \to M_i$.

Therefore, if M is a piecewise parametric manifold, we can define a global parametrization $\varphi : V \to M$, such that for each $V_i \subset V$, the restriction $\varphi|V_i$ of φ to V_i is the parametrization $\varphi_i : V_i \to M_i$. In applications we impose, in general, some regularity conditions on φ, e.g., continuity, class C^k, etc.

Example 3.2. The stratification in Example 3.1 can be used to define a piecewise parametric structure on the sphere by using stereographic projection to parameterize the two open hemispheres and the equator. In this case, the piecewise parametrization obtained has a C^∞ class.

Example 3.3. Parametric surfaces in computer graphics are usually constructed in a piecewise fashion. As an example, the classical "teapot" is defined by a cell decomposition where each cell is parametrized using a Bézier patch.

As we have seen, in general a manifold cannot be covered by just one chart (the sphere needs at least two charts). Nevertheless, every manifold has a piecewise parametric structure. This result follows from the theorem bellow.

Theorem 3.1. *Every differentiable manifold is triangulable.*

The first proof of the above theorem was given by Cairns [Cai34]. A more geometric proof can be seen in [Whi57]. What the above theorem shows is that every differentiable manifold has a combinatorial structure. Later on, we will study computational methods to get combinatorial triangulated structures that approximate a class of manifolds.

3.4 Implicit Description

Intuitively, implicit manifolds are manifolds defined as the solutions of equations of the type $F(x) = 0$, when F is a C^k real function on the Euclidean space. The set of points that satisfy such an equation may or may not be a manifold. For instance, the equation $x_1^2+x_2^2-1 = 0$, as we have seen, defines the unit circle, which is a 1-dimensional manifold. On the other hand, the equation $x_1^2 - x_2^2 = 0$ defines a pair of intersecting lines, which is not a manifold. Therefore, we need to state precisely under which conditions an implicit form defines a manifold. This allows us to derive several properties associated with this class of manifolds.

3.5 Regularity and Transversality

Let M and N be differentiable manifolds, and $f : M \to N$ be a differentiable map between them. The inverse image of a point $c \in N$ by f is the set

$$f^{-1}(c) = \{p \in M : f(p) = c\}.$$

This is illustrated in Figure 3.3.

Note that if $M = \mathbb{R}^m$ and $N = \mathbb{R}$, the set $f^{-1}(c)$ is precisely the solutions of the implicit equation

$$f(x_1, \ldots, x_m) - c = 0$$

The point c is called a *regular value* of f if for all $p \in f^{-1}(c)$, the differential $f'(p) : T_pM \to T_cN$ is surjective. If c is not a regular value,

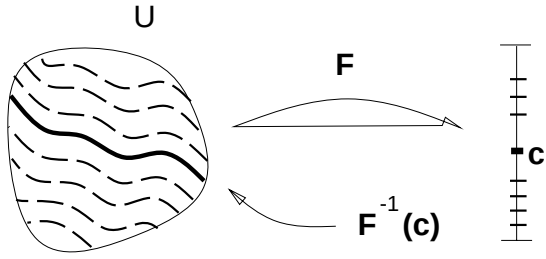


Fig. 3.3. Level Curves

it is called a *critical value*. When $M = \mathbb{R}^m$ and $N = \mathbb{R}$, the regularity condition means that the gradient vector

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_m} \right)$$

does not vanishes at the points of $f^{-1}(c)$. When $N = \mathbb{R}^n$, and f has coordinate functions $f = (f_1, \ldots, f_n)$, the regularity condition means that the gradient vector fields $\nabla f_1, \ldots, \nabla f_n$ are linearly independent.

In order to define a great variety of objects in computer graphics, we should allow the inverse image by f of subsets with more than one point. A proper generalization is to take a submanifold A of N instead of a single point p. In this case, the concept of regular value should be interpreted as a transversality condition: f is *transversal* to a submanifold A of N if for every point p on the inverse image $f^{-1}(A)$ we have

$$f'(p) \cdot T_p M \oplus T_{f(p)} A = T_{f(p)} N.$$

That is, the tangent space of N at $f(p)$ is spanned by the tangent space of A at $f(p)$ and the image of the tangent space of M at p by the derivative of f. This means that the derivative at each point of $f^{-1}(A)$ cannot degenerate too much.

Observe that if the submanifold A reduces to a point c, the concept of transversality reduces to the concept of regular value.

3.6 Implicit Manifolds

The properties defined above give us several conditions under which the inverse image of a submanifold is a manifold. The results of the

theorem below states in which cases a manifold is called an *implicit manifold*.

Theorem 3.2. *Let $f : M \to N$ be a differentiable map between differentiable manifolds. If $c \in N$ is a regular value of f, then $f^{-1}(c)$ is a submanifold of M whose codimension is equal to the dimension of N.*

Theorem 3.3. *Let M and N be differentiable manifolds, N a manifold without boundary, and $A \subset N$ a submanifold (possibly with boundary). If $f : M \to N$ is transversal to A, then the inverse image $f^{-1}(A)$ is a submanifold (possibly with boundary) of M, such that* $\operatorname{codim}(f^{-1}(A)) = \operatorname{codim}(A)$.

Theorem 2 above is also true if the manifold N has boundary, but in this case we have to add one of the conditions bellow:

- $A \subset N - \partial N$, or
- A is neat, or
- $A \subset \partial N$.

In addition, f has to be transversal to ∂A when appropriate (last two conditions).

Example 3.4. Let $f : \mathbb{R}^3 \to \mathbb{R}$; be given by $f(x, y, z) = x^2+y^2+z^2-1$. Then, $f^{-1}((-\infty, 1])$ is the closed ball of radius one. $f^{-1}([\frac{1}{2}, 1])$ is the region delimited by the sphere of radius $\frac{1}{2}$ and 1. The transversality condition is immediately verified here.

Let $f : U \subset \mathbb{R}^{n+k} \to \mathbb{R}^k$ be differentiable and $c \in \mathbb{R}^k$ a regular value of f. If $f_1, \ldots, f_k$ are the coordinate functions of f, then is easy to verify that the vector fields

$$\operatorname{grad} f_1(p), \ldots, \operatorname{grad} f_1(p)$$

are normal to the implicit submanifold $M = f^{-1}(c)$.

It follows from this observation that every manifold defined implicitly as the inverse image of a regular value is orientable.

3.7 Geometric Interpretation

An implicitly defined manifold of codimension 1 can be interpreted geometrically as the level set of the graph of a function. This provides a very good intuition of the real meaning behind the implicit form.

Consider a function $F : U \subset \mathbb{R}^n \to \mathbb{R}$. Its *graph* is the subset of $\mathbb{R}^{n+1}$ defined by

$$\text{graph}(F) = \{(x_1, \ldots, x_{n+1}) : (x_1, \ldots, x_n) \in U, x_{n+1} = F(x_1, \ldots, x_n)\}$$

This can be visualized as a height field where for each point $p \in U$ the value of $F(p)$ gives the elevation of a hypersurface G at that point. This type of surface is also called a *Monge surface*.

The manifold M defined by $F^{-1}(c)$ is given by the intersection of the graph of F with a hyperplane parallel to U at a distance c from it: $G \cap \{x_{n+1} = c\}$.

This is shown in Figure 3.4 for a circle defined implicitly by $x_1^2 + x_2^2 = c$. We use a two-dimensional example for clarity. The graph of F is a paraboloid of equation $x_3 = x_1^2 + x_2^2$, and its intersection with a plane $x_3 = c$ is a circle for $c > 0$. Note that $c = 0$ is not a regular value because it is a minimum of F and therefore a singular point.

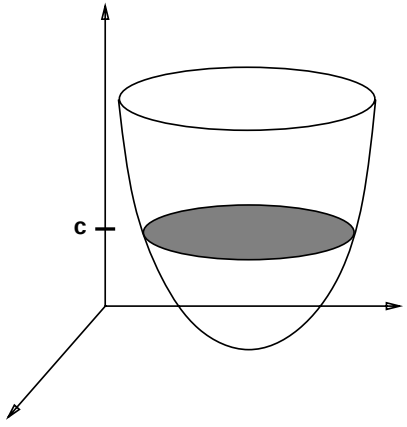


Fig. 3.4. Implicit surface as a level set

In general, we can say that the description of objects in implicit form is intimately related to the problem of intersection. In the particular case of codimension 1, the implicit manifold $F(p) = c$ is the intersection of the graph of F with a hyperplane. For codimension m, the implicit manifold $F(p) = c = (c_1, \ldots, c_m) \in \mathbb{R}^m$ is the intersection of the m hypersurfaces $F(p) = c_1, \ldots, F(p) = c_m$ in $\mathbb{R}^n$.

The regularity condition in the geometric interpretation above means that the manifolds have no tangency along their intersection set.

3.8 Algebraic Varieties

Consider $M = \mathbb{R}^m$, and take a map $F : \mathbb{R}^m \to \mathbb{R}$ with coordinates $F = (F_1, \ldots, F_n)$, such that each coordinate function $F_l : \mathbb{R}^m \to \mathbb{R}$ is a polynomial of degree g in m variables; that is

$$F_l(x_1, \ldots, x_m) = \sum_{i=1}^{n} a_i x_1^{k_{i1}} \ldots x_m^{k_{im}}$$

where $a_i \in \mathbb{R}$, and k_{ij} are non-negative integers. Each term $a_i x_i^{k_{i1}} \ldots x_m^{k_{im}}$ is called a *monomial*. The degree of the monomial is the sum $k_{i1} + \cdots + k_{im}$ of the exponents. The *degree* of F is the highest degree of its constituent monomials with leading term $a_i \neq 0$. The implicit object defined by $F(x_1, \ldots, x_m) = 0$ is called an *algebraic variety*. The degree of the polynomial F is the degree of the variety. We will also call an algebraic variety an *algebraic implicit surface*.

Note that an algebraic variety is not necessarily a manifold since we do not require 0 to be a regular value.

Algebraic varieties constitute a very special class of implicit objects, because we can use techniques from algebraic geometry to study them.

3.9 Algebraic Interpretation

The possibility of associating some algebraic structure to geometric objects makes it possible to use algebraic, symbolic computations in order to solve geometric problems. Suppose M is an algebraic variety, that is, the zero set of the polynomial f. If $g : \mathbb{R}^m \to \mathbb{R}$ is any polynomial, then M is a subset of the zero set of the polynomial $f \cdot g$. In this way, we can characterize M as the zero set of all polynomial multiples of f. This set of polynomials is called the *ideal* generated by f, and is denoted by $\langle f \rangle$.

Considering $F : \mathbb{R}^m \to \mathbb{R}^n$, with coordinate functions $(f_1, \ldots, f_n)$, the same result is true for the ideal $\langle f_1, \ldots, f_n \rangle$ generated by the coordinate functions. This ideal is the set of all polynomials

$$g_1 f_1 + \cdots + g_n f_n,$$

where g_i is an arbitrary polynomial.

Ideals constitute a very nice algebraic structure: they are closed under sums and products. Moreover, the product of any polynomial by a polynomial from an ideal belongs to the ideal.

The association of an implicit algebraic surface with an ideal makes it possible to use the techniques from algebraic geometry in this area. The reader interested in more details should consult chapter 8 of [Hof89] and the references there.

3.10 Parametric versus Implicit

The problem of converting a parametric manifold to an implicit form is called *implicitization*. Consider M a parametric manifold in $\mathbb{R}^n$, that is, M is a submanifold of $\mathbb{R}^n$, and $\varphi : U \to M$ is a global parametrization of M. To implicitize M, we must find a function $F : \mathbb{R}^n \to \mathbb{R}^m$, such that $M = F^{-1}(C)$.

Note that the function F in the definition above is defined on the same space where the manifold M lives. Without this dimension constraint, the implicitization problem becomes trivial. Without the dimension constraint, every parametric manifold can be implicitized easily. In fact, suppose

$$x_1 = f_1(u, v), \quad x_2 = f_2(u, v), \quad x_3 = f_3(u, v),$$

define a parametric surface in $\mathbb{R}^3$, $f_i : U \subset \mathbb{R}^2 \to \mathbb{R}^3$. The equation $h = 0$, where the map $h : U \times \mathbb{R}^3 \subset \mathbb{R}^5 \to \mathbb{R}^3$ is defined by

$$h(u, v, x_1, x_2, x_3) = (x_1 - f_1(u, v), x_2 - f_2(u, v), x_3 - f_3(u, v)),$$

defines implicitly an embedding of M in $\mathbb{R}^5$. Implicitization without dimension constraints is very well exploited in the "dimensionality paradigm" introduced by Hoffmann. For more details the reader should consult [HV89], [Hof90] and also Chapter 17.

In general, if M is a parametric manifold, there is no implicitization for M. An easy way to see this is to observe that an implicit manifold is a closed subset of the ambient space, and this is not necessarily true for an arbitrary parametric manifold. The reader interested in a

more complete discussion about the problem of conversion between implicit and parametric manifolds, with several examples, should consult [Hof89, Sed90a, Sed90b], and the references there.

Locally, the concepts of parametric and implicit manifolds coincide. This means that if M is an m-dimensional manifold in $\mathbb{R}^n$, then the following assertions are true:

1. For every $p \in M$ there exists a neighborhood $M \supset V \ni p$, and a diffeomorphism $\varphi : U \subset \mathbb{R}^m \to V$.
2. For every $p \in M$ there exists a neighborhood $U \subset \mathbb{R}^n$, $p \in U$ and a function $f : U \to \mathbb{R}^{n-m}$ such that $M \cap U = f^{-1}(0)$, and 0 is a regular value of f.

Assertion (1) is the very definition of a manifold. Assertion (2) is an immediate consequence of the implicit function theorem [Spi65].

It is possible to improve the result in assertion (2) even more:

2′. For every $p \in M$, there exists a decomposition $\mathbb{R}^n = \mathbb{R}^m \oplus \mathbb{R}^k$, such that M is the graph of a function $f : U \subset \mathbb{R}^m \to \mathbb{R}^k$.

The local results stated above have been exploited in the literature in order to obtain computational methods for surface interrogations. In [MTV86] approximations of the kind stated in (1) are constructed (see also [BI89]). Chuang [CH89] describes a technique to construct a local implicit approximation to a parametric curve or surface. In both cases, applications are discussed.

Every closed submanifold of $\mathbb{R}^n$ can be defined implicitly. An elementary proof of the result for compact surfaces in $\mathbb{R}^3$ can be found in [dC74].

It is not difficult to find an implicit equation for every surface of genus g. For the sphere ($g = 0$) there is no problem. For $g \geq 1$, we take an implicit equation $F(x, y) = 0$ that defines a compact curve γ in $\mathbb{R}^2$ with $g - 1$ crossings. The implicit equation for the genus-g surface is given by

$$F(x, y)^2 + z^2 - \varepsilon^2 = 0.$$

Geometrically, the surface consists of the points in the space whose distance from the curve is some fixed number $\varepsilon > 0$. Of course, ε depends on the curve $F(x, y) = 0$.

For the torus we can take the curve γ to be the circle of radius 2, $F(x, y) = x^2 + y^2 - 1$, and $\varepsilon = 1$. The surface equation becomes

$$(x^2 + y^2 - 4)^2 + z^2 - 1 = 0.$$

For the 2-torus we can define the curve by $F(x, y) = 4x^2(1 - x^2) - y^2 = 0$, and $\varepsilon = 1/4$. The surface equation becomes

$$(4x^2(1 - x^2) - y^2)^2 + z^2 - \frac{1}{4} = 0.$$

Later on, we will describe in detail computational methods to construct a local parametrization to a given implicit surface.

4. Space Decompositions

This chapter presents an overview of the concepts related to space decompositions and discusses the data structures for their representation. The existence of certain subdivisions of a space allows us to obtain valuable information about that space, as shown in Chapter 3. This will play a very important role in the representation and computation with implicit objects. A more extensive treatment of the subject in the context of graphical applications is given in [CGV92].

Space decompositions are used in several ways in applications. They may serve to represent geometric objects exactly or approximately. They may also serve as an auxiliary representation to facilitate computation of geometric data. When the space decomposition is used to represent the geometric object itself it is called *object-based*. When the space decomposition is used as an auxiliary representation it is called *space-based*.

4.1 Types of Space Decompositions

The intuitive idea of a space decomposition is to subdivide the space into a collection of disjoint connected subsets. Subdivision of the space into simpler pieces, along with a structure that links these pieces together, allows us to obtain precise information about the geometry and topology of the space. This strategy is related to the "divide and conquer" paradigm. We get smaller, easier-to-understand pieces of the space, and structure them together in order to get information about the space as a whole. There is a trade-off involved when obtaining a decomposition of a given space: a more structured decomposition certainly will give us more information about the space, but it is harder to construct and may not even exist in general.

4.1.1 Space Partitions

A *space partition* of a set U is a collection $(U_\alpha)_{\alpha \in I}$, of subsets of U such that

- $\bigcup U_\alpha = U$;
- $U_\alpha \cap U_\beta = \emptyset$ if $\alpha \neq \beta$.

Every set has a trivial space partition, consisting of the collection of all of its points. In fact it is easy to see that, for any infinite set U, there always exists an infinite number of finite partitions. Space partitions are also called in the literature *space decompositions*.

A natural way to define space partitions is by using an equivalence relation defined on the points of the set U. If R is an equivalence relation, for each point $p \in U$ we define its *equivalence class* $U_p = \{q \in U : (p, q) \in R\}$. It is clear that $(U_p)_{p \in U}$ define a partition of the set U. Conversely, every partition of the space induces in an obvious way an equivalence relation by defining two elements to be equivalent if they are in the same set of the partition.

A partition is the most general decomposition scheme that can be used to subdivide a space. All decomposition schemes studied below are partitions with some additional structure, which may impose requirements on the geometry or topology of each set of the partition or on the relationships among these sets. This additional structure enables us to represent geometrical and topological properties of the underlying space.

4.1.2 Cell Decompositions

We now discuss a family of finite space decompositions of a set U, in which further structuring is imposed. Each set of the partition is now required to be a k-dimensional cell, that is, a set which is homeomorphic to an open disk of $\mathbb{R}^k$. Furthermore, we require the boundary of each cell to be a (finite) union of lower-dimensional cells.

More precisely, a *cell complex* is a finite collection of subsets c_j^q (where $q = 0, 1, 2, \ldots, d$ represents the dimension of the cell and j ranges over some index set J_q) such that:

- each c_j^q is homeomorphic to the open q-dimensional disk for $q > 0$ and is a single point if $q = 0$.

- For each $q = 0, 1, 2, \ldots, d$ and each j in J_q, the boundary of c_j^q is equal to the union of all lower dimensional cells that intersect that boundary.

4.1.3 Affine Cell Decompositions

Affine cell decompositions are examples of special cases of cell decompositions obtained by restricting the geometry of the cells. For this decomposition scheme, one requires each cell to be a convex polytope.

An *affine cell decomposition* is a cell decomposition such that every cell is affine. A cell in $\mathbb{R}^n$ is called *affine* if it is implicitly defined by the equations

$$L_i(x) \leq b_i, \quad i = 1, \ldots, m,$$

where each L_i is a linear function $L_i : \mathbb{R}^n \to \mathbb{R}$.

Moreover, in order to avoid unnecessary fragmentation, each face of a cell (that is, a set obtained by turning some of the defining inequalities into equalities) must also be a cell. Note that, this concept generalizes to $\mathbb{R}^n$ the concept of a convex polygon on the plane.

The very special cell geometry present in this type of decomposition allows one to design very efficient algorithms to deal with it. On the other hand, due to its restrictive geometric nature, exact affine cell decompositions are not generally available for a given set. However, important families of subsets of the euclidean space can at least be approximated by an affine cell complex, which is sufficient for a huge number of applications.

Although the conditions imposed on the definition of affine cell decompositions concern mainly the geometry of the cells, they also imply additional combinatorial structure. In particular, the boundary of a d-dimensional cell has dimension $d - 1$. As a consequence, affine cell decompositions have cells of all dimensions from 0 to the maximum dimension d. This is not true for arbitrary cell complexes. For instance, if p is a point on a 2-sphere S, then $\{p\}$ and $S - \{p\}$ determine a cell decomposition of S in which there are only cells of dimension 0 and 2.

4.1.4 Simplicial Decompositions

Simplicial decompositions are special cases of affine cell decompositions in which cell geometry is as simple as possible: each cell is a (relatively) open *simplex*. Given $d + 1$ points v_0, v_1, $\ldots$, v_d not belonging to the

same d-dimensional hyperplane, the set $\{\sum_{i=0,\ldots,d} x_i v_i : 0 < x_i < 1\}$ is an open d-dimensional simplex. Furthermore, single points are considered to be 0-dimensional open simplices.

Every affine cell decomposition can be turned into a simplicial decomposition by using a refinement operation that consists in triangulating in a standard way each of its convex affine cells [Mun66]. Thus, the sets that admit an affine cell decomposition are exactly those that have a simplicial decomposition. There is a natural trade-off between the two types of decomposition. Simplicial decompositions are particularly easy to represent due to the fact that each cell of a given dimension has a fixed number of lower-dimensional bounding cells. On the other hand, general affine decompositions are more concise.

A *simplicial complex* $\mathcal{T}$ over a domain $D \in \mathbb{R}^n$ is a family of simplices with the following properties:

1. $D = \cup_{\sigma \in \mathcal{T}}$
2. $\sigma_1 \cap \sigma_2$ is either empty or a common face (lower dimensional simplex) of both simplices.
3. If D is a compact subset of $\mathbb{R}^n$ then it intersects only finitely many simplices.

Triangulations

A triangulation of a subset $U \in \mathbb{R}^n$ is a homeomorphism $h : \mathcal{T} \to U$ from some simplicial complex C to U. The complex C induces a cell decomposition on U, which is called a *triangulation* of U. A subset U of the Euclidean space is *triangulable* if it admits this homeomorphism. Triangulable sets are also called *topological polyhedra*.

There are many possible triangulations of the space $\mathbb{R}^n$. Some important types of triangulations are the *Coxeter-Freudenthal* triangulation K_1 and the J_1 triangulation of Todd [AG90]. Figure 4.1 shows these triangulations in two dimensions.

It is desirable to have a small number of cell types, e.g., *congruent cells*, which differ only by orientation or reflection. If all the cells are identical, computations are very simple. It can be shown, [Cox63], that in three dimensions the only type of cell that fills space is the *cube*.

A hypercube in $\mathbb{R}^n$ is the cartesian product of n non-degenerate intervals:

$$\prod_{i=1}^{n} [a_i, b_i]$$

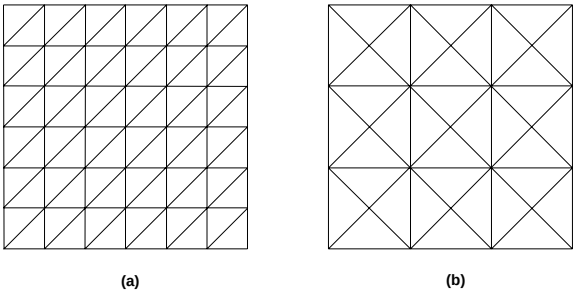


Fig. 4.1. Coxeter-Freudenthal and J_1 triangulations

It is very easy to obtain a cell decomposition of $\mathbb{R}^n$, where each cell is a hypercube of appropriate dimension. Therefore, an easy way to obtain a triangulation of $\mathbb{R}^n$ is to use a triangulation of the hypercube. A classical triangulation obtained with this method is the *Coxeter-Freudenthal triangulation*.

This triangulation can be defined as follows: For the square (the hypercube in $\mathbb{R}^2$), we take its diagonal, and the triangulation obtained has two simplices of dimension 2. For the cube in $\mathbb{R}^3$ with vertices $p_0, \ldots, p_7$, we take the diagonal p_0p_7 and project it onto each face of the cube. We obtain in this way the Coxeter-Freudenthal triangulation of the faces. The simplices of dimension 3 of the triangulation of the cube are constructed by adding to each 2-simplex ρ of the cube's faces the vertex of the diagonal p_0p_7 that does not belong to ρ. (See Figure 4.2).

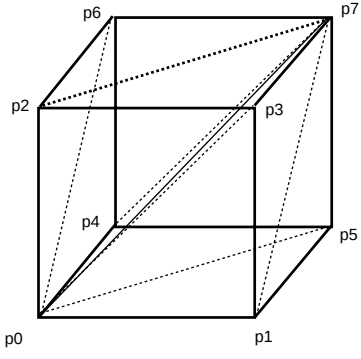


Fig. 4.2. Triangulation of the cube in $\mathbb{R}^3$

We obtain in this way 6 simplices of dimension 3.

$$\rho_0 = (p_0, p_1, p_3, p_7)$$
$$\rho_1 = (p_0, p_1, p_5, p_7)$$
$$\rho_2 = (p_0, p_2, p_3, p_7)$$
$$\rho_3 = (p_0, p_2, p_6, p_7)$$
$$\rho_4 = (p_0, p_4, p_5, p_7)$$
$$\rho_5 = (p_0, p_4, p_6, p_7)$$

It is not too difficult to see that the Coxeter-Freudenthal triangulation of the hypercube in $\mathbb{R}^n$ has $n!$ simplices of dimension n [1].

It is also important to observe that not all triangulations of $\mathbb{R}^3$ are obtained by replicating a triangulation of the cube. An example where this does not occur is in the already mentioned triangulation of Todd, shown in Figure 4.1 for the two-dimensional case.

In three dimensions another interesting triangulation is based on the so-called *cubic* tetrahedra, which are formed by slicing off the corners of a cube [HW90].

4.2 Properties of Space Decompositions

Depending on the application, there are some properties associated with space decomposition schemes that may be very important to impose. We summarize these properties below.

4.2.1 Invariance

Invariance properties associated to space partitions are very important. Different types of invariance are possible. Of great interest are invariance under topological operations (e.g., closure), set operations (e.g. intersection), and invariance under a certain class of transformations of the euclidean space (e.g., rigid motions), that is, if F is such a transformation and $\mathcal{C}(X)$ is a decomposition of a set X, then $F(\mathcal{C}(X)) = \mathcal{C}(F(X))$.

[1] This is not a problem if we are working in 3-space, but this combinatorial explosion can be a serious issue if we need to work in higher dimensional spaces.

4.2.2 Uniqueness

In general, space decompositions of a certain type are not unique. However, in many cases there exists a natural way to define the concept of equivalence between two decomposition schemes, and it is desirable to define uniqueness up to equivalence (e.g., uniqueness up to rigid motions).

4.2.3 Minimality

If it is possible to define some order relation on a decomposition scheme, then we can define a "minimal decomposition scheme". That is, one that has the minimum number of decomposing regions. The existence of a minimal decomposition might imply uniqueness.

4.2.4 Finiteness and Local Finiteness

A decomposition is *finite* if it has a finite number of decomposing regions.

A decomposition is *locally finite* if each point has a neighborhood that intersects only a finite number of decomposing regions.

4.2.5 Regularity

Regularity conditions come in many different flavors. In general, they are imposed when we need to have some nice topology across the boundaries of the decomposing regions, and some homogeneity of the topology of each decomposing region.

4.2.6 Boundary Condition

Boundary conditions are imposed in order to get some hierarchy of the space partition. Both regularity and boundary conditions are related to the desire to have nice gluing properties between the different regions of the partition.

4.2.7 Refinement

The *refinement* is an important unary operation, $\mathcal{R}\colon \mathcal{C}(V) \to \mathcal{C}(V)$, defined on a family $\mathcal{C}(V)$ of all space decompositions of a given set V.

This operation acts by subdividing each decomposing region of some element in $\mathcal{C}(V)$, in order to obtain another decomposition of V. The refinement objective is twofold: to obtain a better geometric behavior of each decomposing region in the new partition of V, and to get a better adjacency relationship between the subdividing regions of the new partition. The refinement operation $\mathcal{R}$ in general defines a partial order $\succeq$ such that $\mathcal{P}_V \succeq \mathcal{R}(\mathcal{P}_V)$ for all partitions $\mathcal{P}_V$ of the set V.

4.3 Algebraic Structure of Space Decompositions

If (U_i), $i = 1, \ldots, m$, is a finite partition of the euclidean space $\mathbb{R}^n$, consider the set $\mathcal{L}$ consisting of the finite unions of the sets U_i in the partition. It is easy to see that $\mathcal{L}$ is a finite Boolean algebra under the usual set operations $\cup$, $\cap$, and complement, $-$. We say that the Boolean algebra $\langle \mathcal{L}, \cup, \cap, - \rangle$ is generated by the partition (U_i) of $\mathbb{R}^n$, and has 2^m elements.

Different space decompositions generate different Boolean algebras. These algebras enable us to associate arithmetic expressions to space decompositions introducing in this way a constructive approach to space decomposition. A good overview of a hierarchy of algebras of $\mathbb{R}^n$ can be found in [Sha91].

4.4 Spatial Data Structures

The spatial decompositions defined in the previous section possess natural graphs associated to them. They describe the adjacency relations between the several elements of the decomposition. Spatial data structures represent these graphs. They are used to structure interesting subsets of geometric objects and to provide a means for operating on them.

There are two ways to describe a region in a space decomposition: either by the point set that the region represents, or by the boundary that defines the region.

Some space decompositions are so simple and structured that it is not necessary to specify their regions explicitly. They are defined indirectly through a *canonical model*. This can be seen in some tessellations of space where the space subdivision is composed by the repetition of a single type of cell.

Spatial data structures can be divided in two main classes: *flat* and *hierarchical*. The former consists of an enumeration of cells, while the latter is defined by a recursive decomposition of cells.

Spatial data structures are designed to encode both the geometry and the topology of space decompositions. Below we analyze these data structures from the most general to the most specific type.

The most general data structure that can be used to represent space partitions is a *list of cells*. This cannot be really considered a spatial data structure because it does not encode any topological or hierarchical information. It is just a way to enumerate a set of cells. Note that this is the only type of data structure that can be used to describe arbitrary partitions of space.

4.4.1 Topological Graphs

The basic data structure that represent space decompositions is a topological graph in which the nodes of the graph contain information about the geometry of each decomposing region and the links of the graph give topological information reflecting adjacency relationships between regions. This type of spatial data structure is suitable to describe a large number of space decomposition schemes.

The fundamental topological information in a space decomposition is the association between a cell and its boundary elements. Note that this is a binary relation which is sufficient to completely specify the topology of a space decomposition. Since the boundary of a decomposing region is composed of geometrical elements of dimension lower than the dimension of the region, the graph in our data structure can be layered in a hierarchical manner such that each layer contains only regions of the same dimension. The links of the graph representing incidence relations occur only between layers. Figure 4.3 shows a diagram of a topological graph.

4.4.2 Trees

Trees encode the hierarchical geometry of nested subdivisions of space. They consist of a set of nodes linked recursively starting from the top (or root) node. In this scheme we say that parent nodes are linked to children nodes. Terminal nodes are the leaves of the tree and do not have links. Each parent node corresponds to a cell which is subdi-

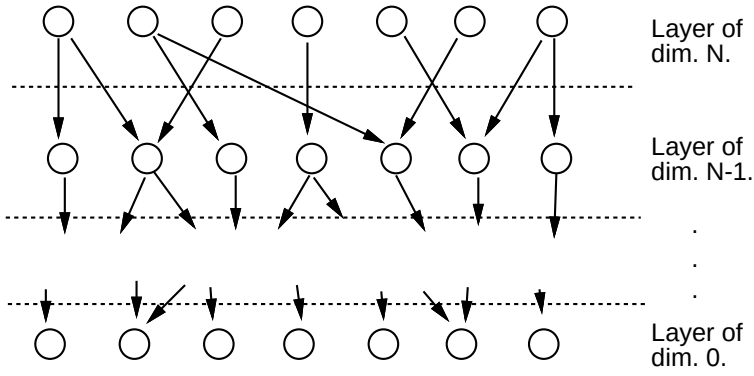


Fig. 4.3. Topological graph

vided into siblings according to some prescribed rule. For example, a n-dimensional cube can be subdivided into 2^n cubes.

The tree structures are classified according to the type of cell subdivision performed at the nodes:

N-trees (Quadtrees, Octrees) – Cells are subdivided n subcells by hyperplanes at regular intervals that are aligned with the axis of the reference frame.

K-d trees (K-dimensional trees) – Cells are subdivided in two subcells by a hyperplane arbitrarily positioned and aligned with each axis of the reference frame cyclically in succession (i.e., $x_1, x_2, \ldots, x_k$).

BSP-trees (Binary Space Partition trees) – Cells are subdivided in two subcells by a hyperplane arbitrarily positioned and oriented.

Restricted trees are trees that correspond to a balanced partition of space. In this structure adjacent cells differ at most by a factor of 2 in terms of the level of refinement. This type of tree is important in adaptive subdivision schemes [vHB87].

All trees can be reduced to binary trees that are constructed by recursively subdividing n-dimensional space into two regions by a $(n-1)$-dimensional hyperplane.

4.4.3 *N*-dimensional Arrays

N-dimensional arrays reflect the topology of regular packings of n-dimensional cells. They consist of a set of records specifying: the dimension of the array and the number of elements in each dimension,

followed by the sequence of elements in a prescribed order. Arbitrary elements can be accessed directly through an n-tuple of indices corresponding respectively to each spatial dimension. Note that the spatial location of each cell relative to a frame of reference can be derived from the index of the element in the array and vice versa.

Arrays are flat data structures in which the cell geometry is given by a canonical model. This data structure tends to be quite large, because all the elements have to be stored.

It is interesting to note that n-dimensional arrays produce uniform subdivision that can be also represented by N-trees in which the leaves are all at the same depth.

This page intentionally left blank

5. Shape and Space

This chapter investigates the relations between the shape of a solid object and the ambient space in which it is embedded. For this we will discuss two main concepts: *tubular neighbohoods* and *medial axes*. These concepts are intrisically connected with the shape and boundary of a solid submanifold $\mathcal{O} \subset \mathbb{R}^n$.

5.1 Tubular Neighborhoods

The concept of *tubular neighborhood* is of fundamental importance in the study of differentiable manifolds, because it relates a surface with its normal vector field. Thus, by investigating the tubular neighborhood of a surface M, it is possible to make a connection between the isocontour $M = f^{-1}(c)$ and the associated implicit function f. This is done indirectly through the vector field normal to $f^{-1}(c)$, given by the gradient of f.

5.1.1 Definitions

A *normal segment* $[p, b]$ to a surface M at the point p is a line segment from p to b such that $p \in M$ and $[p, b]$ is contained in $T_p^{\perp}M$, the orthogonal complement of the tangent space of M at p. The point p is called the *foot point* of b in M. The 1-dimensional set of all segments normal to M at the point p with length less than ε, is denoted as $B^{\perp}(p, \varepsilon)$.

An *admissible normal radius* for a subset $Z \subset M$ is a real number $\varepsilon > 0$ such that any two normal segments, $[p, x]$ and $[q, y]$, with $p \neq q \in Z$ and length $< \varepsilon$ do not intersect. See Figure 5.1.

An *ε-tubular neighborhood* V_ε of a surface M is defined as the union of all segments normal to M with radius $\varepsilon(p)$ such that $\varepsilon(p)$ is an admissible normal radius for M at p. That is

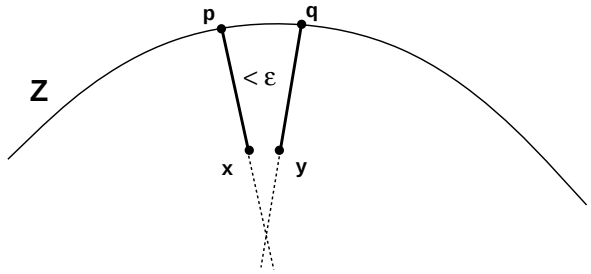


Fig. 5.1. Admissible Normal Radius

$$V_\varepsilon(M) = \bigcup_{p \in M} B^\perp(p, \varepsilon(p)).$$

Figure 5.2 shows an ε-tubular neighborhood of a two-dimensional curve.

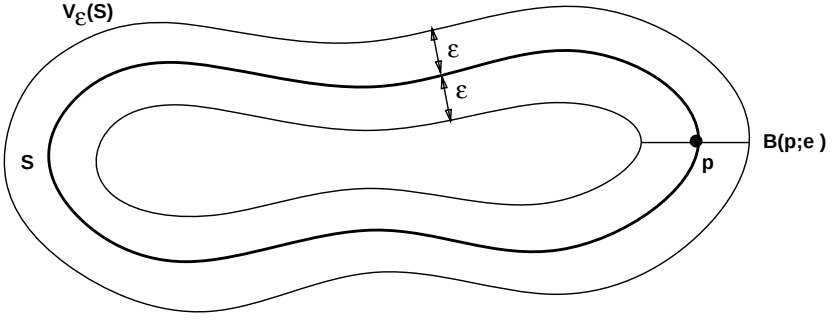


Fig. 5.2. Tubular neighborhood

It is possible to prove that any regular implicit surface $M = f^{-1}(c)$ has a tubular neighborhood. This is a consequence of the fact that the gradient vector field of f does not vanish on M.

5.1.2 The Projection on the Surface

The existence of a tubular neighborhood makes possible to define a projection function $\pi : V_\varepsilon(M) \to M$ which, for each point $x \in V_\varepsilon$, associates the unique foot point p of the normal segment that contains x.

This projection is a very powerful mathematical instrument that can be used for many purposes in the study of surfaces. In particular, it implies that a tubular neighborhood is equivalent to the product space $M \times B(\varepsilon)$, where $B(\varepsilon)$ is an open interval $\in \mathbb{R}$ with center at the origin and radius ε. This corresponds to a topological open cylinder. See Figure 5.3.

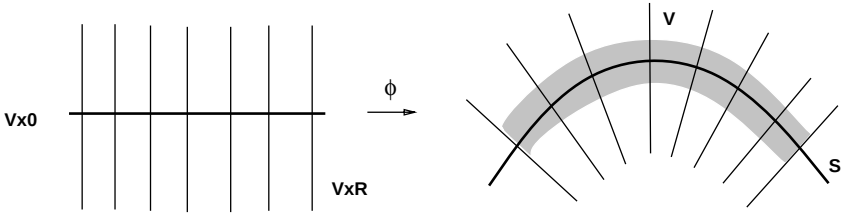


Fig. 5.3. Product space

5.1.3 The Maximal Tubular Neighborhood

A tubular neighborhood of a surface is called *maximal* when it contains all possible ε-tubular neighborhoods of that surface.

Intuitively the maximal tubular neighborhood extends the normal fibers of V_ε as far as possible without violating the projection conditions above.

The maximal tubular neighborhood $V_{\max}$ of a surface $M = f^{-1}(c)$ is *unique*: $V_{\max}$ gives the largest open set $M \subset U \subset \mathbb{R}^n$ where a continuously differentiable distance function $f : U \to \mathbb{R}$, associated with the surface $f^{-1}(c)$, can be defined. In other words, $V_{\max}$ is the maximal domain U in which an implicit function f can be constructed such that f does not have singular points in U (or, equivalently, ∇f does not vanish in U).

Example 5.1. (Unit Circle) The maximal tubular neighborhood of the unit circle is the entire plane minus the origin ($\mathbb{R}^2 - \{(0,0)\}$).

5.2 Medial Axes

The medial axis describes the structural essence of a shape. It provides a means to characterize the topology of solids and to construct a

geometric model of their boundary. In particular, as suggested earlier, the medial axis may serve as the basis for implicit surface models if it is associated with a suitable distance function.

The medial axis has also been extensively used in computer vision for shape recognition and classification purposes [Blu67], [Hof92], [NP85].

5.2.1 Definitions

The *distance* from a point p to a surface M in $\mathbb{R}^n$ is the minimum of the Euclidean distance $d_E(p, s)$, where $s \in M$:

$$d(p, M) = \inf_{s \in S} d_E(p, s).$$

When M is compact, d is continuous, and so for every $p \in \mathbb{R}^n$ there is at least one point $s_0 \in M$ such that $d(p, M) = d(p, s_0)$. Such point s_0 is called the *foot point* of p in M (Note that this definition is equivalent to the one in Section 5.1).

The *medial axis* of a region of $\mathbb{R}^n$ bounded by a surface M is the closure of the set of points $p \in \mathbb{R}^n$ such that $p \notin M$ and p has more than one foot point on M.

The *interior medial axis* consists of all medial axis points that are interior relative to M. Similarly, the *exterior medial axis* consists of the set medial axis points that are exterior relative to M.

5.2.2 Intuition

Intuitively, the medial axis is formed by all the points of the ambient space that have more than one geodesic path to the surface M.

An alternative definition of the medial axis with a geometric flavor employs the notion of maximal spheres. A sphere is called *maximal* with respect to a region R if it is contained entirely in R, but not properly contained by any other sphere in R. See Figure 5.4.

The medial axis can be defined as the locus of the centers of all spheres that are maximal with respect to the interior and exterior regions delimited by M.

The definition above was proposed in connection with the medial axis transform used in vision [Blu67].

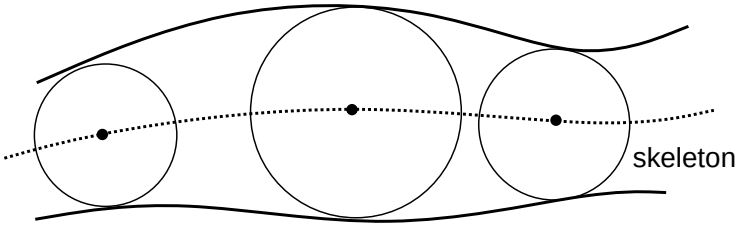


Fig. 5.4. Maximal Sphere

5.2.3 Characteristics

The medial axis of a region of dimension d is formed by the union of elements of dimension $d-1$ or lower.

A solid has a unique medial axis. Figure 5.5 shows the medial axis of a rectangular solid region of $\mathbb{R}^2$.

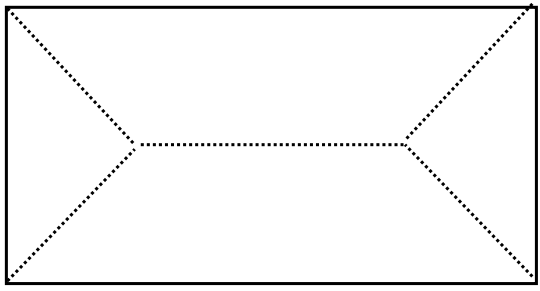


Fig. 5.5. Medial axis

5.3 Morse Theory

Topology studies the properties of a shape that do not change under deformations. Morse Theory describes the topology of a manifold based on configuration of *critical points* [Mil65]. The relationship between the topological structure of the manifold and arrangements of critical points can be encoded using a data structure from algebraic topology called the *CW-complex*.

5.3.1 Critical Points and the Hessian

Let $h : M \to \mathbb{R}$ be a smooth function defined on a smooth manifold M. The *critical points* of h are the points $p \in M$ where the $\nabla h(p) = 0$. The *critical values* are the corresponding values $h(p)$ at these critical points.

The Hessian $H(p)$ of h at $p \in M$ is the matrix of second derivatives of h. Let λ_i be the i-th eigenvalue and v_i its corresponding eigenvector, given in non-decreasing order. Critical points are classified by the number of negative eigenvalues of the matrix H, which correspond to a saddle along some of the principal directions given by v_i. Therefore, there are $n + 1$ categories of critical points for a n-dimensional manifold: 0-saddles (minima of h), n-saddles (maxima of h), and k-saddles, with $0 < k < n$.

5.3.2 Morse Function

A function $h : M \to \mathbb{R}$ is called a *Morse function* if its Hessian is non-singular for every point $p \in M$.

Morse theory gives a connection between arrangements of critical points of a Morse function on M and the topology of M. This can be easily understood through a classical example.

Example 5.2. Consider a parametric torus $g : U \to \mathbb{R}^3$ encircling the z-axis, and a Morse function $h : g \to \mathbb{R}$, such that $h(g(u, v)) = y$. There exist four critical points where the gradient

$$\nabla h = (0, 1, 0) \cdot \nabla g = \frac{\partial g}{\partial y}$$

vanishes: one type-0 critical point at a, on the bottom of the torus; one type-2 critical point at d, on the top of the torus; and two type-1 critical points at b and c, respectively the bottom and top of the hole. See Figure 5.6-(a).

We can imagine the process of constructing this torus from the bottom up. First, a point is created at the minima a. Then, opposing sections are glued at the saddles b and c. Finally, the torus is closed at the top d.

Note that a new surface component is created when a type-0 critical point is reached, opposing surface components are connected when a type-1 critical point is found and a hole in the surface is sealed when a type-2 critical point is reached.

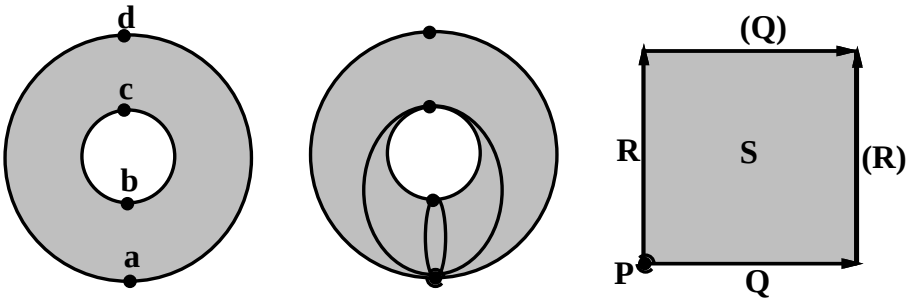


Fig. 5.6. Critical points of a Morse function (a), the CW-complex (b) and topological structure (c) of a torus

5.3.3 The *CW*-Complex

The topology of a shape is conveniently defined in homotopy theory by a topological graph. As we have seen in Chapter 4, the topological graph is constructed using cells of different dimension. An n-cell is the building block of a shape of dimension n. For example, a 0-cell is a point, a 1-cell is a curve segment, a 2-cell is a surface patch and a 3-cell is a solid region.

These cells are closed, i.e., their boundaries are part of the cells. In that way, higher dimensional cells can be constructed form lower dimensional cells. Each n-cell is homeomorphic to the n-ball

$$B^n = \{x \in \mathbb{R}^n : \|x\| \leq 1\}$$

Using this homeomorphism we can map the boundary of a n-cell to the $(n-1)$-sphere

$$S^{n-1} = \{x \in \mathbb{R}^n : \|x\| = 1\}$$

The *CW-complex* is constructed out of cells of increasing dimensions, by identifying the boundary of a n-cell with the union of a collection of $(n-1)$-cells that already belong to the complex.

The construction process of a CW-complex C for a manifold M is as follows. Starting with the empty set, 0-cells are attached to C as the union of disjoint points. Next, 1-cells are constructed as curve segments that lie on these points. The process continues until n-dimensional cells defining the topology of M are generated.

The theorem below relates the topology of a manifold with Morse functions and CW-complexes.

Theorem 5.3.1. *A compact manifold M has the homotopy type of a CW-complex consisting of a λ-cell for each critical point of type λ for a given Morse function h defined on M.*

The CW-complex can be constructed from the separatrix structure connecting critical points to each other. 0-cells exist at type-0 critical points. 1-cells are separatrix curves starting at type-$(n-1)$ critical points and ending at type-0 critical points. The process is similar for n-dimensional cells.

Continuing with our previous example, we can see in Figure 5.6 the CW-complex and the topological structure of the torus.

The 0-cell P is associated with type-0 critical point a. The 1-cells Q and R start respectively at type-1 critical points b and c. The 2-cell S starts at type-2 critical point d.

5.3.4 Distance Fields as Morse Functions

A Morse function h can be interpreted as a smooth height function on M. We can construct h based on a distance function on M from a point $p \in M$. In this case, h will work as a generalized height function. The function h may be computed though a front propagation method, where the points of the evolving front at time t have distance t from p.

This construction is very useful because it allows us to study the topology of piecewise linear manifolds, such as polygonal meshes [AE98].

As an example, we can define topological distance on a polygonal mesh using a wave traversal technique. In a pre-process step, all vertices of the mesh are initialized with distance ∞. The process starts with an initial vertex p_0 of the mesh that is assigned distance 0. Then, the vertices $p \in N_1(p_0)$, in the discrete 1-neighborhood of p_0 are assigned distance 1. The process continues by propagating the wavefront until the whole mesh is covered. This is accomplished assigning distance $t+1$ to all vertices of the wavefront that have their distance greater than t.

As the wavefront traverses the mesh, some vertices are found to be critical points. The initial vertex is a type-0 critical point because all of its neighbors have larger distance values. The process terminates at type-2 critical points, which are vertices whose neighbors have smaller

distance values. The wavefront is divided or reconnected at type-1 critical points, whose neighbors have distances values that vary with peridicity 2 around the critical point.

Distance fields will be also instrumental in defining the topology of implicit shapes, as we will see next.

5.3.5 Topology of Implicit Shapes

We have seen that a Morse function is the fundamental element to investigate the topology of a manifold M. In the above discussion, we employed a scalar function $h : M \to \mathbb{R}$, defined on the manifold M itself.

Such a setting is adequate to study parametric shapes. For example, in the case of a k-dimensional manifold M in $\mathbb{R}^n$, given by a function $g : U \subset \mathbb{R}^k \to \mathbb{R}^n$, the Morse function can also be defined in the domain U of g.

However, this setting is not so convenient to study implicit shapes. If we have the manifold M given in implicit form by a function $f : V \subset \mathbb{R}^n \to \mathbb{R}^{n-k}$, it is more appropriate to use a Morse function defined on V, the embedding space of M. Observe that we can consider M as a submanifold of the manifold V. The topological structure of M is related to the topological structure of M when the function f is used to derive the Morse function.

More specifically, the function f can be interpreted as a generalized distance function from the boundary of M on V. Note that, in the case of a codimension-1 surface, the boundary $f^{-1}(c)$ is a level set of f. Critical points of the distance function occur at key locations defining the topology of the surface. In 3-space there are 4 types of critical points. Maxima critical points (3-saddle) are inside the shape. Minima critical points (0-saddle) are outside the shape. A 2-saddle occurs between two components of the shape. These components are connected if the critical value is positive and disconnected if it is negative. A 1-saddle occurs in the middle of a handle. The handle is filled if the critical value is positive and is pierced if it is negative.

A key insight is that the distance functions and the medial axis of a shape are intrinsically related to the topology of implicit shapes. The distance transform allows us to define a Morse function while the medial axis.

5.4 Surfaces in Space

In order to analyze the effectiveness of implicit models it is necessary to study the extrinsic properties of its bounding surface. The reason this is so important lies in the fact that implicit objects are defined by a function of space. The investigation of how the surface is embedded provides the required criteria to characterize the implicit function.

5.4.1 Surfaces Structuring Space

The maximal tubular neighborhood of a surface M and the medial axis of the region enclosed by M provide a structure of the ambient space that reveals essential aspects of the implicit surface model.

These two geometric entities are dual structures. More than that, one is the complement of the other in $\mathbb{R}^n$. This is clear from their very definition.

The concepts of tubular neighborhood and medial axis are totally independent of the implicit description. But, in some sense, they capture all properties of a surface which depend on its embedding in the ambient space. This is precisely the reason why they are important: they relate a surface with the space in which it lives.

For implicit regular surfaces:

- The medial axis is contained in the set of singular points.
- The tubular neighborhood is contained in the complement of the set of singular points.

The above statements make clear the duality of these structures.

These instruments are useful to analyze and construct implicit surface models. Tubular neighborhoods provide a way to investigate the domain of the implicit function; medial axes provide simpler geometric objects from which the implicit function can be defined.

5.4.2 What is a Good Implicit Model?

There are many properties that a good implicit surface model should have. Some of them, such as simplicity, conciseness, or completeness, are of qualitative nature and apply to any type of geometric model. Others are specific to the implicit model.

In simple terms, specific properties of the implicit surface model are related to:

- the information conveyed by the value of f;
- the extent of the domain of f with valid information.

According to these criteria, a good implicit function should provide the desired information over a prescribed region of space.

One possible criterion to specify such a function is faithfulness to a metric of the ambient space. Under this assumption, the optimal implicit function is the Euclidean distance from the surface. This is a linear function that is singular at the medial axis points. Such a function will be briefly described in Section 5.5.2.

Another optimality criterion is smoothness. In that case, it is not possible to employ a true metric. So, the model has to resort to a pseudo-metric. Also, some control over the domain of the implicit function may be required. The most natural choices are, either the entire $\mathbb{R}^n$, or a prescribed ε-tubular neighborhood of the surface.

5.5 Universal Representation

A universal representation is a canonical description in which any objects of a certain class can expressed. The medial axis and the distance function are two of such a description in relation to implicitly defined surfaces and solids.

5.5.1 Medial Axis Models

The medial axis model of a shape $\mathcal{O} \in \mathbb{R}^n$ is the union of all centers of all maximal disks which fit in $\mathcal{O}$, along with the radius of these disks. This defines the set of points equidistant to the boundary of D. In a sense this representation consists of a medial axis induced by the Euclidean metric.

The medial axis description represents objects in a form that is well suited for shape analysis. Although, it has been extensively used in vision and image processing, only recently it has been explored in computer graphics.

Although an exact construction of the medial axis transform is difficult, there are reasonable computational methods to compute a discrete approximation of it.

An algorithm to obtain the discrete medial axis is as follows: First generate a set of points on the object boundary. Calculate a Delaunay triangulation of these points. Classify the centers of the Delaunay

regions and group them into a medial axis which forms the discrete medial axis. For a sufficiently dense set of points the discrete medial axis converges to the analytical medial axis.

It must be possible to compute the medial axis of implicit objects more effectively exploiting the intrinsic characteristics of the implicit form. This is an open research topic and is related to level sets and PDE's [Tei98, Bar01].

5.5.2 Distance Function Models

Distance function models are closely related to cyclographic maps and to the solution of the eikonal equation [Hof91].

This implicit surface model is based on the Euclidean metric. The surface M is defined as the zero set $f^{-1}(0)$ of the function $f : \mathbb{R}^n \to \mathbb{R}$,

$$f(x) = d(x, p) - r(p),$$

where $d(x, p)$ is the distance from x to its closest point p on the interior medial axis of M and $r(p)$ is the distance of p from M (as defined in Section 5.2).

The value of f gives the signed distance from the point x to its foot point on M. The gradient of f is a unit vector field, defined over the maximal tubular neighborhood of M, that points in the normal direction to the surface.

Observe that this model generalizes an offset surface model with constant radius r.

5.6 Summary

In this chapter we demonstrated that a solid and an implicit function are related by dual structures: the medial axis of the solid and the maximal tubular neighborhood of its boundary. These fundamental structures define the embedding of a shape in space and, therefore the characteristics of the implicit function. Finally, we discussed a set of criteria which could be used for analyzing implicit models. From an interpretation of the implicit function as a distance function, we identified two classes of implicit models – the ones based on a true metric and on a pseudo-metric.

6. Implicit Objects

This chapter studies in detail objects whose geometry is defined in implicit form. We will be mainly concerned with three-dimensional surfaces and solids described implicitly. Applications of implicit curves arise primarily when we need to compute curved edges as intersections of implicit surfaces [Hof89]. This is important in the conversion between CSG and boundary representations, but will not be addressed here. Many important problems in different areas can be reduced to the computation of n-dimensional implicit manifolds (see Chapter 17).

6.1 Definition of an Implict Object

A subset $\mathcal{O} \subset \mathbb{R}^n$ is called an *implicit object* if there exists a function $f : U \to \mathbb{R}^k$, $\mathcal{O} \subset U$, and a subset $V \subset \mathbb{R}^k$, such that $\mathcal{O} = f^{-1}(V)$. That is,

$$\mathcal{O} = \{p \in U : f(p) \in V\}\,.$$

The definition of implicit object given above is broad enough to include a large family of subsets of the space. In fact, this definition is too broad to be tackled with robust tools from Mathematics. To show the generality of this definition, we just remark that *any closed subset of the space can be represented as an implicit object of a smooth function* [Whi57]. This includes, for instance, fractal sets. Therefore, without imposing more restrictions on the function f and the set V, it is very difficult to develop a computationally sound framework for implicit objects.

An implicit object is called *regular* if the function f satisfies some of the regularity conditions of Chapter 3.4. An implicit object is called *valid* if it defines a topological manifold. Note that a regular implicit object is valid.

The implicit object $\mathcal{O} = f^{-1}(V)$ may have many connected components. A particular, and very important case, occurs when $V = \{c\}$,

$c \in \mathbb{R}^k$, f is a differentiable function, and c is a regular value of f ($\mathcal{O}$ is a regular object). From Chapter 3.4 we conclude that $f^{-1}(c)$ is an orientable submanifold of $\mathbb{R}^3$.

When f is a real valued function, that is $k = 1$, then f is a *point-membership classification function* that returns a value according to the relationship of a point $p = (x_1, \ldots, x_n)$, given as its argument, with the implicit object $\mathcal{O}$ defined by f.

$$f(p) \begin{cases} > 0 \; p \in \text{exterior of } \mathcal{O} \\ = 0 \; p \in \text{boundary of } \mathcal{O} \\ < 0 \; p \in \text{interior of } \mathcal{O} \end{cases}$$

When $\mathcal{O}$ is connected it defines a codimension-1 manifold that separates the space into two connected components A, B whose common boundary is the object $\mathcal{O}$. An important situation occurs when one of the regions A, B is bounded. In this case we are able to define well behaved implicit solids. We will analise this in dimensions 2 and 3.

Take $f : U \subset \mathbb{R}^2 \to \mathbb{R}$, $c \in \mathbb{R}$ regular value of f, $f^{-1}(c)$ connected and compact. We may suppose that the set $S = \{(x, y) \in \mathbb{R}^2 : f(x, y) < 0\}$ is the bounded component of the complement of $f^{-1}(c)$ in $\mathbb{R}^2$. Now consider the set $\overline{S}$ consisting of the implicit object defined by $\overline{S} = f^{-1}((-\infty, c])$. If f is conveniently chosen, $\overline{S}$ is a solid figure on the plane whose boundary is the implicit curve $f^{-1}(c)$. See Figure 6.1-a.

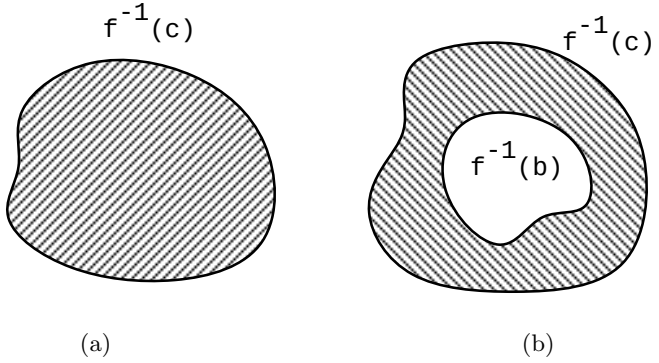


Fig. 6.1. Implicit solids with one (a) and two (b) shells

The implicit object $\overline{S}$ is regular, as can be easily verified. Note that if $b < c$ is also a regular value of f, the implicit object $R = f^{-1}([b, c])$

represents a region with disconnected boundary, constituted by the curves $f^{-1}(c)$ (one component) and $f^{-1}(b)$ (one or more components). See Figure 6.1-b.

The above description generalizes easily to define "implicit solids" in 3-D space. A solid can be defined as the inverse image of an interval $(-\infty, a]$ or $[a, b]$ by a function $f : \mathbb{R}^3 \to \mathbb{R}$. For a convenient choice of the function f, the boundary of the solid corresponds respectively to the sets $f^{-1}(a)$ or $f^{-1}(a) \cup f^{-1}(b)$. The boundary of the solid is the union of disjoint implicit surfaces. Note that $x \in f^{-1}((-\infty, a])$ is equivalent to $f(x) \leq a$ and similarly $x \in f^{-1}([a, b])$ is equivalent to $a \leq f(x) \leq b$. Another important observation is that a solid of the second form, $f^{-1}([a, b])$, may also be obtained from a difference of two solids of the first form, $f^{-1}((\infty, a]) \setminus f^{-1}((\infty, b])$ (see Chapter 8). Therefore, the first form is sufficient in a modeling system that includes boolean operations. [1]

In the following analysis we will be interested in the properties of implicitly defined objects from a mathematical and computational point of view. These properties will be exploited in the development of models and in the implementation of algorithms for implicit objects.

Now we are able to define implicit objects to be used in geometric modeling and computer graphics. An *implicit curve* is a valid implicit object of dimension 1. An *implicit surface* in $\mathbb{R}^3$ is a valid implicit object of dimension 2. An *implicit solid* is a valid implicit object of dimension 3 with boundary.

6.2 Mathematical Elements

We were very careful in defining the notion of implicit objects. This is justified because of the attributes we want to attach to them. The power and robustness of this definition provides the tools to study implicit objects.

6.2.1 The Function f

An implicit object $\mathcal{O} = f^{-1}(V)$ is completely characterized by the point membership function f and the subset V.

[1] We should also point out that, depending on the choice of an implict function f, it is possible to obtain a boundary, $f^{-1}(c)$, consisting of two or more disconnected surfaces.

Regularity conditions relating f and V are very difficult to be verified computationally. If $V = \{c\}$, Sard's theorem says that c can be chosen to be a regular value with probability one (the set of regular values is open and dense). (See [Mil65].) But this is not of much use from a computational point of view.

For codimension-one implicit surfaces there are two *canonical forms* defined respectively by the value of $c = 0$ and $c = 1$. In the first, the surface is the zero-set of f and in the second, the surface is the set of level one. These forms will be useful for operations with implicit objects as will be seen in Chapter 7.

6.2.2 The Domain of f

The domain U of $f : U \subset \mathbb{R}^n \to \mathbb{R}^k$, is the region of space where we want to evaluate the funtion f.

For a regular implicit object $\mathcal{O} = f^{-1}(V)$, it is convenient to choose the domain U as some tubular neighborhood of $\mathcal{O}$. This guarantees that $\text{grad} f(p) \neq 0$ for $p \in U$.

6.2.3 The Characteristic Function

The *characteristic function* $\chi_{\mathcal{O}} : \mathbb{R}^n \to \{0, 1\}$ of a solid implicit object $\mathcal{O} = f^{-1}(V)$, $V = [-\infty, c]$ is defined as $\chi_{\mathcal{O}}(p) = 1$ if $p \in \mathcal{O}$ and $\chi_{\mathcal{O}}(p) = 0$ if $p \notin \mathcal{O}$.

The function $\chi_{\mathcal{O}}$ can be trivially derived from the function f:

$$\chi_{\mathcal{O}}(p) = \begin{cases} 1 & \text{if } f(p) \leq c \\ 0 & \text{otherwise} \end{cases}$$

6.2.4 The Gradient of f

If $\mathcal{O} = f^{-1}(V)$ is an implicit object, $f : U \subset \mathbb{R}^{m+k} \to \mathbb{R}^k$ and $f_1, \ldots, f_k$ are the coordinate functions of f, then we can take the k gradient vector fields, $\text{grad} f_1, \ldots, \text{grad} f_k$, for each $c \in V$. These vector fields are orthogonal to each level set $f^{-1}(c)$ of the object $\mathcal{O}$.

A particular and important case occurs for codimension-one regular implicit surfaces, $S = f^{-1}(c)$ with $f : U \subset \mathbb{R}^n \to \mathbb{R}$. Then we have

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots \frac{\partial f}{\partial x_n} \right)$$

The gradient vector points to the direction of greatest change in f.

The gradient field associated with implicit objects have been extensively used in physically-based constructions related to modeling and animation of implicit objects. (See [VdMG91a], [dFdMGTV92].) This will be discused in more detail later on.

6.2.5 The Hessian of f

The Hessian form associated with a function $f(x_1, x_2, ...x_n)$, is the matrix of second-order partial derivatives of f with respect to x_i:

$$Hf(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}$$

The Hessian indicates the rate of change in the gradient of f and will be useful, among other things, for computing the curvature of implicit objects.

6.3 Geometrical Characterization

Using the mathematical elements discussed above we further develop a geometrical characterization of an implicit surface.

6.3.1 Local Parametrization

We saw in Section 3.10 that an implicitly defined surface can always be parametrized locally in a trivial way. This parametrization is given by the graph of a function relative to one of the coordinate axis in $\mathbb{R}^n$.

Let $p = (x_1, \ldots, x_n)$ be a point of $M = f^{-1}(c)$. Since c is a regular value we can assume, by renaming axis if necessary, that $\frac{\partial f}{\partial x_n} \neq 0$. We construct a parametrization of M in the neighborhood of p such that

$$f(x_1, \ldots, x_n) = (x_1, \ldots, h(x_1, \ldots, x_{n-1}))$$

The inverse function theorem guarantees that if $\frac{\partial f}{\partial x_n} \neq 0$ then there exists a neighborhood of p where is possible to compute $x_n = h(x_1, \ldots, x_{n-1})$ from $f(x_1, \ldots, x_n) = c$.

Figure 6.2 shows a diagram of the local parametrization of an implicit surface.

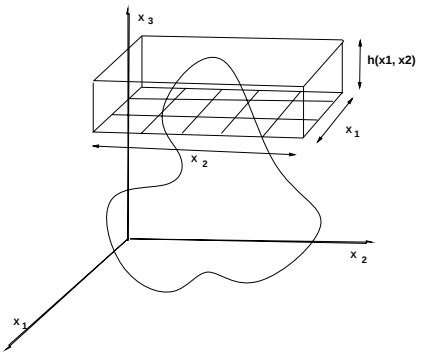


Fig. 6.2. Local parameterization

6.3.2 Orientation and Surface Normal

The results of Section 3.6 show that regular implicit surfaces and solids as defined in the beginning of this chapter are always orientable. The orientation is given by the gradient vector fields of the components of the implicit function.

In the case of a codimension-one implicit surface, $M = f^{-1}(c)$, the vector field

$$N = \frac{\nabla f(p)}{\|\nabla f(p)\|}, \quad p \in M$$

defines an orientation of M.

On a two-dimensional surface in $\mathbb{R}^3$ we use orientation to define a direction of rotation in the tangent planes of the surface.

6.4 Differentiable Attributes

Differentiable attributes are very important, because they enable us to use techniques from differential topology and differential geometry in the study of implicit objects. In this subsection we will describe the most important of these attributes.

6.4.1 Geodesics

Geodesics are, from a local point of view, curves of minimum length on a surface. They play the same role as do straight lines in $\mathbb{R}^n$, but

due to topological complexity of the surface, they might not minimize length globally.

A *geodesic* in a surface M is a parametrized curve $\alpha : I \to M$ whose acceleration is orthogonal to M. It has no acceleration component tangent to the surface. They also have constant speed.

A geodesic $\alpha(t) = [x_1(t), \ldots, x_n(t)]$ has to satisfy this system of differential equations:

$$\frac{d^2x_i}{dt^2} + \sum_{j,k=1}^{n} N_i(x_1, \ldots, x_n)\frac{\partial N_j}{\partial x_k}(x_1, \ldots, x_n)\frac{dx_j}{dt}\frac{dx_k}{dt} = 0$$

Given a point $p \in M$ and a velocity vector v at p, then there is a unique geodesic that is the solution of the system of differential equations above with these initial conditions.

6.4.2 The Gauss Map

We know that for curves the curvature at a point p is measured by a number. For surfaces, it is measured by a map.

Let M be an oriented codimension-one submanifold in $\mathbb{R}^{n+1}$. Denote by $N(p)$ the unit normal vector to M at p. The Gauss map, $N : M \to S^n$, associates to each $p \in M$, the point $N(p)$ on the unit n-dimensional sphere S^n. (See Figure 6.3).

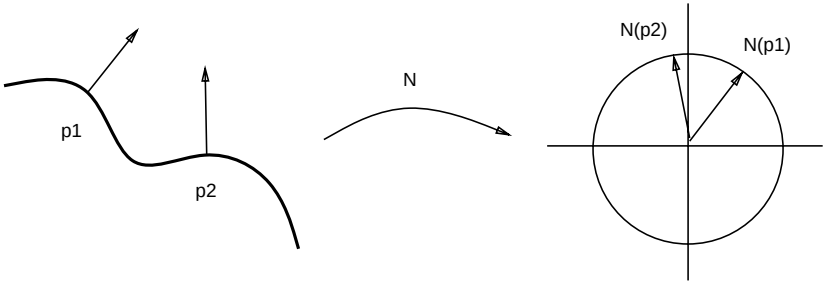


Fig. 6.3. The Gauss map

The derivative N' of N is a measure of how the normal vector is changing. Since N is a unit vector, N' indicates the change in its direction, and therefore, N' conveys information about the curvature of the surface. It is easy to show that:

- $N'(p)$ is a linear operator on T_pM.
- $N'(p)$ is self-adjoint.

$N'(p)$ is sometimes called in the literature by *Weingarten map.*

6.4.3 The Fundamental Forms

For any self-adjoint linear transformation on a vector space with dot product there is a real valued function $\mathcal{Q}(v) = N(v) \cdot v$ called the *quadratic form* associated with N.

The *first fundamental form* of M at p is the quadratic form $\mathcal{F}_p$ associated with the identity transformation on T_pM.

$$\mathcal{F}_p(v) = v \cdot v$$

Therefore, this quadratic form defines the inner product in each tangent plane to the surface. All the metric properties of the surface are connected to it.

The *second fundamental form* of M is the quadratic form $\mathcal{S}_p$ associated with the Weigarten map N_p at a point p.

$$\mathcal{S}_p(v) = N_p'(v) \cdot v$$

A surface is completely determined up to rigid motion by its first and second fundamental forms. See [dC74].

If $M = f^{-1}(c)$ is a regular implicit surface in $\mathbb{R}^{n+1}$ with orientation given by the normal vector field

$$\frac{\nabla f}{\|\nabla f\|}$$

and $v = (v_1, \ldots, v_{n+1})$ is a tangent vector to M at a point p, $v \in T_pM$, the second fundamental form is related to the Hessian form of f. More precisely,

$$\mathcal{S}_p(v) = \frac{-1}{\|\nabla f(p)\|} \sum_{i,j=1}^{n+1} \frac{\partial^2 f}{\partial x_i \partial x_j}(p) v_i v_j$$

or in matrix notation

$$\mathcal{S}_p(v) = \frac{-1}{\|\nabla f(p)\|}\, v^T \cdot Hf(p) \cdot v$$

6.4.4 Surface Curvature

The second fundamental form allows us to investigate the curvature of a surface.

The *normal curvature* of M at p in the direction v is defined by

$$k(v) = \mathcal{S}_p(v) = \langle N'_p(v)v \rangle$$

when $\|v\| = 1$.

In other words, $k(v)$, is equal to the normal component of acceleration of any curve, contained in M, passing through p with velocity v.

Since $N'(p)$ is a self-adjoint linear transformation of T_pM, there exists an orthonormal basis $v_1, \ldots, v_n$ of T_pM whose vectors, v_i, are eigenvectors of $N'(p)$. The eigenvalues $k_1(p), \ldots, k_n(p)$ of $N'(p)$ are called *principal curvatures* of M at p and the correspondent unit eigenvectors of $N'(p)$ are called *principal directions*. The principal curvatures are stationary values of normal curvature $k(p)$ and among them $k(p)$ attains its minimum and maximum values.

In general, we can diagonalize the Hessian matrix H to obtain the eigeinvalues and eigenvetors of $N'(p)$. Alternatively, the formulas below [MAS91] allow us to compute the principal curvatures k_i and principle directions v_i directly from H. We have

$$k_i = \frac{a^T Ha + b^T Hb \pm \sqrt{(a^T Ha - b^T Hb)^2 + 4(a^T Hb)^2)}}{2\|\nabla f\|}$$

and

$$v_i = \begin{pmatrix} a_1 + b_1 \dfrac{\|\nabla f\| k_i - a^T Ha}{b^t Ha} \\ a_2 + b_2 \dfrac{\|\nabla f\| k_i - a^T Ha}{b^t Ha} \\ a_3 + b_3 \dfrac{\|\nabla f\| k_i - a^T Ha}{b^t Ha} \end{pmatrix}$$

for $i = 1, 2$, where

$$a = \left(\frac{1}{\gamma} \frac{\partial f}{\partial x_2}, \quad -\frac{1}{\gamma} \frac{\partial f}{\partial x_1}, \quad 0 \right)^T$$

and

$$b = \left(\frac{1}{\gamma \|\nabla f\|} \frac{\partial f}{\partial x_1} \frac{\partial f}{\partial x_3}, \quad \frac{1}{\gamma \|\nabla f\|} \frac{\partial f}{\partial x_2} \frac{\partial f}{\partial x_3}, \quad -\frac{\gamma \|}{\|\nabla f\|} \right)^T$$

The trace and determinant of the Gauss map are important intrinsic properties of a surface.

The *mean curvature* $\overline{K}(p)$ of M at p is $1/n$ times the trace of $\mathcal{S}(p)$:

$$\overline{K}(p) = \text{trace } \mathcal{S}(p) = \frac{1}{n} \sum_i^n k_i(p)$$

It is the average value of the principal curvatures at p.

The determinant of $\mathcal{S}(p)$ is called the *Gauss-Kronecker curvature*, K_G of M at p.

$$K_G(p) = \det \mathcal{S}(p) = \prod_i^n k_i(p)$$

It is equal the product of the principal curvatures.

6.5 Computational Attributes

The elements described in this chapter completely characterize the general aspects of implicit objects. They represent their essence – the common properties to the class of objects defined in implicit form. Different types implicit objects have other properties that are specific to their particular definitions.

6.5.1 Object Oriented Approach

From a computational standpoint it is important to explore these general and specific properties of implicit objects in a computer graphics system.

The object oriented paradigm [Wis90] establishes a framework where a uniform representation can coexist with specialized methods. This is done through the specification of basic functions and operations that an object has to perform in the system.

This paradigm is well suited to the representation and manipulation of implicit objects on the computer. An implicit object is associated with some class, and its methods implement the basic functions describing the mathematical attributes of the object.

6.5.2 Basic Functions

The mathematical elements of Section 6.2 can be used to define basic functions for the implicit object.

Some of the basic functions for codimension-one implicit objects are:

- `List f_domain(f)` - returns the list of bounding boxes that encloses the implicit object.
- `Real f_value(p)` - returns the value of the implicit function at point p.
- `Vector f_grad(p)` - returns the gradient of the implicit function at point p.
- `Matrix f_hessian(p)` - returns the hessian of the implicit function at point p.

This page intentionally left blank

7. Manipulating Implicit Objects

This chapter investigates how implicit objects can be manipulated for modeling purposes. We study the effects of operations used to modify objects in different ways.

There are two alternatives to perform transformations on implicit objects. One is to change the function F, the other is to alter the space in which F is defined. We will study unary operations that act on the

- range of F;
- domain of F.

7.1 The Implicit Function as a Metric

The implicit function $F : \mathbb{R}^n \to \mathbb{R}$ can be interpreted to estimate the distance from a point in space to the surface described by F. The *algebraic distance* of $p \in \mathbb{R}^n$ to $F^{-1}(0)$ is given by $F(p)$. This is essential to several unary operations considered in this chapter, as well as to some binary operations studied in the next chapter, such as blends.

In this sense, $F : U \subset \mathbb{R}^n \to \mathbb{R}$ gives the signed distance, induced by some pseudo-metric $d : \mathbb{R}^n \to \mathbb{R}$, of points $p \in U$ to the level surface $S = F^{-1}(0)$.

Example 7.1. (Circle) The circle with center o and radius r can be defined as the inverse image $F^{-1}(0)$ of the implicit function $F(p) = d(o, p) - r$, where d is the Euclidean metric.

In a finite dimensional vector space all distance functions derived from a norm are topologically equivalent. Particularly, in $\mathbb{R}^n$ we have the following classical metrics

$$d_1(p, q) = |p_1 - q_1| + \cdots + |p_n - q_n|$$
$$d_2(p, q) = \sqrt{|p_1 - q_1|^2 + \cdots + |p_n - q_n|^2}$$
$$d_\infty(p, q) = \max(|p_1 - q_1|, \ldots, |p_n, q_n|)$$

Note that, although these metrics are equivalent from a topological standpoint, they produce different geometric results when used to define an implicit surface. Figure 7.1 shows the circle of Example 7.1 using the metrics above.

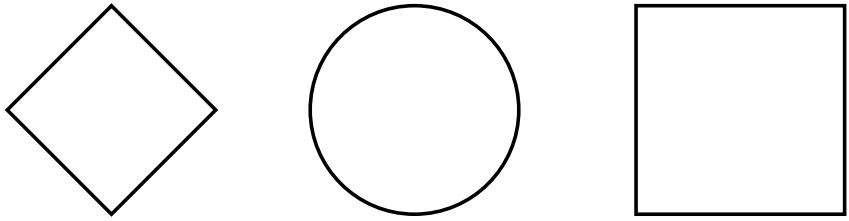


Fig. 7.1. The unit ball in different metrics

The metrics d_1, d_2 and d_∞ can be generalized to form the m-norms, a one parameter family of distance functions:

$$d_m(p,q) = \left[\sum_i \|p_i, q_i\|^m\right]^{1/m}.$$

The fact that altering the metrics causes shape changes may be exploited by the implicit surface model if this is incorporated into the implicit function.

Example 7.2. (*Superellipse*) The superellipse model allows the control of shape through a modification of the metric. The function $f(x)$ is defined as

$$\left(\left(\frac{x_1}{a}\right)^e + \left(\frac{x_2}{b}\right)^e\right)^{\frac{1}{e}},$$

where e controls the roundness of the shape and a and b are respectively horizontal and vertical scaling factors.

In the next chapter we will investigate further superquadrics in the context of blends.

7.2 Properties of the Implicit Function

Several properties of the implicit function will be important to define operations with implicit objects.

Invariance

Consider, without loss of generality, the boundary of an implicit object defined as the zero surface $F^{-1}(0)$ of an implicit function F. Then, the same surface is described by the function G:

$$G(\mathbf{x}) = \alpha F(\mathbf{x}) \tag{7.1}$$

where $\mathbf{x} = [x_1, \ldots, x_n] \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$.

If F is strictly positive and the implicit surface is $F^{-1}(c)$, $c > 0$, then

$$G(\mathbf{x}) = F(\mathbf{x})^{\beta} \tag{7.2}$$

also describes the same surface.

The surface $F^{-1}(0)$ is said to be embedded in both F and G. Note that although these two implicit functions describe the same surface, they do not induce the same metric.

Redundancy

Implicit functions are constructable. Given $F^{-1}(0)$ it is possible to find a function $G(\mathbf{x})$ such that $G^{-1}(0) \equiv F^{-1}(0)$.

If F is an algebraic function it is possible, at least theoretically, to factor it into irreducible components.

Level Contours

A level surface of the graph of F is also called an *isocontour* of F. The contours of F can be traversed by a function H defined as

$$H(\mathbf{x}) = F(\mathbf{x}) - \delta \tag{7.3}$$

where $\delta \in \mathbb{R}$.

Note that $H^{-1}(0) = F^{-1}(\delta)$ does not describe the same surface as $F^{-1}(0)$. The surface $H^{-1}(0)$ is an *offset surface* of $F^{-1}(0)$ in the metric induced by F.

7.3 Operations on the Range of F

The properties of the function F discussed above can be used to define unary operations on implicit objects.

Algebraic operations with the function F reflect in the implicit object altering its geometry or topology. This will be exploited in this chapter to define operations that modify the implicit object.

Note that some of these operations may change the implicit surface defined by F, therefore, it is possible that the regularity condition may be violated.

7.3.1 Density Change

When the function F is multiplied by a constant, or raised to a power, the algebraic distance induced by F is altered. This operation affects the density of the contour sections of F and is useful to control how a surface blends with other surfaces (Chapter 8).

$$\text{density}_M(d, F(\mathbf{x})) = d * F(\mathbf{x})$$
$$\text{density}_P(d, F(\mathbf{x})) = F(\mathbf{x})^d$$

where $d > 0 \in \mathbb{R}$. Note that $d > 1$ corresponds to a density expansion, pushing contours away from the surface; while $d < 1$ corresponds to a compression, pulling contours towards the surface.

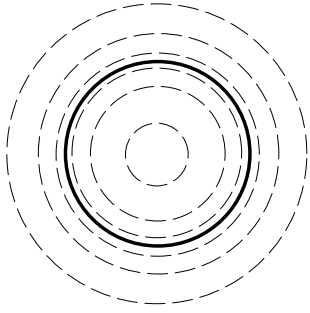


Fig. 7.2. Density change

Leaving the Surface Invariant.

The density change, depending on the value of c, modifies the function F without affecting the surface $F^{-1}(c)$.

If a surface S is defined as the level set $F^{-1}(0)$, then it is also implicitly defined by any function G:

$$G(x) = \alpha F(x),$$

where $\alpha \in \mathbb{R}$, $\alpha \neq 0$.

If $F : U \to (0, \infty)$ and S is defined as $F^{-1}(1)$, then it is also implicitly defined by any function G:

$$G(x) = F(x)^{\beta},$$

where β is a positive real number.

In these two cases, the surface S is said to be embedded in both F and G. Note that although these implicit functions describe the same surface, they are not induced by the same pseudo-metric.

7.3.2 Mapping between Canonical Forms

In the previous subsection, we have seen that the following pairs (R, v) of range and value

$((-\infty, +\infty), 0)$
$((0, +\infty), 1)$

have special properties in relation to the invariance of $F^{-1}(v)$ for $\text{Range}(f) \in R$.

There exists a simple map that can be used to convert between these two canonical forms as required. Given an arbitrary function $f : U \to (-\infty, \infty)$ the transformation:

$$G(x) = \exp(F(x)),$$

maps the range $(-\infty, \infty)$ into $(0, \infty)$ and maps the level surface $f^{-1}(0)$ into $h^{-1}(1)$.

This transformation has an inverse, defined by taking the logarithm

$$F(x) = \log(G(x)).$$

7.3.3 Complement

Consider a solid implicit object defined by $\mathcal{O} = F^{-1}((-\infty, b])$ or $\mathcal{O} = F^{-1}([a, +\infty))$. If the function F is multiplied by a negative number,

the solid object defined by F is replaced by its complement. This operation changes the polarity of F and is useful in combination with CSG operations (Chapter 14).

$$\text{complement}(F(\mathbf{x})) = -F(\mathbf{x})$$

Complementation reverses the sense of inside and outside of an implicit object.

7.3.4 Dilations and Erosions

When a constant is added to the function F, the contour defined by F changes. This operation produces a dilation or an erosion of the volume enclosed by $F^{-1}(0)$.

$$\text{offset}(c, F(\mathbf{x})) = F(\mathbf{x}) - c$$

where $c \in \mathbb{R}$. Note that $c < 0$ results in an offset surface nested inside the original surface and $c > 0$ results in an offset surface surrounding it.

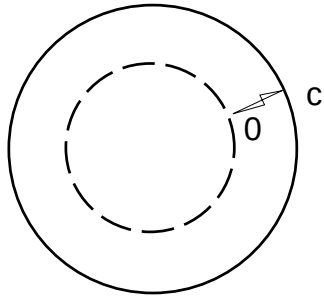


Fig. 7.3. Dilations and erosions

Note that when F is a distance function these operations are precisely the morphological operations with the structuring element as a ball.

7.4 Mappings of the Embedding Space

The space in which the implicit object is embedded can be transformed by a map $M : \mathbb{R}^n \to \mathbb{R}^n$. This transformation warps the domain of

the function F and as a consequence modifies the associated implicit object. The characteristics of M determine how the object is affected.

One way to get an intuitive feeling of how space mappings work is through the analogy of a rubber sheet. Consider an elastic sheet where figures are plotted. If we stretch it in one direction and then draw a circle on its surface, when the rubber is allowed to relax the circle becomes an ellipse. The circle has been squashed along the axis which the sheet was stretched. Note that this type of space transformations has the inverse effect on the embedded objects.

Although it is possible to do transformations in a different fashion, they are not so useful, as will become clear with the next example. Returning to the situation in our previous example, we could have drawn the circle before stretching the rubber sheet. This would have caused the circle to deform in the same direction as the sheet. But, suppose we want add another ellipse to the drawing by the same process. If we keep stretching the rubber and drawing subsequent deformations will affect the second as well as the first ellipse. This is probably not what was intended. Now, if we apply the previous scheme we will get better results. Stretch the rubber and draw a circle. Let the rubber relax: we have the first ellipse. Stretch the rubber in another direction and draw a second circle. This distorts the first ellipse, but when the rubber sheet is relaxed to get the second ellipse, the first one is restored! Space transformations can be applied to different objects without interference. See Figure 7.4.

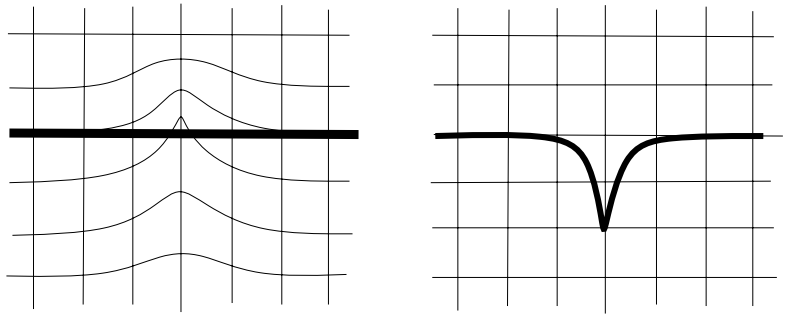


Fig. 7.4. Spatial mappings

7.4.1 Affine Transformations

An affine transformation is a map $T : \mathbb{R}^n \to \mathbb{R}^n$ defined by:

$$T(\mathbf{x}) = L(\mathbf{x}) + \mathbf{v}$$

where L is a linear transformation. L is represented by an $n \times n$ matrix and $\mathbf{v}$ is a n-vector.

T can be expressed more conveniently as a homogeneous transformation in real projective space $\mathbb{R}\mathbf{P}^n$. In this case T is represented by an $(n+1) \times (n+1)$ matrix and the vector $\mathbf{v}$ is incorporated in this matrix.

An important subset of the affine transformations is a map $R : \mathbb{R}^n \to \mathbb{R}^n$ such that $\|R(p) - R(q)\| = \|p - q\|$ for all $p, q \in \mathbb{R}^n$. This type of mapping preserves distance between points and is called a *rigid motion* or *isometry* of $\mathbb{R}^n$. R is represented by an $n \times n$ orthonormal matrix.

The most common examples of affine transformations are: *translation*, *scale*, *rotation*, *shear* and *reflection on a plane*. Any affine transformation can be obtained as a combination of these ones.

7.4.2 Deformations

A deformation of space is a bijective map $W : \mathbb{R}^n \to \mathbb{R}^n$ of, at least, class C^1. When W is a diffeomorphism (invertible with differentiable inverse) it maps surfaces into surfaces.

Kleck suggested an intuitive class of deformations based on space warps [Kle89]. These include operations to fold an object at a given plane, and to push (pull) it away (towards) a given point.

Another interesting class of global deformations was proposed by Barr [Bar84]. These include taper, bend, and twist operations.

A free-form deformation technique was introduced in [SP86]. This involves a mapping of $\mathbb{R}^n$ through n-variate tensor product Bernstein polynomials.

A deeper investigation of space deformations would lead to the study of elasticity theory and physically-based methods. In what concerns modeling and animation with implicit objects this is still an open area of research.

7.5 Operations on the Domain of F

A transformation is essentially a change of coordinate systems. This paradigm works in the following way. The function F is defined in a canonical space called *local coordinate system* of the implicit object. This space is related to a *global coordinate system* where all objects coexist by the transformation M that maps canonical to global space and its inverse M^{-1} that maps global to canonical space.

In general, to transform an implicit object the argument $\mathbf{x}$ of F has to be converted to the local coordinate system through the map M^{-1} before F is evaluated, that is we evaluate $F(M^{-1}(p))$. The result of any computations performed in the local coordinate system have to be mapped back to the global coordinate system through the map M.

7.5.1 Analytical Transformation of F

In some particular cases, it is possible to transform the description of the implicit object, incorporating the M directly into the function F and avoiding the change of coordinate systems. An example is the case of homogeneous transformations and quadrics [Bli84].

7.5.2 Transformation of Points

A map $M : \mathbb{R}^n \to \mathbb{R}^n$ transforms a point $p = [x_1, \ldots x_n] \in \mathbb{R}^n$ in a point $\bar{p}$ such that $M(p) = \bar{p}$ or

$$t(x_1, \ldots, x_n) = [\bar{x}_1, \ldots, \bar{x}_n]$$

where

$$\begin{aligned} \bar{x}_1 &= t_1(x_1, \ldots, x_n); \\ &\vdots \\ \bar{x}_n &= t_n(x_1, \ldots, x_n); \end{aligned}$$

The functions m_i, $i = 1, \ldots, n$ are called components of the function M.

As we have seen, when M is an affine or a projective transformation, we can represent M by a, $(n+1) \times (n+1)$, matrix M, and and points in homogeneous form by an $n+1$ vector $p = [x_1, \ldots x_n, x_n + 1] \in \mathbb{R}\mathbf{P}^n$. In this case, to compute the transformation we simply use the matrix-vector multiplication $\bar{p} = Mp$.

7.5.3 Transformation of the Tangent Plane

The derivative dM_p of M at a point p is a linear transformation that constitutes a first order approximation of M at p. The matrix of dM also denoted by J is called the *jacobian matrix*

$$J = dM = \begin{bmatrix} \frac{\partial M_1}{\partial x_1} & \cdots & \frac{\partial M_n}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial M_1}{\partial x_n} & \cdots & \frac{\partial M_n}{\partial x_n} \end{bmatrix}$$

J transforms tangent planes and can be used to derive the transformations for normal vectors. If $\mathbf{v}$ is a normal vector then the transformed normal vector $\bar{\mathbf{v}}$ is

$$\bar{\mathbf{v}} = \det J (J^{-1})^T \mathbf{v}$$

Another way to see this is to notice that we can represent a plane n as a row vector $n = (a, b, c, d)$, corresponding to the coefficients of the implicit equation of this plane. Thus, all points $p \in n$, satisfy the equation

$$\{p = (x, y, z, 1) \quad | \quad ax + by + cz + d = 0\}$$

that can be concisely formulated using the inner product, $\langle n, p \rangle = 0$.

When we apply a transformation given by some projective transformation matrix W to the plane n, the condition that a point p belongs to n corresponds, after the transformation by M, to the equation

$$(nW^{-1})(Wp) = 0$$

That is, the transformed point Wp lies on the transformed plane nW^{-1}.

Therefore, we see that to transform a tangent plane (represented as a vector), we need to use the transpose of the inverse matrix

$$\bar{n} = (W^{-1})^T n$$

Note that if the matrix is orthogonal $(W^{-1})^T = W$. Only in this case we can transform tangent planes as points.

8. Combining Implicit Objects

This chapter investigates how implicit objects are combined in a modeling system. We analyze binary operations used to create compound objects. These operations will form an algebra of implicit objects.

8.1 Compound Objects

Implicit objects can be joined to form a compound implicit object. This is accomplished by combining, according to certain rules, the corresponding functions describing the individual objects.

Algebraic operations with the implicit functions defining a set objects produce another implicit function that defines a compound object. These operations with the implicit function correspond to composition operations with the implicit objects.

It is worth noting that the algebraic structure defining the implicit objects can be exploited in many ways in a modeling system.

8.1.1 Proper Functions

An implicit function $F : \mathbb{R}^n \to \mathbb{R}$ associated with an implicit object $\mathcal{O}$ and a regular value A is called *proper* if its inverse image $F^{-1}(A)$ is a valid implicit object.

Given a proper function F, $F(\mathbf{x}) \in (-\infty, \infty)$, then αF and $F + c$ are also proper functions. Given a strictly positive proper function F, then F^{β} is also a proper function.

8.1.2 Closure Properties of F

The set of real valued functions used to describe implicit objects is closed under the following operations:

- Sum: $F_1 + F_2$;
- Product: $F_1 * F_2$;
- Composition: $F(F_1, \ldots, F_k)$;
- Maximum: $\max(F_1, F_2)$;
- Minimum: $\min(F_1, F_2)$;

In other words, if two functions F_1 and F_2 are proper then their combination using any of these operations is also a proper function.

Note that the unary operations, αF and F^α, defined in the previous chapter are respectively particular cases of the n-ary operations $F_1 + \cdots + F_n$ and $F_1 * \cdots * F_n$, when $\alpha = n$ is a positive integer.

8.2 Boolean Operations

An effective method to build complex objects from simple ones is through CSG [1] operations. This scheme is based on regularized boolean operations on point sets. A CSG object is constructed from the union and intersection of primitive objects. Other operations, such as difference are defined using complementation [Req80].

8.2.1 Functional Description

Let P_i be a set of *primitive functions* from $\mathbb{R}^n$ to $\mathbb{R}$ of class C^1. The set of functions S_j generated by P_i is defined as follows:

1. $P_i \subset S_j$.
2. $F \in S_j \Rightarrow -F \in S_j$.
3. $F_1, F_2 \in S_j \Rightarrow \max(F_1, F_2) \in S_j$.
4. $F_1, F_2 \in S_j \Rightarrow \min(F_1, F_2) \in S_j$.

where $i = 1, \ldots, k$, and $j = 1, \ldots, l$.

In other words, if $F \in S_j$ is not a primitive function, then it is constructed from primitives by max and min operations [Ric73].

[1] Constructive Solid Geometry

8.2.2 Implicit CSG Objects

An implicit CSG solid is defined as any set of points in $\mathbb{R}^n$ which satisfy $F(\mathbf{x}) \leq 0$ for some $F \in S_j$. The *boolean operations* are defined as:

$F_1 \cup F_2 = \min(F_1, F_2)$.
$F_1 \cap F_2 = \max(F_1, F_2)$.
$F_1 \setminus F_2 = F_1 \cap \overline{F_2} = \max(F_1, -F_2)$.

Figure 8.1 shows an example of these operations.

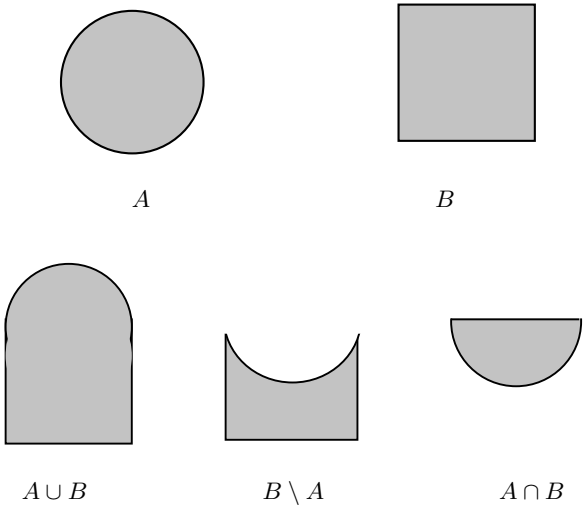


Fig. 8.1. CSG operations

A CSG model is usually represented as a tree structure in which internal nodes are associated with set operations and the leaf nodes are associated with primitive objects. This will be discussed in more detail in Chapter 14.

8.2.3 Differentiable Boolean Operations

One drawback of boolean operations is that the functions *max* and *min* are not differentiable in general. Consequently implicit objects constructed with these operations are not differentiable along of the intersection of surfaces. Although this problem does not invalidate the usefulness of previous scheme, it would be desirable to have a

differentiable approximation of boolean operations. One solution for this problem is proposed in [Ric73].

Assuming strictly positive functions $G(\mathbf{x}) \in (0, \infty)$, define implicit objects such that $G(\mathbf{x}) = 1$, $G(\mathbf{x}) < 1$ and $G(\mathbf{x}) > 1$ correspond respectively to the boundary, interior and exterior of objects. [2]

Given $G_1, \ldots, G_k$ differentiable functions as above, the following limits are true:

$$\lim_{p \to \infty} (G_1^p + \ldots + G_k^p)^{\frac{1}{p}} = \max(G_1, \ldots, G_k) \tag{8.1}$$

$$\lim_{p \to \infty} (G_1^{-p} + \ldots + G_k^{-p})^{-\frac{1}{p}} = \min(G_1, \ldots, G_k) \tag{8.2}$$

Ricci's formulation corresponds, essentially to taking the L_n norm on $E \times E$ as an approximation to the maximum, or L_∞, norm on the ring E of expressions defining the implicit function F. Since $(f_1^n + f_2^n)^{1/n}$ approaches $\max(|f_1|, |f_2|)$, as $n \to \infty$.

$\partial(G_1^p + \ldots + G_k^p)^{\frac{1}{p}}/\partial x_i$ and $\partial(G_1^{-p} + \ldots + G_k^{-p})^{-\frac{1}{p}}/\partial x_i$ are approximations of the partial derivatives $\partial G_j/\partial x_i$ of the component functions G_j in the vicinity of the curves of intersection.

In [TdMG89], it is shown that the convergence in (8.2.3) and (8.2.3) is uniform up to derivatives of order r, that is

$$\frac{\partial^r (G_1^p + \ldots + G_k^p)^{\frac{1}{p}}}{\partial x_1^{r_1} \cdots x_n^{r_n}} \to \frac{\partial^r G_j}{x_i}$$

outside of an open neighborhood of the intersection points.

Note that for small values of p these formulations result in a blend of the implicit objects. We will return to this observation in Section 8.3.

8.2.4 *R*-Functions

A function $C : \mathbb{R}^n \to \mathbb{R}$ is called an R-function if its sign is completely determined by the sign of its arguments. Intuitively, $C(F_1, F_2, \ldots, F_n)$ acts as a Boolean switching function, changing its sign only when its arguments F_i change their signs.

R-functions are a generalization of CSG implicit objects. They constitute n-ary operators that algebraically combine implicit functions

[2] This is not a big restriction in practice, since any implicit object $F^{-1}(0)$, described by an arbitrary function $F(\mathbf{x}) \in (-\infty, \infty)$ could also be represented by $H^{-1}(1)$ where H is a function defined in terms of F by $H(F, \mathbf{x}) = \exp(F(\mathbf{x}))$

similarly to the way point set operators combine solid objects. This is due to the fact that every formal logical sentence has a corresponding class of R-functions. It is immediate to see from their definition that the sign of the resulting function is determined by the truth table of the logical sentence.

Just as any logical sentence can be written as a composition of three logical operations $\neg$, $\wedge$, $\vee$, every R-function can be written as a composition of the three R-functions corresponding to these operators. Without loss of generality we can consider only the unary negation $\neg$, and the binary disjunction $\wedge$ and conjunction $\vee$. These logical operators are sufficient to construct any n-ary logical sentence.

It should be clear that R-functions are not unique. Therefore, we can define various systems of R functions that exhibit different properties. The negation operation $\neg$ is naturally accomplished by changing the sign of the function f. Three systems of R-functions, R_α, R_0^m, and R_p, have been identified by the founder of the theory, V. L. Rvachev, and used in various applications [Sha88]:

$$R_\alpha(F_1, F_2): \quad \frac{1}{1+\alpha}\left(F_1 + F_2 \pm \sqrt{F_1^2 + F_2^2 - 2\alpha F_1 F_2}\right) \tag{8.3}$$

where $\alpha(F_1, F_2)$ is a symmetric function such that $-1 < \alpha(F_1, F_2) \leq 1$.

$$R_0^m(F_1, F_2): \quad \left(F_1 + F_2 \pm \sqrt{F_1^2 + F_2^2}\right)\left(F_1^2 + F_2^2\right)^{\frac{m}{2}} \tag{8.4}$$

where m is any even positive integer.

$$R_p(F_1, F_2): \quad F_1 + F_2 \pm \left(F_1^p + F_2^p\right)^{\frac{1}{p}} \tag{8.5}$$

where p is is any even positive integer.

In each case, choosing the $(+/-)$ sign determines the type R-function operator. For the R-disjunction we use $(+)$ while for the R-conjunction we use $(-)$.

The most popular system of R-functions is R_α. The simplest class of this family is defined by setting $\alpha = 1$. Note that the resulting functions correspond to the implicit CSG boolean operators, with the R-conjunction $\min(F_1, F_2)$ and the R-disjunction $\max(F_1, F_2)$.

An analysis of the differencial properties of R-functions can be found in [ST99].

8.3 Blending Operation

A *blend* is a surface which forms a smooth transition between intersecting surfaces. As we will see, the implicit form is particularly well suited for the construction of blends.

8.3.1 Developing a Blend

A blend can be conceived as a composition of functions. Define a blending function $B : \mathbb{R}^k \to \mathbb{R}$ that takes as its argument a list of functions defining surfaces to be blended $B(F_1(\mathbf{x}), \ldots, F_k(\mathbf{x}))$. This function creates a transition between the countour lines of those surfaces. Note that the blend itself is given in implicit form. Consequently, B produces a new implicit surface in which the other surfaces are embedded.

This approach serves the purpose of separating the intrinsic characteristics of the blend from the effects of functional composition. An outline of the main steps to develop implicit blending for arbitrary solid objects is presented in [RO87]:

- Define a blend model in two dimensions.
- Generalize to n dimensions by composition of functions.
- Extend to multiple unions and intersections of primitives.

8.3.2 Linear Blend

The simplest type of blend is the linear blend. It is given by:

$$B(F_1(\mathbf{x}), \ldots, F_k(\mathbf{x})) = \sum_{i=1}^{k} F_i(\mathbf{x}) - 1$$

The two-dimensional model of this blend is the equation $y = 1 - x$ of a line at 45-degree angle of the coordinate axis.

One characteristic of the linear blend is that blending surface is actually an offset of the original surfaces. This type of blend is the basis of several implicit primitive objects and will be considered in greater detail in Chapter 11.

Figure 8.2 illustrates the construction of a linear blend.

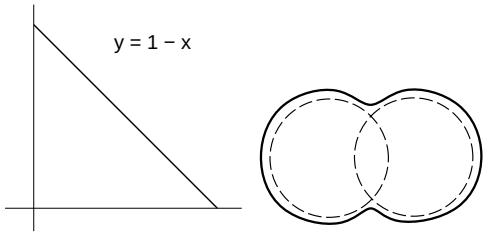


Fig. 8.2. Linear blend

8.3.3 Hyperbolic Blend

The hyperbolic blend [Kle89] is given by:

$$B(F_1(\mathbf{x}), \ldots, F_k(\mathbf{x})) = \prod_{i=1}^{k} F_i(\mathbf{x}) - 1$$

The two-dimensional model of this blend is the hyperbolic equation $y = 1/x$. The hyperbola puts a smooth curve joining the coordinate axis.

This blend is also an offset blend. Another of its characteristics is that if the original surfaces intersect, then the blending surface will have more than one connected component (as in the hyperbola). The external surface covers the union of the volume enclosed by the original surfaces and the internal surface covers the intersection of their volume.

Figure 8.3 illustrates the construction of a hyperbolic blend.

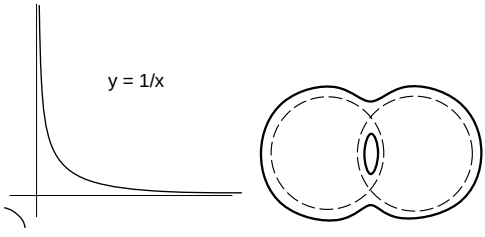


Fig. 8.3. Hyperbolic blend

8.3.4 Super-Elliptic Blend

The super-elliptic blend [Roc89] is given by:

$$B(F_1(\mathbf{x}),\ldots,F_k(\mathbf{x})) = 1 - \left(\sum_{i=1}^{k} [1-\alpha_i F_i(\mathbf{x})]_+^{\beta}\right)^{\frac{1}{\beta}}$$

where $[\star]_+ = \max(0,\star)$. The parameters α and β are used to change the shape of the blend. α_i bias the blend towards the primitive i and β controls the overall tightness of the blend.

The two-dimensional model for this blend is a super-elliptic arc that touches the coordinate axis. It is a generalization of the circular arc of radius 1 defined by the equation $1 - \left[(1-x)^2 + (1-y)^2\right]^{1/2}$.

This blend, differently from the previous ones, interpolates the primitive surfaces involved. To understand how it works, consider a set of surfaces described by functions of the form $f = 0$. Then, take the external 1-contour of their complement. The new surfaces, given by $1-f = 0$, are the internal 1-contours of the f_is. Now construct a linear blend of the external 1-contour of their complement. The resulting blending surface interpolates the original surfaces because the external 1-contours of this function corresponds to the 0-contours of the original functions. The complementation also restores inside-outside sense we started with. This can be confirmed from the observation that this operation is the identity when applied to only one object — $1-(1-f) = 0 \Rightarrow f = 0$.

Figure 8.4 illustrates the construction of a super-ellipitc blend.

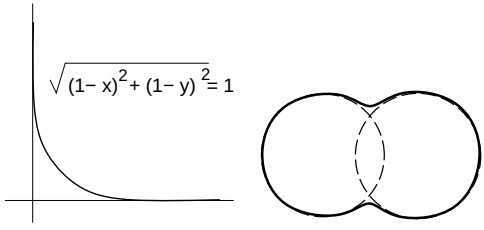


Fig. 8.4. Super-elliptic blend

8.3.5 Convolution Blend

A blend can be constructed from a convolution operator. The basic principle is quite intuitive: the convolution of a function with a low-pass filter results in a smoother version of this function. When applied to a collection of primitive functions this operation will produce a blend of them. The shape of the convolution kernel defines the blend.

[BS91] propose a method based on three-dimensional convolution to construct a distance function that is used for blending of implicit objects. Because convolution is a linear operator, the convolution of a sum of primitives is equal to the sum of convolutions of the primitives. The implementation of the method exploits the separability property of convolution filters to reduce the required computations.

8.4 Global and Local Blends

Blends may be applied globally to implicit objects or locally to a particular intersection of primitives.

8.4.1 Blends as Boolean Operations

It is possible to characterize global blends as an approximation of the CSG boolean set operations. The union operation corresponds to an *added material blend*, and the intersection operation corresponds to a *subtracted material blend*.

We will show that the differentiable approximation of boolean operations discussed in Section 8.2 is equivalent to the super-elliptic blend.

In subsection 7.1, implicit surfaces where defined as the inverse image $G^{-1}(1)$. The same surfaces could be defined as $F^{-1}(0)$ for $F(\mathbf{x}) = G(\mathbf{x}) - 1$.

The expression in equation 6.1 can be rewritten as

$$\left[\sum_{i=0}^{k}(F_i+1)^p\right]^{\frac{1}{p}} - 1$$

since $G(\mathbf{x}) = F(\mathbf{x}) + 1$. For uniformity reasons the composite surface is defined as the inverse image of 0.

The difference between this expression and the one for the super-elliptic blend is that the former complements the primitive and the composite surfaces. Thus, in what concerns the blending implicit surface these expressions are totally equivalent.

8.4.2 Local Blends

A local implicit blend is defined by restricting a global blend in one of two ways: it can be prescribed only to certain pairs of surfaces in a CSG structure; the other alternative is to limit the domain in which the blending operation is performed.

9. Computational Methods

This chapter considers the various ways of performing actual computations with implicit objects. In the previous chapters, we have seen different types of operations with implicit objects that need to be implemented in a computer. This is done using symbolic and numerical operations which are the building blocks of higher level computational methods.

Since the implicit object $\mathcal{O}$ is defined indirectly by an implicit function F, we do not have a direct way of generating the set of points $F^{-1}(A)$ from it [1]. Therefore, we must settle for solutions which generate a subset $S \subset F^{-1}(A)$ of points that are samples of the implicit object $\mathcal{O}$.

In some applications, we only need local geometric information about $\mathcal{O}$, and the result of this sampling procedure provides all the data required to solve the problem. In other situations, we really need global information about the object, and is necessary to reconstruct $F^{-1}(A)$ in a piecewise manner from the samples. This involves an additional structuring step which supplies the topological data to link the samples.

Sampling and structuring are the basic procedures used in the computation of implicit objects. Both processes are intimately related to space decomposition schemes. (See Chapter 4). First, because they often exploit the coherence of embeddings of implicit objects in the ambient space. Second, because implicit objects themselves can be interpreted as abstract spaces which are decomposed using structuring and sampling.

If the implicit function is very complicated, the computational effort can be too high and even sampling the implicit object becomes impractical. In those cases is common to substitute the implicit func-

[1] This can only be done if we can find a global parametrization $f(u) = x$ of this set which gives these points explicitly. As we have seen in Chapter 3, it is not possible, in general, to construct such a parametrization.

tion by a suitable approximation with which it is easier to compute samples.

Below we will talk more about sampling and structuring, discussing some aspects of the mathematical and computational theory behind them. In the next chapter, we will address the problem of reconstructing the implicit object using these two processes.

9.1 Numeric and Symbolic Computation

Although it is possible to perform extensive symbolic computations with certain classes of implicit objects (such as those defined by algebraic implicit functions), numeric computations are always necessary in one way or another. For this reason, we will concentrate here on numerical methods.

9.2 Interval Arithmetic

Interval arithmetic provides a robust way to perform numerical calculations and gives, at the same time, a sound basis to implement approximate computations within a desired accuracy [Moo79]. This is particularly important in the context of applications of implicit objects [Sni92], [Duf92].

In practice, it is not possible to work with real numbers on a computer. We really deal with floating-point numbers that approximate real numbers to a limited precision. For this reason, the machine implementation of arithmetic operations suffers from "round-off" errors. These errors are a major source of flaws in computer graphics algorithms. Interval arithmetic brings this problem under control. First, the concept of a real number is substituted by the concept of an interval that is represented by a pair of "real" numbers, i.e., its endpoints [2].

Interval arithmetic defines how to operate with these entities, generalizing ordinary arithmetic operations:

An interval is a closed bounded set of real numbers $[a, b] = \{x : a \leq x \leq b\}$. If X is an interval, its endpoints are denoted by $\underline{X}$ and $\overline{X}$,

[2] Note that when the endpoints of an interval are the same we have a degenerate interval which can be identified with a real number. Therefore, real quantities are a subset of interval quantities

with $\underline{X} \leq \overline{X}$. An n-dimensional interval vector is an ordered n-tuple of intervals $(X_1, \ldots, X_n)$.

The basic interval arithmetic operations are:

$$\begin{aligned} X + Y &= [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}] \\ X - Y &= [\underline{X} - \underline{Y}, \overline{X} - \overline{Y}] \\ X \times Y &= [\min(\underline{XY}, \underline{X}\overline{Y}, \overline{X}\underline{Y}, \overline{XY}), \max(\underline{XY}, \underline{X}\overline{Y}, \overline{X}\underline{Y}, \overline{XY})] \\ X/Y &= [1/\underline{Y}, 1/\overline{Y}] \end{aligned}$$

We can use the interval arithmetic rules to compute bounds on the value of a rational expression $F(x)$ inside any interval X.

A function $F(X)$ composed using interval arithmetic operations is an *interval extension* of the corresponding real function. Furthermore, F is *inclusion monotonic*. That is, if $X \subseteq Y$, then $F(X) \subseteq F(Y)$. Consequently, $x \in X$ implies that $F(x) \in F(X)$.

When applied to implicit functions these techniques are useful for the implementation of most tasks involving numerical computations. To mention just a few, this is the case of CSG operations, polygonization, rendering, collision detection and calculation of integral properties of implicit objects.

A flavor of the application of interval arithmetic in the above framework is given in the following example: Consider a solid implicit object $\mathcal{O}(F, 0)$: If $\overline{F}(X) < 0$, we know that all points $x \in X$ are inside the object. Similarly, if $\underline{F}(X) > 0$, they are all outside. If $\underline{F}(X) \leq 0 \leq \overline{F}(X)$, we can guess that the implicit surface $F^{-1}(0)$ intersects X and, therefore, X deserves a closer analysis. This observation is the basis of powerful robust adaptive algorithms.

9.3 Root Finding

In order to get samples of an implicit surface it is necessary to compute solutions for $F(\mathbf{x}) = c$, where $\mathbf{x} = [x_1, \ldots, x_n] \in \mathbb{R}^n$. This is equivalent to finding the roots of the equation $F(\mathbf{x}) - c = 0$. Root finding is a well understood problem, and can be done analytically or numerically, depending on the complexity of the implicit function F and the class of point set desired.

We want to find the real roots of the equation $f(x) = 0$, where f is a piecewise continuous function of x. If x is a scalar variable, the problem simplifies a great deal and there are efficient numerical techniques to solve it. If $\mathbf{x}$ is a vector variable, the problem becomes more

complicated, but, as we will see in the next section, is often possible to reduce its dimensionality, solving several times a reformulated lower dimensional version of the original problem. In this section we will concentrate on techniques to solve the one-dimensional case which, in principle, generalizes to higher dimensions.

Numerical root finding involve the following steps:

- Use some knowledge of the function f to get an approximate solution of $f(x) = 0$.
- Improve the previous solution to the required precision.

There are two main classes of methods to compute the roots of $f(x)$: interval subdivision and fixed-point search.

9.3.1 Interval Subdivision Methods

Interval subdivision methods use the fact that if f is continuous and there is an interval $[a, b]$ where $f(a)$ and $f(b)$ have opposite signs, then f has at least one root in this interval. The first step of this method is to find an interval $[a, b]$ containing the solution. The second step is to refine this interval until it bounds tightly the solution. This is very much in the spirit of the interval arithmetic techniques discussed in the previous section.

Examples of interval subdivision methods are the bisection and the regula-falsi methods. They both refine the interval $[a, b]$ by splitting it into two subintervals $[a, c]$, $[c, b]$ and discarding the one for which the condition $f(x)f(c) < 0$, $x = a, b$, is not satisfied. The difference between them is essentially in the way c is computed.

Bisection Method This is the simplest numerical root finding method. It computes c as the mid-point $c = \frac{a+b}{2}$. Note that it does not use any additional knowledge about f to bracket the interval.

Regula-Falsi Method This method tries to obtain a faster conversion of the process by subdividing the interval $[a, b]$ at a point c closer to the root based on the values of f at extremes of the interval. The assumption behind the regula-falsi method is that if $|f(a)|$ is large and $|f(b)|$ is small, then the root is probably closer to b. One way to implement this is substituting f in $[a, b]$ by an affine approximation $\overline{f}$ given by the linear interpolation of $f(a)$ to $f(b)$. The root of $\overline{f}$ can be computed explicitly:

$$c = a - \frac{f(a)(b-a)}{f(b) - f(a)}$$

9.3.2 Fixed-Point Methods

Fixed-point methods explore the fact that there are some contraction mappings M for which the sequence x_i defined by the recursion $x_{i+1} = M(x_i)$ converges to a fixed-point [3] x that is a root of f, (i.e. $f(x) = 0$). The first step of this method is to obtain an initial value for the sequence x_i. The second step consists in iterating the recursion formula $x_{i+1} = M(x_i)$ until the fixed point is reached to a desired precision. Examples of fixed-point methods are Newton's method and variants of it.

Newton-Rapson The idea behind Newton's method is to use the derivative of f at a point x_i to get a new point x_{i+1} closer to the root. The tangent to f at x_i is given by $L(x) = f(x_i) + f'(x_i)(x - x_i)$. We choose as the new point x_{i+1} for which $L(x) = 0$, obtaining

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

9.4 Sampling Implicit Objects

In order to capture the geometry of an implicit object it is necessary to sample the implicit function defining it. There are several alternatives to do this, depending on the specific task one wants to execute. Basically, we simplify the original problem of root finding and instead of trying to compute the k-dimensional point set $\mathcal{O} = \{x : F^{-1}(A)\}$, $k = 2, 3$, we compute a family of point sets $S_i \subset \mathcal{O}$ of lower dimension.

For example, in a ray tracing application the intersection of a ray with the implicit surface must be calculated. This is done by finding the common roots of a line equation with the implicit function.

The sampling techniques can be classified according to the dimension of the family of point sets $\{S_i\}$ generated by the process. Thus, *point sampling* corresponds to families of zero-dimensional samples; *curve sampling* corresponds to families of one-dimensional samples. Additionally, we have *volume sampling* which corresponds to families of three-dimensional cells approximating an implicit solid $\mathcal{O}$.

[3] A fixed point of of a mapping M is a value of x such that $M(x) = x$

9.4.1 Point Sampling

Point sampling processes generate a collection of points lying on an implicit surface, e.g. points x satisfying $F(x) - c = 0$. These processes essentially compute the intersection points of a curve in $\mathbb{R}^3$ with $F^{-1}(c)$.

Ray Casting Methods. Ray casting methods intersect $F^{-1}(c)$ with a family of straight lines l_i in $\mathbb{R}^3$. This amounts to substituting the equation for l into F and solving $F(l) - c = 0$. A line can be defined by the parametric equation $x = o + td$ that when substituted for x in the implicit function will give a function $G(t)$ in one variable. We want to find the values of the parameter t for which $G(t) = 0$. If the implicit function F is of sufficiently low degree the above equation has a closed-form solution. When an analytic solution is not available we have to compute the roots numerically by one of the methods defined in Section 9.3.

Continuation Methods. Continuation methods start with a point p lying on the implicit surface $F^{-1}(c)$ and obtain new points on the surface using the gradient field $\text{grad}F(x) = \frac{\partial F}{\partial x_i}$. Since the gradient gives a vector pointing away from the surface, new points can be found by taking an orthogonal complement of $\text{grad}F(p)$ and integrating along each direction in this complement. This method is also called *moving frame* method [AG90].

Physically-Based Methods. Physically based methods also sample points on an implicit surface by integrating ordinary differential equations related to the gradient field of F [dF92]. The main difference between these two methods is that the equations are based on a modified gradient field, and do not require an initial point on the surface (which might not be available). Samples are obtained through a particle system driven by a force field. The motion of this system is derived from the potential function $|F|$. Particles will seek equilibrium positions on the manifold $F^{-1}(0)$ because these are positions of minimum potential energy.

We can give two interpretations to this potential force field: *kinematic* and *dynamic*. In the kinematic interpretation F describes velocity in terms of position:

$$\frac{dx}{dt} + \text{sign}(F)\text{grad}F = 0,$$

In the dynamical interpretation F is a force field in a dissipative medium:

$$\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + \text{sign}(F)\text{grad}F = 0,$$

where γ is a positive real number representing friction proportional to the velocity.

The dynamical approach has the advantage that friction guarantees that the particles will tend to a rest position on $F^{-1}(c)$ [dF92]. A similar approach has been used in numerical analysis for solving systems of nonlinear equations [IPZ79].

9.4.2 Curve Sampling

One-dimensional samples of an implicit object are a set of curves belonging to the boundary of the implicit object. Curve sampling can be obtained by intersecting the implicit surface with a family of surfaces.

Surface intersection is a difficult problem and only in a few particular cases it is possible to obtain a closed from solution. Therefore, in general, the intersection has to be computed numerically, and approximated by a set of connected point samples along the curve.

In spite of all the difficulty, curve sampling is an important operation that must be performed in many applications with implicit objects. Some examples are: the computation of silhouette curves for rendering purposes [SZ89]; and boundary evaluation of CSG implicit objects [Hof89]. On the other hand, sometimes curve sampling arises naturally in some applications. This is the case of volume data produced by certain types of sensor devices, such as in CT (Computerized Tomography) and MRI (Magnetic Resonance Imaging).

9.4.3 Volume Sampling

Volume sampling is achieved by intersecting an implicit solid with a family of three-dimensional cells. This can either be used to calculate volumetric properties of the implicit object or to construct a spatial enumeration associated with the implicit object.

9.5 Structuring Implicit Objects

The space decomposition schemes described in Chapter 4 can be employed to build piecewise descriptions of a subset of the ambient space

associated with the domain of the implicit function; or to build piecewise descriptions of the implicit object and its boundary. An important aspect is the connectivity structure linking the different elements of the decomposition. This structure is given by the adjacency relationship between geometric entities, and defines the topology locally and globally.

The first case employs a space-based decomposition and the structuring process is driven by the topology of the ambient space, while the second case uses an object-based decomposition and the structuring process is driven by the topology of the implicit object. In both cases, we may need to sample the geometry of the implicit object to guide the structuring process.

Note that besides the adjacency information, each cell could also store several other local attributes of the object.

9.5.1 Space-Based Structures

These structures are used mainly as auxiliary representations in order to facilitate computation with implicit objects.

All the space decomposition schemes for $\mathbb{R}^3$ can be used for this purpose.

Most algorithms favor uniform decompositions of space, such as those derived from a rectangular lattice, because their computational simplicity.

Adapted and hierarchical space decompositions produced by refinement operations are also very important.

9.5.2 Object-Based Structures

These structures can be used either to subdivide the volume defined by an implicit object or its bounding surface.

In the case of surfaces, the approximation defines a piecewise manifold that has the same topology of the implicit object and is a good approximation to its geometry.

In the case of volumes, the decomposition has internal elements requiring a non-manifold structure to represent it.

Normally, the subdivision will be a piecewise approximation of the original object which can be linear or higher order.

A common approximation is a polyhedral decomposition given by a triangulation of the solid object [BdF90]. Polyhedral tesselations

should be built and manipulated using *Euler* operators to ensure that their consistency is maintained.

It is usually desirable that triangulations are well adapted to the geometry of the object and composed of very regular cells. This is critical to prevent numerical instabilities in some types of computation, such as finite element analysis [TWM85]. In [VdMG91b] a physically-based method is presented to create regular triangulations of implicit objects.

9.5.3 Hybrid Structures

Hybrid structures represent at the same time a decomposition of the domain of the implicit function, as well as a decomposition of the implicit surface. The extended octree representation is an example of hybrid structure which allows the description of general solids [Nav89]. This structure is an octree that encodes at each leaf node information of the boundary of the solid. It is suited to a variety of geometrical operations including boolean set operations.

This page intentionally left blank

10. Approximating Implicit Objects

This chapter studies methods for creating discrete representations of implicit objects. In general it is difficult to compute directly the implicit object $F^{-1}(A)$ associated with a function defined by $F : U \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$. This is equivalent to solving the "family" of equations $F(x_1, \ldots, x_n) = p,\ p \in A$. Unless we impose some restrictions on the function F and the set A, the solution set $F^{-1}(A)$ can be a very complicated subset of U. For this reason it is often necessary to employ indirect computational methods and resort to approximations.

The methods to compute approximations to $F^{-1}(A)$ usually involve two main processes: sampling the implicit object and structuring the samples to form a piecewise description of the object and reconstruction of the object from the samples. The sampling process deals with geometric information; the structuring process deals with topological information and the reconstruction process tries to get a geometric description of the object. These procedures can be combined to produce piecewise descriptions of implicit objects.

Another good strategy for manipulating implicit objects more effectively consists in using coarser approximations to them. One such approximation is to approximate them by spatial subdivision enumeration. This auxiliary structure helps in the execution of operations that are difficult to perform on the implicit representation alone.

Below we will show how to construct these approximations of the domain and the boundary of implicit objects.

10.1 Structuring and Sampling

There are two ways to produce approximations of implicit objects through sampling and structuring. One performs sampling before structuring; and the other performs structuring before sampling.

The former approach computes a set of samples from the implicit function and attempts to derive from them an approximation that faithfully represents the object. This problem can be viewed as a reconstruction from scattered data.

The latter approach structures the domain of the implicit function and uses this structure to generate an approximation of the object.

In general, the approximation is a cell decomposition which gives a piecewise parametrization of the implicit object. The order of the approximation is determined by the geometry of each cell. In this chapter we will discuss piecewise linear approximations of implicit objects, in this case each cell is affine. Higher-order approximations are often desirable (piecewise cubics cells are particularly attractive because of their flexibility and relatively low degree).

10.2 Polygonization Methods

Piecewise linear (PL) methods to approximate implicit objects generate a decomposition of the domain of the implicit function, as well as a piecewise linear parametrization associated with it. Since the latter is a polygonal approximation to the object's boundary, it is also known as *polygonization method*.

The problem of finding polygonal approximations to a manifold can be compared to the problem of lapidating a precious stone: we have to carve planar faces with a minimum loss of material. Mathematically, it can be stated in the following way:

Polygonization Problem. *For a given surface M, find a polygonal surface $\widetilde{M}$ that approximates M and has the same topology.*

Since this problem involves sampling, it is a very delicate task as illustrated in Figure 10.1

We can define precisely the polygonization problem above as follows:

- The polygonal surface $\widetilde{M}$ is a combinatorial manifold (Chapter 3);
- That $\widetilde{M}$ and M have the same topology means that there exists a homeomorphism $h\colon M \to \widetilde{M}$;
- That $\widetilde{M}$ approximates M means that there is a real number $\varepsilon > 0$ such that $d(p, h(p)) < \varepsilon$, where d is the distance in the ambient space. We should note here that finer, higher order approximations are sometimes required.

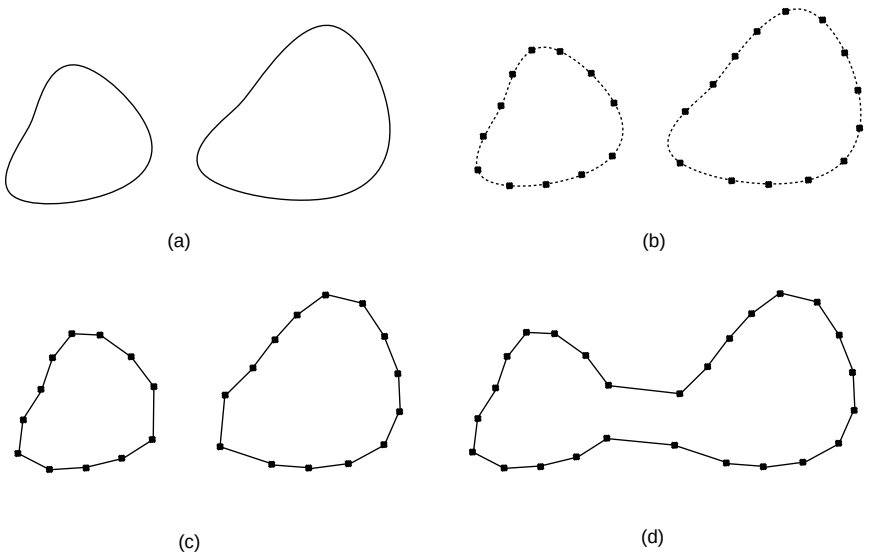


Fig. 10.1. (a) Implicit surface geometry; (b) sampling; (c) correct reconstruction using a piecewise linear approximation; (d) wrong reconstruction.

From our point of view the polygonization problem should be posed into two steps:

- Existence of the polygonal approximation;
- Algorithm to compute the polygonal approximation.

The importance of the existence of the approximation is twofold: Besides the guarantee that the problem has a solution it should, ideally, furnish a clue of how to devise an algorithm, and also indicate its robustness. In this section we are interested in the computation of the polygonal approximation for implicit surfaces, but we will first discuss the existence of the polygonization.

10.2.1 Existence of a Polygonization

The existence of the approximation is given by a classical theorem:

Theorem 10.1. *If M is a compact m-dimensional submanifold of class C^k, $k \geq 1$, of the euclidean space $\mathbb{R}^n$, and $\varepsilon > 0$ is arbitrary, there exists a PL m-dimensional manifold $\widetilde{M}$ and a homeomorphism $h\colon M \to \widetilde{M}$ such that $d(p, h(p)) < \varepsilon$ for all $p \in M$.*

Although not important from our point of view, it should be noted that from the above theorem it follows that the submanifold M is triangulable (see [Whi57]).

A proof of the theorem above, with a good geometrical flavor, has been given by H. Whitney [Whi57] and A. Bing [Bin57]. Whitney's proof holds for n-dimensional differentiable manifolds, while Bing's proof holds for topological surfaces in euclidean 3-dimensional space.

Sketch of the Proof: A constructive approach to prove the theorem is based on a very simple idea: construct a sufficient fine triangulation $\mathcal{T}$ of the euclidean space $\mathbb{R}^n$ such that for every simplex $\sigma \in \mathcal{T}$, the intersection $M \cap \sigma$ is homeomorphic to an m-dimensional disk D_σ. Substitute $M \cap \sigma$ by the disk D_σ and glue the disks adequately to get the combinatorial submanifold $\tilde{M}$.

We will discuss the sketch above for the codimension-one case, with special emphasis on the case of a two dimensional manifold in 3-space. The reader should consult [Whi57] for the details of proof for any dimension.

For each point $p \in M$ consider an open ball $B_p(\delta)$ with center p and radius δ such that for each point $q \in B_p(\delta) \cap M$, M is a graph over the tangent space T_qM. The family $\mathcal{B} = B_p(\delta)$, $p \in M$ is an open covering of M. Let ε be the Lebesgue number of $\mathcal{B}$, and consider a tubular neighborhood of M of radius $\leq \varepsilon$.

We say that a triangulation $\mathcal{T}$ of the euclidean space $\mathbb{R}^n$ is *good* if the following conditions are satisfied:

1. the diameter of each simplex $\sigma \in \mathcal{T}$ is $\varepsilon/2$;
2. $\mathcal{T}$ has no vertices on the surface M;
3. M intersects transversally the 1-dimensional skeleton $\mathcal{T}$;
4. each edge of $\mathcal{T}$ intersects M at most at one point.

The above conditions are not difficult to obtain:

1′. $\mathcal{T}$ can be a CFK triangulation of the space;
2′. This is obtained with a small perturbation of the triangulation;
3′. Also this condition is attained with a small perturbation by the transversality theorem;
4′. This condition follows immediately from the size of the diameter of each simplex of the triangulation.

With the above conditions the intersection of M with a simplex σ of $\mathcal{T}$ is homeomorphic to a disk. In fact, M intersects the edges of a

simplex $\sigma \in \mathcal{T}$ in either 3 or 4 points (see Figure 10.2). In the first case the intersection is approximated by a 2-simplex (triangle); in the second case the intersection can be approximated by the union of two 2-simplices.

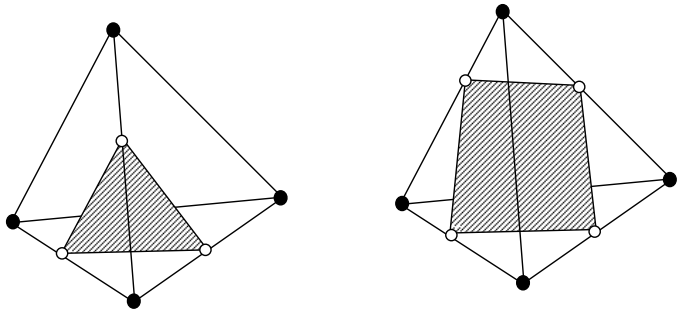


Fig. 10.2. Intersection of the surface with a $3D$-simplex.

10.2.2 Polygonization Algorithm

The appeal of the proof of the polygonization theorem is in its constructive nature. Is it really possible to devise an algorithm to compute the approximating PL manifold based on the proof? We can mention the following difficulties:

- to find a priori estimates of the tubular neighborhood. This is important to guarantee step 1.
- to devise an algorithm to obtain a triangulation of the space satisfying conditions (1), (2), (3) and (4). We call this a *good triangulation* of the euclidean space;
- to compute the intersection of the manifold M with the one-dimensional skeleton of the triangulation;

A priori estimates of the tubular neighborhood is a very difficult issue and it is a fundamental step in order to obtain good triangulations of the ambient space. An algorithm to compute the intersection of the manifold M with the 1-dimensional skeleton of the triangulation depends on the way M is defined.

We will describe the algorithm with details for implicitly defined manifolds of codimension 1, that is, $M = F^{-1}(0)$, $F\colon \mathbb{R}^m \to \mathbb{R}$ is a

function of class C^1 and 0 is a regular value of F. Since we are not able to estimate the tubular neighborhood we construct a "sufficiently fine" triangulation in the hope that the condition (1) is satisfied. Condition (2) is easy to check explicitly; condition (3) can be, possibly, attained with a small perturbation of the triangulation, and condition (4) is attained using a "very fine" cell decomposition of the space.

To generate the combinatorial manifold $\widetilde{M}$ approximating an implicit surface M we proceed as follows: First we determine all of the triangles that intersect M. These triangles constitute a spatial enumeration of M.

Given that the spatial enumeration C associated with M has already been computed, $\widetilde{M}$ is generated by the following method:

Scan and classify the simplices c_i of C.
For all simplices c_j intersecting M
  Compute the intersection of M and c_j.
  Create and add the corresponding polygons to $\widetilde{M}$.

The reader should observe each of the above steps in Figure 10.3. In what follows we will describe the three main steps of the algorithm.

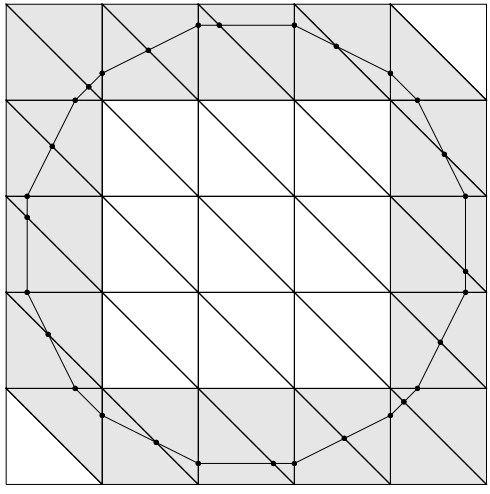


Fig. 10.3. Illustration of the polygonization algorithm

Simplex Classification. The simplex classification relies on the point membership property of the implicit function F. It is possible to

determine whether a simplex is inside, outside or intersects the boundary of the object according to the values of F at the vertices of the simplex. Assuming that the subdivision C is sufficiently fine the classification is as follows: If F is negative at the vertices of c then the simplex is in the interior of the object. If F is positive at the vertices of c then the simplex is in the exterior of the object. When the sign of F is not the same at all vertices of the simplex then c must intersect the boundary of the object. An intersecting simplex is also called a *transverse simplex*. These cases are illustrated in Figure 10.4.

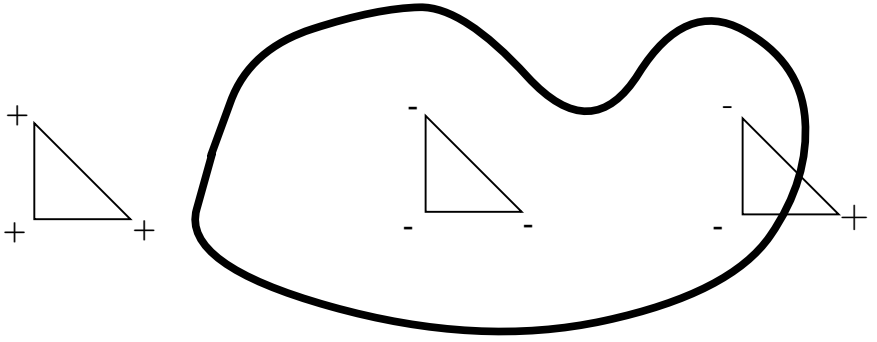


Fig. 10.4. Cell classification

Notice that in the classification scheme above, we rely on the fact that we have a good triangulation, so that situations such as indicated in Figure 10.5 do not occur.

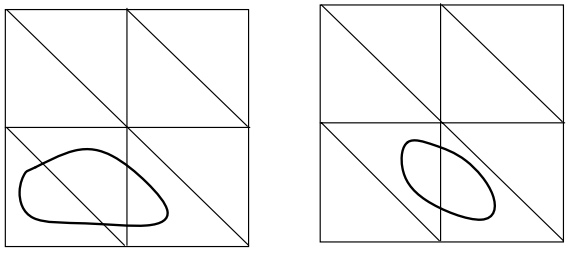


Fig. 10.5. Examples of inadequate space decompositions.

Intersection Computation. The problem here is: for each one-dimensional face σ_1 of the simplex, verify whether M intersects σ_1 and

if it is the case, compute $M \cap \sigma_1$. From condition (4), the intersection with each edge consists of just one point. To check for the intersection we just have to verify the sign of F on each of the two vertices v_0 and v_1 of σ_1. If $F(v_0)$ and $F(v_1$ have opposite signs, $M = F^{-1}(0)$ intersects the edge σ_1. The computation of the intersection point (only one from condition (4)) reduces to a root finding problem (see Section 9.3): If $\gamma(t) = tv_1 + (1-t)v_0$ is the parametric equation of the edge support line, we must find the roots of the equation $F(\gamma(t)) = 0$ on the interval $(0, 1)$.

Polygon Generation. Polygons are created in two steps: First, the polygon vertices have already been computed by intersecting the 1-dimensional faces of each simplex c with the implicit surface M. Then, the polygons are formed by ordering vertices in a consistent way induced by the orientation of the original space triangulation.

Note that this method takes advantage of the space subdivision enumeration associated with the implicit object to perform both sampling and structuring.

10.3 Implicit Solids

The above method computes a polygonal approximation for a manifold $M = F^{-1}(0)$. It can be easily extended to compute approximations to implicit solids such as $F(x) \leq a$ or $a \leq F(x) \leq b$. In this case we proceed as follows:

- compute the spatial enumeration based on the space decomposition, considering now the inequalities that define the solid;
- For each simplex on the boundary (that is, not contained entirely in the interior of the solid), compute the polygonal approximation as before. These polygons constitute faces of new simplices on the boundary of the solid.

10.4 Approximation Theory

We can view the polygonization method described in the previous section from the point of view of approximation theory. This will give a different and useful insight into the problem.

As before, let $M = F^{-1}(0)$ be an implicit manifold, such that 0 is a regular value. Consider a function $\widetilde{F}\colon \mathbb{R}^3 \to \mathbb{R}$ that approximates F in some metric and 0 is also a regular value for $\widetilde{F}$. Then $\widetilde{M} = \widetilde{F}^{-1}(0)$ is a manifold that approximates M. Therefore, instead of using the function F to compute the manifold $\widetilde{M}$ that approximates M, we use the approximating function $\widetilde{F}$ to F.

In the polygonal approximation above, the function $\widetilde{F}$ is an affine approximation to F, defined in each simplex $\sigma = (p_0, p_1, \ldots, p_n)$ by a linear interpolation in barycentric coordinates

$$\widetilde{F}(\sum_{i=0}^{n} \lambda_i p_i) = \sum_{i=0}^{n} \lambda_i F(p_i).$$

The condition of being a good triangulation guarantees that 0 is a regular value of $\widetilde{F}$, that is, in each simplex σ, the hyperplane $\widetilde{F}^{-1}(0)$ contains no faces of σ, therefore its intersections with σ is an affine cell. The vertices of the affine cell are obtained by solving the system of linear equations

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ f(p_0) & f(p_1) & \cdots & f(p_n) \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_n \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

It should be noted that the polygonal approximation obtained here is not the same we obtained before, because the roots of $\widetilde{F}$ on a simplex σ are not the same as those of F. This is illustrated in Figure 10.6

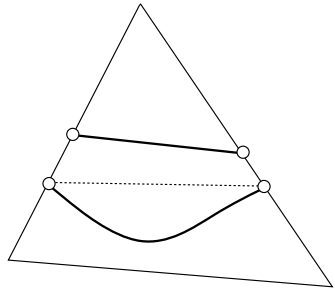


Fig. 10.6. The polygon obtained from $F^{-1} \cap \sigma$, indicated with a dashed line, is not the same as the polygon obtained from $\widetilde{F}^{-1} \cap \sigma$.

Higher order approximations $\widetilde{F}$ of F can be used in order to get better approximations of the implicit manifold M. Some work in this direction can be found in [BI92].

10.5 Classification of Polygonization Methods

Polygonization methods are classified according to three main criteria:

1. **Type of decomposition** – It can be either *extrinsic*, when the space domain of the implicit function is subdivided, or *intrinsic*, when the implicit object itself is subdivided directly.
2. **Sampling strategy** – It is related to the root finding method used.
3. **Structuring method** – It can be carried either before, after or in parallel with the sampling process.

10.5.1 Intrinsic Decomposition

Intrinsic decomposition methods usually combine sampling and structuring in one single step, and adopt a continuation strategy. They start with a known sample point p on the implicit surface and construct new triangles by finding more neigbour points. These points can be generated on the tangent plane at p and then project onto the implicit surface. Note that this is always possible for a smooth manifold by the implicit function theorem. This strategy is used in the following works: [Hen93], [Har98] [KS01] and [AG01].

Other intrinsic decomposition methods perform sampling before structuring. Some examples of these methods are [Rhe87] and [dFdMGTV92].

Some hybrid decompositions schemes are also present in the literature. As an example we can mention [Chu90].

10.5.2 Extrinsic Decomposition

Extrinsic decomposition methods are the most popular. Allgower's seminal work, [AS85], is a classical example in this category. The polygonization algorithm described in section 10.2.2 also belongs to this category.

10.6 Extrinsic Polygonization Methods

According to the three criteria discussed above, the polygonization methods using extrinsic space decomposition are further subdivided according to:

(i) the class of cellular complex employed in the space decomposition,
(ii) the tracking strategy adopted to scan intersecting cells, and
(iii) the type of subdivision used.

In relation to the domain decomposition, polygonization methods can be: simplicial or non-simplicial. *Simplicial methods* triangulate the domain of F forming a simplicial complex. *Non-simplicial methods* tessellate the domain of F building a general cellular complex.

In relation to the tracking strategy, polygonization methods can use: continuation or full scan. *Continuation methods* start with a seed cell that is known to intersect the surface and search its immediate neighbor cells to determine which ones also intersect. This process is repeated until all the transverse cells are found. *Full scan methods* visit and classify all elements of the cellular complex to find which cells intersect the surface.

In relation to the type of space subdivision, polygonization methods can be: uniform or adaptive. *Uniform methods* subdivide space at regular intervals generating a decomposition of fixed resolution. *Adaptive methods* partition space unevenly such that the resulting decomposition is finer where more detail is needed.

We will discuss briefly each of the methods above.

10.6.1 Non-simplicial Methods

The most popular non-simplicial polygonization method is the marching cubes algorithm [LC87]. It employs a rectangular uniform cubic tessellation of space. The method consists essentially of three steps: the cellular complex is scanned to identify transverse cubes; the topology of polygons in each cube is determined through a table look-up procedure; and the vertex locations are computed by interpolation.

A similar method was developed independently by [WMW86b] and [PP88]. The difference between them is in the way polygon topology is derived.

Non-simplicial methods are fast and very simple to implement. The main drawback is that they are not robust. This is because the intersection of $\widetilde{F}^{-1}(0)$ and a rectangular cell cannot be uniquely determined.

As a consequence inconsistent decisions may produce holes in the polygonization. Figure 10.7 shows a two dimensional example where an ambiguous situation arises.

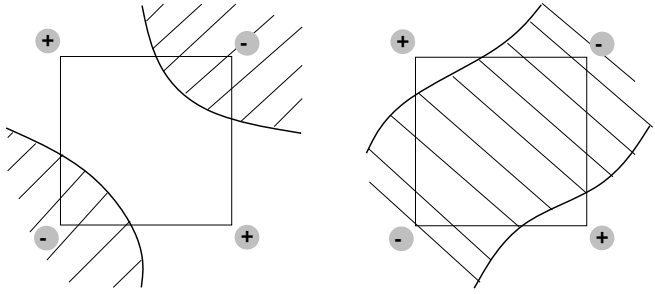


Fig. 10.7. Ambiguous Polygonization

This problem can fixed using different approaches. An ad-hoc disambiguating approach, proposed by [Bei90], consists of checking the center points of the cube faces. This only alleviates the problem using extra samples. The correct solution consists in choosing a globally consistent way to tesselate each intersecting cell [MSS94, Nat94, CGMS00].

A polygonization method that follows the same algorithmic structure of the standard Marching Cubes method and performs feature sensitive sampling is described in [KBSS01]. It employs distance functions relative to the main coordinate directions in order to compute exact intersections from a sampled volumetric representation.

10.6.2 Simplicial Methods

The polygonization algorithm described in Section 10.2.2 is in this category.

Simplicial polygonization methods are theoretically more sound because they rely on the piecewise linear structure induced by the triangulation of the domain of F. The theoretical formulation of this theory has been studied for years in the area of algebraic topology. This fact is reflected in the theoretical discussions preceding the description of the polygonization algorithm, including the triangulation theorem for differentiable manifolds.

The work in simplicial polygonization methods was pioneered by Allgower [AS85, AG87]. His method is based on the Coxeter-Freudenthal triangulation and it uses a pivoting scheme to move from one simplex to another [AG91] The simplicial approximation of the implicit surface in each cell is obtained computing the kernel of $\widetilde{F}$ over the simplex. This requires the solution of a linear system of equations.

A variation of the previous method was suggested by [CFT88]. The main goal was to improve performance.

10.6.3 Continuation Methods

Continuation methods exploit the coherence of the PL manifold $\widetilde{M}$ to reduce the search space of transverse cells [AG90], [ZJ91].

Given a compact 2-manifold M and a cell c that intersects M it is possible to generate a closed combinatorial manifold $\widetilde{M}$ that approximates M visiting only the transverse cells in C. The method operates by processing in a systematic way the neighbors of c. It maintains a list W of working cells. Neighbors of cells in W are inserted in the list, and a cell c is kept in W while there are neighbors of c to be inserted. When the list W is empty all the transverse cells have been processed.

A characteristic of continuation methods is the requirement of an initial value. In the case of polygonization we need a point in each connected component of the object. This is usually not a problem because it cames naturally in most applications.

10.6.4 Adaptive Methods

Since we do not have, in general, a priori, estimates of the tubular neighborhood of the implicit manifold, we must start with a sufficiently fine decomposition of the space. This increases the number of cells and consequently the number of polygons in the final model. This can be avoided in part by refining the decomposition cells only where necessary. This is the basis of adaptive methods.

Adaptive methods create spatial decompositions that are sensitive to some characteristic of the object. They start with an initial subdivision of domain of the object, and refine it recursively until the adaptation criteria is met. This is very much application dependent. In the case of polygonization of implicit surfaces it can measure the fidelity of the PL approximation. For finite element analysis it is often related to material properties of the object.

There are two basic approaches to adaptivity: Their main difference is in the way they constraint the subdivision. One constraints the cells of the decomposition, while the other constraints the edges of the polygonization.

The first approach subdivide cells independently constraining polygon edges based on the refinement criteria. This guarantees that the resulting polygonization will not have cracks between different levels of subdivision. A simplicial implementation of this method is given in [Vel90a]. A similar method is [HJ99]. A variant of this approach that refines the polygonization as a post-process was suggested by [Bei90]. An implementation of this algorithm is described in [Vel96]. A weakness of such a solution is that it may fail to detect some features of the surface, since it operates on the polygonization alone.

The second approach generates a restricted tree. In order to maintain the subdivision structure balanced, this method performs repeated refinement steps modifying at each stage only one cell. Whenever a cell is divided its neighbors are constrained to be at levels immediate above or bellow. An implementation of this algorithm based on octrees was introduced by [Blo88]. [HW90] proposed a version of this algorithm for simplicial complexes. A general adaptive tesselation method for implicit and parametric surfaces that uses a restricted binary tree by construction is [VdFG99].

As we have seen in Section 10.2.2, extrinsic polygonization methods need to start with decomposition of the euclidean space that captures the topology of the implicit shape. One way to do that is through a Morse function defined on the domain of F. An adaptive polygonization for dynamic surfaces based on this principle is [SH97].

11. Primitive Implicit Objects

This chapter discusses primitive implicit objects. We present formulations to define them and analize their main characteristics.

As we have seen, implicit objects are described by functions of the embedding n-dimensional space. Primitive implicit objects are defined by a single function along with its parameters. According to how these functions are specified, primitives are grouped into tree basic classes: analytical, procedural, and sample-based.

Although most formulations generalize to n dimensions, we will consider only surfaces and volumes in three-space.

11.1 Analytical

Analytical primitives are implicit objects defined by analytic functions. This includes algebraic functions, such as the quadrics, as well as non-algebraic functions, such as superquadrics.

An algebraic function $F(x, y, z)$ is a polynomial involving the variables x, y, z.

The general function is given by the equation

$$F(x,y,z) = \sum_{i=0}^{l} \sum_{j=0}^{m} \sum_{k=0}^{n} a_{ijk} x^i y^j z^k \tag{11.1}$$

The degree of this polynomial is the maximum combined degree of the non-zero terms $d = l + m + n$. The number of terms of a function of degree d is $(n + 1)(n + 2)(n + 3)/6$.

Quadrics are given by algebraic functions of degree 2. Superquadrics generalize quadrics providing control parameters to manipulate the shape of these primitives [Bar81]. Superquadrics are not algebraic, because they are defined by polynomial equations involving fractional powers.

11.1.1 Plane

A plane is the simplest algebraic primitive. It is given by an affine polynomial

$$ax + by + cz + d \tag{11.2}$$

where (a, b, c) is a vector orthogonal to the plane and the value of d is related to the distance of the plane to the origin.

This equation simplifies if the plane is perpendicular to one of the coordinate axis, a common situation in space subdivision procedures.

A half-space is defined by $F(x, y, z) \leq 0$. Polyhedra can be constructed from a combination of half-spaces. This modeling scheme employs CSG operations as discused in Chapter 8.

11.1.2 Quadrics

A quadric is defined by a polynomial of degree 2

$$\begin{aligned} ax^2 + bxy + cxz + dx \\ + ey^2 + fyz + gy \\ + hz^2 + iz \\ + j \end{aligned} \tag{11.3}$$

If a homogeneous representation is used, this can be rewritten in matrix notation as $p^t Q p$

$$[x, y, z, w] \begin{bmatrix} a & b & c & d \\ b & e & f & g \\ c & f & h & i \\ d & g & i & j \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \tag{11.4}$$

The different types of quadric surfaces can be distinguished by examining the eigenvalues of the matrix Q.

A projective transformation can be directly incorporated into the representation of a quadric. Given a transformation defined by a 4×4 homogeneous matrix T, the transformed quadric is

$$\overline{Q} = T^{\star} Q T^{\star t} \tag{11.5}$$

where $\star$ indicates the adjoint of a matrix.

A subset of the general quadrics, which contains all non-degenerate quadrics, are the quadrics of revolution. Below we give the equations for the main quadrics of revolution in their canonical coordinate systems.

Sphere.

$$x^2 + y^2 + z^2 - 1 \tag{11.6}$$

Cylinder.

$$x^2 + y^2 - 1 \tag{11.7}$$

Cone.

$$x^2 + y^2 - z^2 \tag{11.8}$$

Paraboloid.

$$x^2 + y^2 + z \tag{11.9}$$

Hyperboloid of One Sheet.

$$x^2 + y^2 - z^2 - 1 \tag{11.10}$$

Hyperboloid of Two Sheets.

$$x^2 + y^2 - z^2 + 1 \tag{11.11}$$

Figure 11.1 shows a drawing of the main quadric surfaces.

Because of their simplicity, many computational techniques, such as ray-surface intersection and silhouette extraction [Bli84], can be specialized in a efficient way to quadric surfaces.

11.1.3 Torus

indexTorus

A torus is the cartesian product of two circles of radius a and b. It is defined by a polynomial of degree 4:

$$(x^2 + y^2 + z^2 - (a^2 + b^2))^2 - 4a^2(b^2 - z^2) \tag{11.12}$$

The torus is an example of surface that has both a parametric and implicit description. This property can be exploited in many problems. The particular structure of its algebraic description make it possible to compute ray-surface intesections in closed form.

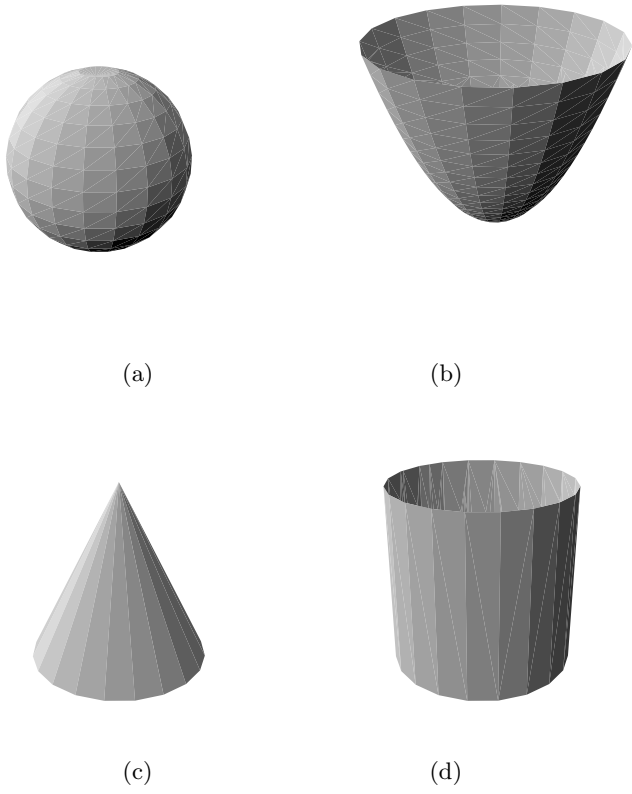


Fig. 11.1. Main quadric surfaces: Sphere (a); Paraboloid (b); Cone (c); Cylinder(d)

11.1.4 Superquadrics

Below we give the equations defining the main types of superquadrics in their canonical coordinate systems.

An interpretation of superquadrics as blend surfaces will be given later in this section.

Superquadrics constitute another class of surfaces which possess a natural parametric and implicit description. An example of application of this property to model deformable surfaces can be seen in [SP91], where implicit superquadrics have been used as the base surface for a displacement field defined on the parametric domain of the superquadric.

The superquadrics are: the superellipsoid, the superhyperboloid of one and two sheets, and the supertoroid.

Superellipsoid.

$$\left(\left(\frac{x}{a_1}\right)^{\frac{2}{e_2}} + \left(\frac{y}{a_2}\right)^{\frac{2}{e_2}}\right)^{\frac{e_2}{e_1}} + \left(\frac{z}{a_3}\right)^{\frac{2}{e_1}} - 1 \tag{11.13}$$

Superhyperboloid of One Sheet.

$$\left(\left(\frac{x}{a_1}\right)^{\frac{2}{e_2}} + \left(\frac{y}{a_2}\right)^{\frac{2}{e_2}}\right)^{\frac{e_2}{e_1}} - \left(\frac{z}{a_3}\right)^{\frac{2}{e_1}} - 1 \tag{11.14}$$

Superhyperboloid of Two Sheets.

$$\left(\left(\frac{x}{a_1}\right)^{\frac{2}{e_2}} - \left(\frac{y}{a_2}\right)^{\frac{2}{e_2}}\right)^{\frac{e_2}{e_1}} - \left(\frac{z}{a_3}\right)^{\frac{2}{e_1}} - 1 \tag{11.15}$$

Supertoroid.

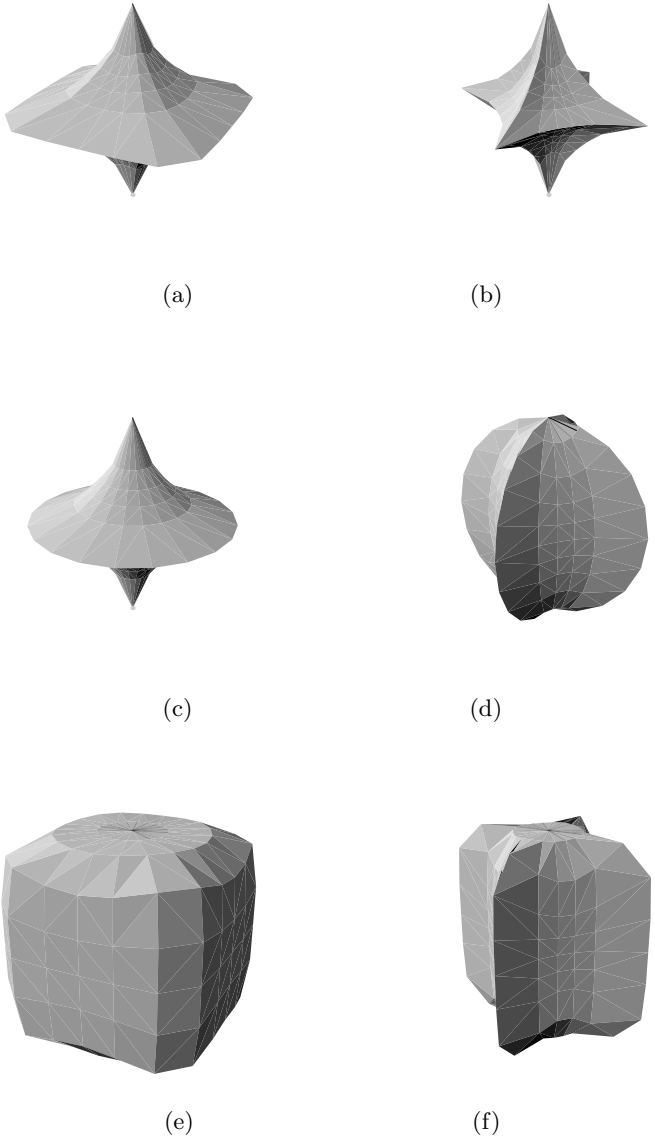
$$\left(\left(\left(\frac{x}{a_1}\right)^{\frac{2}{e_2}} + \left(\frac{y}{a_2}\right)^{\frac{2}{e_2}}\right)^{\frac{e_2}{e_1}} - a_4\right)^{\frac{2}{e_1}} + \left(\frac{z}{a_3}\right)^{\frac{2}{e_1}} - 1 \tag{11.16}$$

where $a_4 = \frac{r}{\sqrt{a_1^2+a_2^2}}$ and r is the torus radius.

Superquadrics can be formulated parametrically as spherical products of superconic curves. Superconics are similar to normal conics but raised to an arbitrary power. Superquadric surfaces are to quadrics as superconic curves are to conics. The exponent e control the roundness of the curves. If $e \approx 1$ the curve is a conic. If $e < 1$ the curve tends to a square. If $e < 2$ the curve becomes pinched and is concave. The parameters e_i are used to pinch, round, or square off portions of the surface, and to produce edges, fillets of arbitrary smoothness.

Figure 11.2 shows examples of the superellipsoid varying the parameters e_1 and e_2.

For a discussion of applications in computer graphics based on superquadrics see [Bar81].



(a) (b)

(c) (d)

(e) (f)

Fig. 11.2. Superquadric ellipsoids

Two Interpretations of Superquadrics

We now give two different interpretations to superquadrics: one as a blend primitive and the other as a variable metric distance function.

Let a_1 and a_2 be the characteristic functions of the half-spaces whose boundaries are the coordinate axis

$$A_1(x, y) = x$$
$$A_2(x, y) = y$$

And let s be the characteristic function of the unit circle

$$S(x, y) = x^2 + y^2 - 1$$

If we use s as a blending function on a_1 and a_2, the result is

$$\begin{aligned} S(A_1(x, y), A_2(x, y)) &= (A_1(x, y))^2 + (A_2(x, y))^2 - 1 \\ &= x^2 + y^2 - 1 \end{aligned}$$

which is a blend of the two coordinate axis A_1 and A_2. A generalization of the equation above will lead us to the super-ellipse. Note that the super-elliptic blend is inspired precisely in this equation. (See Chapter 8.)

Now, consider a distance function D according to the different metrics:

$\|\mathbf{x}\|_1 = |x_1| + \ldots + |x_n|$;
$\|\mathbf{x}\|_2 = (x_1^2 + \ldots + x_n^2)^{\frac{1}{2}}$;
$\|\mathbf{x}\|_\infty = \sup(x_1, \ldots, x_n)$.

D corresponds respectively to the L_1, the Euclidean, and the Manhattan distances.

From the super-ellipse equation

$$f(x, y) = \left(\frac{x}{a_1}^{\frac{2}{e_2}} + \frac{x}{a_1}^{\frac{2}{e_2}} \right)^{\frac{e_2}{e_1}} - 1$$

we can get exactly these metrics. To verify that, normalize the axis of the super-ellipse, $a_1 = a_2 = 1$, make the exponent $e_1 = 1$, and set $e_2 = 2$, $e_2 = 1$ and $e_2 \ll 1$ to obtain the one-norm, two-norm, and the infinity-norm respectively.

Note that when $e_2 \to \infty$ the shape converges to the union of the two coordinate axis reinforcing the interpretation of superquadrics as a blend.

11.2 Procedural

Procedural primitives are implicit objects whose characteristic function F is defined *algorithmically*. This seems too general since any function can be generated procedurally. For this reason an implicit object is classified as a procedural primitive only when its description is inherently algorithmic. For a discussion of procedural modeling see [GV98].

11.2.1 Fractals

Fractals objects have a natural algorithmic description. They can be specified either through a recursive or an iterative procedure.

The recursive method stars with an approximation of the object and refine it by repeatedly altering its parts. The iterative method determines whether or not a point is part of the object based on its assymptotic behavior. The condition is that the orbit of the point converges to the basin of attraction of the fractal.

This last method is the most suited to describe fractals implicitly. A good example of this type of fractal is the Julia set [Nor82].

The basic procedure for fractal evaluation is as follows

$$F(x,y,z) = \begin{cases} 1 & \text{if } I_n(x,y,z) \to \infty \text{ as } n \to \infty \\ 0 & \text{otherwise} \end{cases} \tag{11.17}$$

where $I : \mathbb{C} \to \mathbb{C}$ and the subindex means iteration.

Hart and co-authors have investigated techniques for computation and visualization of fractal implicit objects [HH95] [Har92] [BH97] [CHF98] [Har89] [HD91] [CLH01].

11.2.2 Hypertexture

Hypertexture is a procedural modeling scheme based on functional composition [PH89]. Objects are described by implicit primitives whose characteristic function is controlled by a successive application of shaping functions. The model is conceived as an assemble of density distributions in space.

The characteristic function F gives a base shape which has a hard solid region and a soft region that is manipulated by the shaping functions. F is also called *object density function* and is given by:

$$F : \mathbb{R}^3 \to [0,1] \tag{11.18}$$

where the soft region is the set of points that satisfy $0 < F(\mathbf{x}) < 1$.

The shaping functions g_i, also called *density modulation functions*, modulate density within the soft region, controlling some spatial characteristic of the object.

A hypertexture primitive H is defined by

$$H(F(\mathbf{x}), \mathbf{x}) = g_n(\ldots g_2(g_1(F(\mathbf{x})))) \tag{11.19}$$

One essential component of the toolkit of shaping functions is the *noise* function [Per85a]. It is an effective means to introduce controlled randomness into the model. The base level toolkit include the functions *bias* and *gain*, as well as arithmetic and control flow operations.

Boolean operations can also be applied to hypertexture primitives if they are properly redefined:

$\overline{F} \equiv 1 - F(\mathbf{x})$.
$F_1 \cap F_2 \equiv F_1(\mathbf{x})F_2(\mathbf{x})$.
$F_1 \setminus F_2 \equiv F_1(\mathbf{x}) - F_1(\mathbf{x})F_2(\mathbf{x})$.
$F_1 \cup F_2 = \overline{\overline{F_1} \cap \overline{F_2}} \equiv F_1(\mathbf{x}) + F_2(\mathbf{x}) - F_1(\mathbf{x})F_2(\mathbf{x})$.

In that sense, hypertexture objects constitute a complete scheme for modeling with implicit shapes. In this scheme, the primitive is an object density function, shaping functions are unary manipulation operations, and the redefined CSG operations are used for combination of primitives.

Hypertexture was intended to address the *shape plus texture* problem, but can also be used to construct fractal shapes. This framework has been used to generate realistic models of fuzzy geometry such as in hair, fur, fire, glass, and erosion effects. (See Figure 11.3.)

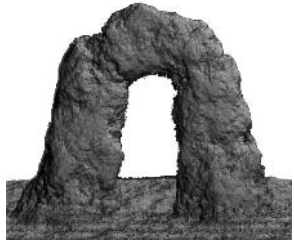


Fig. 11.3. Hypertexture Object

11.3 Sample-Based Implicit Primitives

Sample-based primitives are implicit objects defined by a collection of values $\{F_i = F(x_i)\}$ of the function F, at a discrete set of sampling locations x_i, $i = 1, \ldots, N$.

These samples form a spatial data set which directly specifies the function F. In order to generate a continuous function from these discrete samples an interpolation scheme must be used.

Sample-based implicit primitives can be classified based on two criteria: (i) the sampling pattern and; (ii) the interpolation scheme employed to reconstruct $F(x)$. The sampling pattern may be regular or irregular. The reconstruction depends on the type of sampling pattern and on the order of the interpolationg scheme.

This type of primitive assumes an underlying structure. The purpose of such an structure is twofold — it organizes the data for storage and also define the topological relations for function reconstruction.

11.3.1 Irregular Samples

Irregular samples are generated from an unorganized collection of data points. This type of primitive requires some kind of structuring to be performed before reconstructing the function F. The reconstruction problem is known as scattered data set interpolation [Alf89].

Piecewise Linear Voronoi Models.

In this type of model sample points are structured by calculating the Voronoi diagram of their spatial locations and then taking the associated Delaunay triangulation. The triangulation provides a simplicial complex where barycentric interpolation can be performed.

For each tetrahedron T with vertices $\{v_1, v_2, v_3, v_4\}$, $v_i \in \mathbb{R}^3$ and corresponding sample values $\{f_1, f_2, f_3, f_4\}$. Linear barycentric interpolation within T is done in the following way:

$$F(x)_{|T} = \sum_{i=1}^{4} \alpha_i f_i \tag{11.20}$$

where $\alpha_1, \alpha_2, \alpha_3$, and α_4 are barycentric coordinates, i.e. $\sum_{i=1}^{4} \alpha_i = 1$.

Since adjacent tetrahedron of this simplicial complex share sample values, this gives a piecewise linear reconstruction of F.

Note that in this model, sample points can be anywhere in space and are not restricted to be on the implicit surface $F^{-1}(c)$.

Simplicial Hull Models.

In this type of model we start with a polyhedron that defines the shape of the implicit surface. Then, we construct a simplicial hull that conforms to a triangulation of this polyhedron. Next, a Bézier-Bernstein polynomial is computed within each tetrahedron of the simplicial complex, such that the zero set of this polynomial is an implicit piecewise polynomial description of the surface. Each of these pieces of the surface can be considered as an implicit surface patch. Continuity between adjacent patches is enforced by requiring that the vertex/edge/face adjacent polynomials be continuous with each other.

A polynomial of degree m over a tetrahedron T is written in Bézier-Bernstein form as

$$F(x) = \sum_{|\lambda|=m} b_{\lambda_1\lambda_2\lambda_3\lambda_4} B_\lambda^m(\alpha) \tag{11.21}$$

where

$$B_\lambda^m(\alpha) = \frac{m!}{\lambda_1!\lambda_2!\lambda_3!\lambda_4!}\alpha_1^{\lambda_1}\alpha_2^{\lambda_2}\alpha_3^{\lambda_3}\alpha_4^{\lambda_4} \tag{11.22}$$

$\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ are barycentric coordinates of the point x relative to T, $|\lambda| = \sum_{i=1}^{4} \lambda_i$, and B_λ^m is the Bernstein polynomial.

The coefficients $b_{\lambda_1\lambda_2\lambda_3\lambda_4}$ are control values associated with vertices of the tetrahedron.

In the interpolating simplicial hull model the implicit surface passes through the vertices of the original tetrahedron. Therefore, the control values at these vertices are set to zero and the control values at the remaining vertices of the hull are computed to guarantee continuity. Note that in order to achieve higher-order continuity the simplicial hull must be subdivided.

Smooth interpolation models that produce C^1 and even C^2 continuous implicit surface patches based on this framework have been proposed by many researchers [Dah89] [Guo91b] [Guo91a] [MW91] [XHB01] [BCX94] [BCX95b] [BCX95a].

Variational Models.

Variational models are defined as an scattered-data interpolation using a variational approach. Under this framework, an implicit function is constructed from samples as the minimizer of some energy functional subject to interpolative constraints [PASS95][TO99].

Given a set of sample locations x_i and sample values f_i, we want to find a function F, that minimizes the energy $E_F(x)$, while interpolating the sample values, i.e. $F(x_i) = f_i$. The energy functional E

can measure either the first, second or third order energy of F or a combination of those [CS96].

Sample points are divided in three categories: surface points s_i with $f(s_i) = 0$; interior points p_i with $f(p_i) > 0$ and exterior points e_i with $f(e_i) < 0$). The surface normal can be used to determine the location of p_i and e_i.

The second order energy functional is the most popular and corresponds to the thin-plate spline. In three-dimensions the thin-plate solution is equivalent to interpolating the sample points using radial basis functions $\phi(r) = |r|^3$. Other energy functionals also have a known closed form solution, with different radial basis functions.

The resulting interpolating function is given by

$$F(x) = \sum_{i=1}^{n} w_i \phi(\|x - x_i\|) + P(x) \tag{11.23}$$

where w_i is the weight of the radial basis function centered at x_i, and P is a first degree polynomial that accounts for the linear and constant portions of F, ensuring positive definiteness of the numerical solution.

To solve for the set of weights w_i that satisfy the interpolation constraints $F(x_i) = f_i$, we substitute each f_i into equation 11.23

$$\sum_{j=1}^{n} w_j \phi(\|x_i - x_j\|) = f_i \tag{11.24}$$

where we assumed that $P(x) = 0$, for simplicity.

This results in the following linear system:

$$\begin{pmatrix} \phi_{11} & \cdots & \phi_{1n} \\ \vdots & \ddots & \vdots \\ \phi_{n1} & \cdots & \phi_{nn} \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \tag{11.25}$$

where $\phi_{ij} = \phi(\|x_i - x_j\|)$.

The above matrix is real-symmetric and can be made to be positive-definite for a proper choice of basis function ϕ and polynomial P. Various numerical methods can be used to solve the matrix equation (11.25), such as LU decomposition of the conjugate gradient method. Once the weights w_i have been computed, equation 11.23 gives the interpolating implicit function.

The main difficulty with variational model approach is the computational complexity for building, solving, and evaluating the function F. The complexity of building and solving the system is $O(n^2)$ and the complexity to evaluate F is $O(n)$, where n is the number of sample points. When the number of points is large, these costs can be prohibitive.

One way to overcome this difficulty is to employ a locally-supported radial basis function [Ter01]. The general solution is

$$\phi(r) = \begin{cases} (1-r)^p P(r) & r < 1 \\ 0 & \text{otherwise} \end{cases} \tag{11.26}$$

This reduces the complexity to $O(n \log n)$ to build the system, and to $O(\log n)$ to evaluate F.

Another option is to resort to faster numerical methods for fitting and evaluating radial basis functions, such as the Fast Multipole Method [CBC+01].

11.3.2 Regular Samples

Regular samples are generated from an organized colletion of data points. These points are disposed according to a regular structure and, in general, are associated with the vertices of a three-dimensional mesh.

Voxel Arrays.

A voxel array is a structured collection of values based on a three-dimensional grid. Usually, this grid is a uniform rectangular lattice. Note that the location x_{ijk} of datapoints is not explicitly encoded since is intrinsic to the grid and can be derived from the indices i, j, k.

This type of representation is very common in medical applications, where the data sets are produced by sensing devices, such as Computer Assisted Tomography (CAT), Magnetic Resonance Imaging (MRI), and other techniques. In such applications there is a great emphasis placed on both surface and volume information [BW01b];

The funtion F is often reconstructed from the samples using trilinear interpolation.

The large amount of data samples demand compression techniques. For certain classes of data sets, such as sampled distance fields, it is possible to adopt a more economical adapted structure, based on octrees [FPRJ00].

Volume Meshes.

Volume meshes has the structure of a regular or semi-regular grid. This grid is non-uniform in general. It can be either a tetrahedral or hexahedral grid.

This type of representation is often employed in scientific visualizations, where the data sets are generated by numerical simulations, such as finite element analysis (FEM).

The regular structure is suitable to reconstruction methods using higher order polynomial functions.

In the case of hexahedral grids, we can use tensor product basis functions, such as the Bézier basis

$$F_C(x) = \sum_{i=0}^{m} \sum_{j=0}^{n} \sum_{k=0}^{p} w_{ijk} B_i^m(x_1) B_j^n(x_2) B_k^p(x_3) \tag{11.27}$$

where

$$B_i^m(x) = \binom{m}{i} x^i (1-x)^{m-i} \tag{11.28}$$

is the Bernstein polynomial, and F_c is retricted to a hexahedral cell with normalized parametric coordinates $x \in [0,1] \times [0,1] \times [0,1]$.

The reconstruction function F is defined in a piecewise manner from the collection of F_C and the sample values.

Note that a voxel array can be represented as a volume mesh if higher order interpolation is desired.

Volume meshes structures and smooth reconstruction schemes based on the Bézier basis have been investigated by [Las85].

12. Skeleton-Based Implicit Primitives

This chapter studies skeleton-based implicit primitives. Skeleton primitives are implicit objects defined in terms of a distance measure to some point-set, such as points, curves or surfaces. These sets are called *skeletons*. The boundary of the object is a level surface c units away from the skeleton in a appropriate metric. Usually c is a parameter of the primitive and its characteristic function is expressed as $F(x, y, z) - c$.

Note that a skeleton primitive is a special case of the medial axis description, where the radius of the maximal balls is constant (equal to c), and the metric is arbitrary.

Skeletons are also closely related to blending, because a set of primitives can be naturally combined using blend operations, as we will see in the end of this chapter.

In the next sections, we will describe primitive skeletons according to the dimensionality of the point set.

12.1 Point Skeletons

The simplest skeleton is formed from a set of isolated points $\{\mathbf{p}_i\}$. Most of the effort in constructing these primitives goes into creating a flexible distance function. The properties of the distance function will determine the shape of primitives and how they blend together.

12.1.1 Blobby Models

Blinn [Bli82] was the first to introduce a skeleton based implicit model. He was motivated by the need to display molecular structures more accurately. This model is inspired on electron density maps. The distance function is a Gaussian centered at each point p of the skeleton. The parameters a and b are respectively the standard deviation and the height of the function.

$$D_i(\mathbf{x}) = b_i \exp(-a_i r_i^2) \tag{12.1}$$

where $r_i = \|\mathbf{x} - \mathbf{p}_i\|$ is the euclidean distance from $\mathbf{x}$ to the skeleton point $\mathbf{p}_i$.

A more intuitive shape specification is given in terms of the radius ρ of an isolated point skeleton, and the *blobbyness* factor β that controls how the point blends with others.

The new equation is

$$D_i(\mathbf{x}) = c \exp(\frac{\beta_i}{\rho_i^2} r_i^2 - \beta_i) \tag{12.2}$$

Since c now is included in the contribution of each term, it can be set to a standard value such as 1. The effect of changing the level surface can be achieved through the blobbynes factor.

Note that the function D is symmetric resulting in a spherical shape. D can be generalized to allow for arbitrary quadric shape functions if r_i^2 is substituted in the equation by $\mathbf{x}Q_i\mathbf{x}$.

12.1.2 Metaballs

A variation of Blinn's model is the metaball model developed by [NHK+85]. The metaball model uses a piecewise quadratic instead of an exponential function.

$$D_i(x) = \begin{cases} a_i(1 - 3m_i^2) & 0 \le m_i \le 1/3 \\ \frac{3}{2} a_i(1 - m_i^2) & 1/3 \le m_i \le 1 \\ 0 & 1 \le m_i \end{cases} \tag{12.3}$$

where $m_i^2 = \mathbf{x}Q\mathbf{x}$.

One important characteristic of this model in terms of computational efficiency is that the contribution of each point drops to zero at certain distance from it. This makes possible to consider only a subset of the points in the calculation of the function value at a given location.

12.1.3 Soft Objects

A similar point skeleton primitive was proposed by [WMW86b]. In this model the extent of influence e of each point p is an explicit parameter. The distance function is based on an Hermite cubic whose value is 1 at p and 0 at e. The slope of the curve is zero at both points, which

guarantees smoothness. The square of the distance to p is substituted into this equation giving rise to the sixth degree polynomial

$$D_i(\mathbf{x}) = 1 + \frac{-4w^6 + 17w^4 - 22w^2}{9} \tag{12.4}$$

where $w = \|\mathbf{x} - \mathbf{p}_i\|/e_i$ and e_i is the extent of the influence of the skeleton element p_i.

12.1.4 Other Formulations

There are many other formulations for primitive point skeletons. In [Gas93] and [BS95] a local shaping function is used to control the hardness of the blend between different primitives.

The hardness parameter is given by h_i in the two equations below.

The model proposed in [Gas93] is

$$D_i(x) = \begin{cases} \frac{1}{4}(2 + h_i + 2h_i r_i) & r_i \leq 1/2 \\ (-2 + h_i + 8r_i - 2h_i r_i) & 1/2 \leq r_i \leq 1 \\ 0 & 1 \leq r_i \end{cases} \tag{12.5}$$

where $r_i = \|x - p_i\|$.

The model proposed in [WW89] employs an anisotropic shaping function based on non-euclidean metrics, such as the L_m metrics.

A similar, more flexible, scheme was proposed in [BS95]. In this model, a point source is defined by 3 parameters: a position $p_i = (x_i, y_i, z_i)$, a radius of influence d_i and a primary direction, specified by the normalized vector $q_i = (a_i, b_i, c_i)$. From these parameters the following variables are derived:

$$u = \frac{x - x_i}{d_i} \qquad v = \frac{y - y_i}{d_i} \qquad w = \frac{z - z_i}{d_i}$$

and

$$t^2 = u^2 + v^2 + w^2 \qquad s = ua_i + vb_i + wc_i$$

Note that $t^2 \in [0, 1]$ and $s \in [-1, 1]$. The anysotropic distance functions are defined using these two variables, where t^2 gives the distance of a point x to the point skeleton x_i and s characterizes the its position relative to the primary direction. More complicated functions can be constructed by introducing additional directions and radius.

Figure 12.1 shows examples of isotropic and anysotropic primitive point skeletons.

Fig. 12.1. Isotropic and anysotropic primitive point skeletons

12.2 Curve Skeletons

A curve skeleton is constructed from a curve segment in $\mathbb{R}^3$. This line defines the axis of a *generalized cylinder* with possibly variable radius and cross-section [BS91].

The basic equation for a curve skeleton implicit object is

$$\|C(\mathbf{x}, \mathbf{p}_i) - \mathbf{x}\| - c \tag{12.6}$$

where C is the closest point on the curve, and c is the contour value. Note that c may be a function of the curve parameter t, and the euclidean distance function may be substituted by an arbitrary distance function.

In this context, curve skeletons are closely related with sweep surface models that are generated by sweeping a shape along a prescribed trajectory [SW97].

12.2.1 Lines

For a line segment $\overline{\mathbf{p}_0\mathbf{p}_1}$ the distance from a given point x and the closest point on the line is

$$C = \begin{cases} \mathbf{p}_0 & \alpha \leq 0 \\ \mathbf{p}_0 + \alpha(\mathbf{p}_1 - \mathbf{p}_0) & 0 < \alpha < 1 \\ \mathbf{p}_1 & \alpha \geq 1 \end{cases} \tag{12.7}$$

where $\alpha = \mathbf{l} \cdot (\mathbf{x} - \mathbf{p}_0)/(\mathbf{l} \cdot \mathbf{l})$, and $\mathbf{l} = \mathbf{p}_1 - \mathbf{p}_0$.

12.2.2 Splines

For a parametric spline curve defined by the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ we find a point on the curve solving $\mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$, where $t \in [0, 1]$ is the

spline parameter. The closest point on the curve to a given point $\mathbf{x}$ is obtained from the condition $\partial|\mathbf{x}-\mathbf{q}|/\partial t = 0$.

$$\frac{\partial|\mathbf{x}-\mathbf{q}|}{\partial t} = 3\mathbf{a}\cdot\mathbf{a}t^5 + 5\mathbf{a}\cdot\mathbf{b}t^4 + 2(2\mathbf{a}\cdot\mathbf{c} + \mathbf{b}\cdot\mathbf{b})t^3 + 3(2\mathbf{a}\cdot\mathbf{e} + \mathbf{b}\cdot\mathbf{c})t^2 + (2\mathbf{b}\cdot\mathbf{e} + \mathbf{c}\cdot\mathbf{c})t + \mathbf{c}\cdot\mathbf{e} \tag{12.8}$$

where $\mathbf{e} = \mathbf{d} - \mathbf{q}$. The multiple roots of this equation must be tested to determine which one gives the closest point.

Figure 12.2 shows an example of primitive curve skeleton for a line segment.

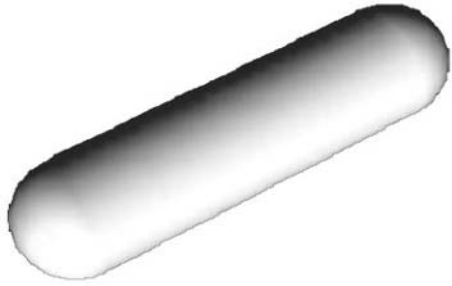


Fig. 12.2. Skeleton-based implicit objects of a line segment

12.3 Surface Skeletons

This primitive is defined as an offset surface constrained to lie at a fixed distance normal to another surface. The skeleton can be specified either in parametric or implicit form, providing a mechanism to incorporate parametric models in implicit based systems. In practice, the complexity of the distance calculation dictates only the implementation of simpler forms of this model which explore special cases.

12.3.1 Polygon

The distance from a point p to a polygon can be computed considering two cases. If p projects to the inside of the polygon it is the distance from p to the support plane of the polygon. Otherwise, it is the distance from p to the closest polygon edge or vertex.

Figure 12.3 shows an example of offset surface primitive for a triangle.

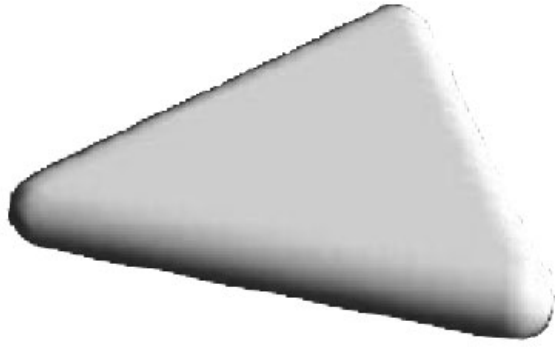


Fig. 12.3. Skeleton Based Implicit Objects

12.3.2 Height Field

If we allow the distance function to vary over a planar skeleton we obtain a height field surface.

12.4 Skeletons and Blending

Skeleton primitives can be naturally combined using blending operations. (See Chapter 8.)

The generalized distance $D(x)$ plays an important role in how the primitives are blended. In many cases, D can be defined as a monotonic modulation of the euclidean distance, (i.e. $D(d_e(x, S))$, where d_e is the euclidean distance). In the case of distance functions induced by a non-euclidean metric it is desirable for shape control that the distance is evaluated in the local coordinate system of the primitive [WW89].

12.4.1 Blending Schemes

As we have seen in Chapter 8, a blending scheme combines the functions of individual primitives such that their shapes are smoothly joined.

This functional composition is usually done using the linear blend or the convolution blend. Next, we will review these two types of blending and discuss how they apply to skeletons.

Linear Blend.

The linear blend is defined by summing distance functions to each skeletal element. The blending equation for a set of primitive skeleton elements $\mathbf{S}_i$ is

$$F(x) = \sum_{i=0}^{k} D(\mathbf{x}, \mathbf{S}_i) - c \tag{12.9}$$

where $D(\mathbf{x}, \mathbf{S})$ is a function that gives a measure of the generalized distance between the points $\mathbf{x}$ and the primitive skeleton $\mathbf{S}$.

Figure 12.4 shows an example of blending of point skeletons.

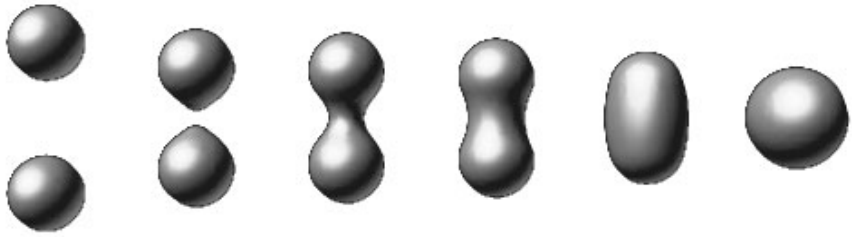


Fig. 12.4. Skeleton Based Implicit Objects

Linear blends work very well for point skeletons. However, when curve or surface skeletons are combined using linear blend, the resulting surface exhibits a bulge at the places where primitive elements meet [Blo97]. This problem can be solved using convolution blend.

Convolution Blend.

The convolution surface [BS91] is obtained via 3D convolution of a distance kernel $D(x)$ and a skeleton shape S, by integrating for all points $p \in S$

$$\chi_S(x) \star D(x) = \int_S \chi_S(p) G(p - x) dp \tag{12.10}$$

where χ_S is the characteristic function of the skeleton S.

Then, the implicit description is

$$F(x) = (\chi_S(x) \star D(x)) - c \tag{12.11}$$

where $c \in \mathbb{R}$ is the isovalue of the level surface.

The convolution operation gives a weighted average of the distances to points $p \in S$ on the skeleton that is equivalent to the union operation.

Since the convolution is a linear operator, the sum of individual convolutions of separate skeletal pieces is equal to the convolution of the union of all skeletal elements:

$$D(x) \star (\chi_{S_1}(x) \cup \chi_{S_2}(x)) = (D(x) \star \chi_{S_1}(x)) \cup (D(x) \star \chi_{S_2}(x))$$

Thus, the convolution blend can be computed simply by summing the convolutions of individual skeleton elements

$$F(x) = \sum_{i=0}^{k} (\chi_{S_i}(x) \star D(x)) - c \tag{12.12}$$

The superposition property of convolution implies that the blend acts in the same way on all points of the skeleton. Therefore, arbitrary division of a skeleton into pieces does not introduce any bulge in the surface near the joins of these pieces.

Note that for point skeletons, convolution blend is equivalent to the linear blend.

One difficulty with convolution surfaces is that, in general, there is no closed form solution to the convolution integral, for most skeleton shapes and distance kernels. For this reason, [BS91] suggested a numerical volumetric integration by parts using image processing techniques. Another option, proposed by [She99], is to carefuly design a kernel distance function that can be convolved analytically with basic skeleton shapes.

The Cauchy kernel is the most suitable for convolution blends. It allows a closed form solution of the convolution integral for five types of skeleton shapes: point, line segment, plane, arc, and triangle.

The Cauchy kernel function is

$$H(r) = \frac{1}{(1 + s^2 r^2)^2} \tag{12.13}$$

As an example we give the formula for the convolution of $H(r)$ with a line segment of lenght L, defined as

$$p(t) = o + tv \tag{12.14}$$

where o is the origin of the segment, v is the normalized line direction, and $0 < t < L$.

$$H(x) \star \chi_{p(t)} = \frac{a}{2p^2(p^2 + s^2a^2)} + \frac{L-a}{2p^2q^2} + \frac{1}{2sp^3}\left(\arctan\frac{sa}{p} + \arctan\frac{s(L-a)}{p})\right)$$

where $a = (o - x) \cdot v, \quad d = \|(o - x)\|, \quad p, q$ are

$$p^2 = 1 + s^2(d^2 - a^2), \qquad q^2 = 1 + s^2(d^2 + L^2 - 2La)$$

and we set the kernel width $s^2 = 0.25$.

The formulas for the other primitives can be found in [MS98].

Figure 12.5 shows an example of convolution blend.

Fig. 12.5. Convolution blend

This page intentionally left blank

13. Multiscale Implicit Objects

This chapter studies a model of implicit objects based on a multiscale decomposition of the implicit function.

This model is adapted to the local variations of the implicit function at multiple scales. It is described in terms of simple functions that are suitable for computation.

Decompositions of the implicit function for modeling purposes should reveal relevant aspects of the function, such as its variations (or its frequency content). It is also desirable that a functional decomposition results in a representation that exhibits spatial locality (an exception to this rule is the case where certain operations are performed more efficiently using a representation without this property, such as the Fourier series [TL93]).

Spatial and frequency localization are incompatible requirements. So, if we want a representation that provides frequency information and has spatial locality, we have to settle for a compromise. Good localization in both space and frequency is achieved by the wavelet transform that uses scaled "small waves" [Dau92].

In the next sections we will discuss functional decompositions that are related to wavelets and are adequate for modeling with implicit surfaces.

13.1 Multiscale Decompositions

The decomposition of a function in terms of elements at multiple scales provides a representation that reflects the function behavior over neighborhoods of variable size. It is a hierarchical structure which is adapted to the function variations.

Definition.

A multiscale decomposition is a description based on a set of functions localized both in space and scale. Under this representation a function $f(x)$ is expressed as a linear expansion over this set.

A suitable family of functions for this purpose can be generated by scaling and translating a single function $\varphi(x)$. A member of this family is denoted by φ_γ, where the index $\gamma = (s, u)$ specifies the scaling parameter s and the translation parameter u. Depending upon the properties of φ, different types of decompositions are produced.

We assume that both f and φ belong to $\mathbf{L}^2(\mathbb{R}^n)$, the vector space of measurable, square-integrable real functions of n variables. In the following exposition, we will consider only the one-dimesional case. The definition of a multiscale decomposition for n-dimensional functions $f : \mathbb{R}^n \to \mathbb{R}$ in $\mathbf{L}^2(\mathbb{R}^n)$ is a direct extension.

13.1.1 Dictionaries

The most general kind of decomposition uses the notion of a *dictionary*. $\mathcal{D} = \{(\varphi_\gamma, \widetilde{\varphi}_\gamma)\}_{\gamma \in \Gamma}$ is constituted by a family of dual functions, φ and $\widetilde{\varphi}$. In the case of a multiscale dictionary, the index set Γ is defined as above.

Given an arbitrary function $f \in \mathbf{L}^2(\mathbb{R})$, we select from the dictionary $\mathcal{D}$ a countable subset of elements $(\varphi_{\gamma_i}(x))_{i \in \mathbb{N}}$, with $\gamma_i = (j, k)$, such that f can be reconstructed by a linear expansion into these selected elements

$$f(x) = \sum_{i=0}^{\infty} a_i \varphi_{\gamma_i}(x),$$

where γ_i is an element of the index set Γ.

The coefficients a_i are computed by orthogonal projection on the duals of the selected elements φ_{γ_i} of the dictionary $\mathcal{D}$.

$$a_i = \langle f, \widetilde{\varphi}_{\gamma_i} \rangle$$

Intuitively, this decomposition indicates the features of f that are "in-tune" with each element φ_γ.

A multiscale representation is given by the list of coefficients a_i, together with the corresponding indices $\gamma_i = (j_i, k_i)$ in the dictionary $\mathcal{D}$.

In practice, we use only a finite number of elements from $\mathcal{D}$. For this reason, it is important to obtain a sequence of decompositions with increasing number of elements that converges to the function f:

$$\|f - \sum_{i=0}^{m} a_i \varphi_{\gamma_i}\| \leq \varepsilon \|f\|$$

This provides a mechanism to approximate f with a desired precision ε.

13.1.2 Non-Redundant Dictionaries

Two important issues arise in the context of multiscale representation:

- Which is the best multiscale decomposition of a function?
- How to perform computations efficiently with the decomposition?

Obviously, the answer to these questions depends on the choice of the multiscale dictionary $\mathcal{D}$. A qualitative or a quantitative criterion could be used choose $\mathcal{D}$: (i) The representation should be adapted to the function f, i.e., the one whose elements would identify features of interest in f; (ii) The representation should be compact, i.e., a linear expansion of f should have a small number of elements.

The dictionary $\mathcal{D}$ may be redundant, which implies that the multiscale decomposition is not unique in general. Therefore, a third criterion is the uniqueness of the representation of f For that we must choose a non-redundant dictionary.

Two particular multiscale dictionaries fulfill the three above criteria — they are based on scaling functions and wavelets.

In the following sections we will discuss in detail these two decomposition approaches and methods to compute them based on the wavelet transform.

13.2 Multiresolution Analysis and Wavelets

There is an intimate connection between wavelets and a multiresolution analysis. Wavelets can be used to generate a multiresolution analysis, or conversely, we can derive dyadic wavelets from a multiresolution analysis.

The idea is to decompose $\mathbf{L}^2(\mathbb{R})$ into a direct sum of closed subspaces W_j, spanned by the functions $\psi_{j,k}(t)$. Consequently, the complementary subspaces

$$V_j = \cdots \oplus W_{j-2} \oplus W_{j-1}, \quad j \in \mathbb{Z}$$

form a nested sequence of subspaces of different scales. This motivates the investigation of a *scaling function* $\phi(t)$ that generates the spaces V_j, $j \in \mathbb{Z}$, in the same manner that $\psi(t)$ generates the spaces W_j, $j \in \mathbb{Z}$.

13.2.1 Multiresolution Analysis

A multiresolution analysis (MRA) is a hierarchical decomposition of $\mathbf{L}^2(\mathbb{R})$ into a sequence of closed subspaces $\{V_j\}_{j\in\mathbb{Z}}$ satisfying

$$\cdots V_{-1} \subset V_0 \subset V_1 \cdots$$

with $\overline{\bigcup_{j\in\mathbb{Z}} V_j} = \mathbf{L}^2(\mathbb{R})$, $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$ and

$f(t) \in V_j \leftrightarrow f(2t) \in V_{j+1}$

$f(t) \in V_j \leftrightarrow f(t - 2^{-j}k) \in V_j$

The spaces V_j are called *approximation spaces*. In a multiresolution analysis, all approximation spaces are scaled versions of the central space V_0. For this, we require that there exists a function $\phi \in V_0$ of the form

$$\phi_{j,k}(t) = 2^{j/2}\phi(2^j t - k)$$

with $j, k \in \mathbb{Z}$, such that $\{\phi_{0,k} : k \in \mathbb{Z}\}$ is a Riesz basis in V_0.

The function ϕ is called a *scaling function* of the multiresolution analysis. A given space V_0 may have many functions that satisfy the above properties, but only one will be an *orthonormal basis* of V_0.

We denote by P_j the orthogonal projection operator onto V_j

$$P_j f = \sum_{k\in\mathbb{Z}} \langle f, \phi_{j,k} > \phi_{j,k}$$

This corresponds to the approximation of f at the resolution 2^j.

The fact that the limit of the sequence of spaces $\{V_j\}_{j\in\mathbb{Z}}$ is dense in $\mathbf{L}^2(\mathbb{R})$ ensures that $\lim_{j\to\infty} P_j f = f$ for all $f \in \mathbf{L}^2(\mathbb{R})$.

Although $\{\phi_{j,k}\}_{(j,k)\in\mathbb{Z}^2}$ spans the space $\mathbf{L}^2(\mathbb{R})$, it is not a minimal spanning set, due to the nested nature of the multiresolution analysis. On the other hand, it is possible to construct an orthogonal decomposition of $\mathbf{L}^2(\mathbb{R})$ if we exploit the complementary structure of a multiresolution analysis.

13.2.2 Detail Spaces

We define W_j to be the orthogonal complement of V_j in V_{j+1}. Then we have

$$V_{j+1} = V_j \oplus W_j$$

and

$$W_j \perp W_{j'}, \qquad \text{if } j \neq j'.$$

The spaces W_j are called *detail spaces*, in the sense that they contain the difference in information between the spaces V_j and V_{j+1}. Therefore, W_j contains the detail information needed to go from the approximation at resolution 2^j to the approximation at resolution 2^{j+1}.

Observe that, because the spaces $W_j \subset V_{j'} \perp W_{j'}$ if $j < j'$, then we have

$$V_j = V_J \oplus \bigoplus_{k=0}^{J-j-1} W_{J+k}$$

for $j > J$.

Also, the family of spaces $\{W_j\}_{j\in\mathbb{Z}}$ constitute an orthogonal decomposition of $\mathbf{L}^2(\mathbb{R})$ expressed as

$$\mathbf{L}^2(\mathbb{R}) = \bigoplus_{j\in\mathbb{Z}} W_j$$

The spaces W_j inherit from V_j the scaling property of the multiresolution analysis

$$f \in W_j \leftrightarrow f(2^j) \in W_0$$

It is possible to find a wavelet ψ such that, for fixed j, $\{\psi_{j,k} : k \in \mathbb{Z}\}$ constitute an orthonormal basis of W_j. Therefore, the whole family $\{\psi_{j,k}\}_{(j,k)\in\mathbb{Z}^2}$ constitutes an orthonormal basis of $\mathbf{L}^2(\mathbb{R})$.

13.2.3 Scaling Functions and Wavelets

We are now faced with the problem of defining the scaling and wavelet functions ϕ and ψ. This can be done exploiting the 2-scale relation that exists in a multiresolution analysis.

It is possible to find functions ϕ and ψ which satisfy the two-scale relations below:

$$\phi(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} h_k \phi(2t - k)$$

$$\psi(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} g_k \phi(2t - k)$$

and that generate an orthogonal basis of, respectively V_j and W_j.

Note that, ϕ and ψ are uniquely determined respectively by the l^2-sequences $H = \{h_k\}$ and $G = \{g_k\}$. Also, observe that ψ is defined in terms of ϕ.

H and G form a pair of discrete filters that can be used to compute the wavelet transform.

13.3 The Wavelet Decomposition

Wavelets can be used for constructing a multiscale dictionary. Since there exists wavelets $\psi(\mathbf{x})$ that form an orthonormal basis of $\mathbf{L}^2(\mathbb{R}^n)$, the associated dictionary is non-redundant.

The wavelet decomposition a function f is the projection of f onto the wavelet spaces W_j and is computed using the wavelet transform.

The wavelet representation consists of the coefficients $\{d_{jk}\}$ of the linear expansion relative to the wavelet basis ψ_{jk}

This representation can be interpreted as a decomposition of a signal in a set of *independent* frequency channels, as in Marr's vision model [Mar82].

13.3.1 The Wavelet Transform

An efficient method to compute the coefficients $\{d_{j,k}\}$ of the wavelet transform exploits the hierarchical structure of the multiresolution analysis. It is based on the two scale relations and the associated pair of discrete filters H and G.

We start with a function $f_J \in V_J$. Since $V_j = V_{j-1} \oplus W_{j-1}$, f_J has a unique decomposition $f_J = f_{J-1} + g_{J-1}$ where $f_{J-1} \in V_{J-1}$ and $g_{J-1} \in W_{J-1}$.

By applying this recursively N times we have

$$f_J = f_{J-N} + g_{J-N} + \cdots + g_{J-2} + g_{J-1}$$

for $f_j \in V_j$ and $g_j \in W_j$. This is called the *wavelet decomposition.*

Since $f_J \in V_J$, it can be represented as $(P_J f)(t) = \sum_k c_{J,k}\phi_{J,k}(t)$ with $c_{J,k} = \langle f_J, \phi_{J,k}\rangle$. But, as we have seen,

$$\langle f_J, \phi_J\rangle = \langle f_J, \phi_{J-1}\rangle + \langle f_J, \psi_{J-1}\rangle$$

where $\langle f_J, \phi_{J-1}\rangle$ and $\langle f_J, \psi_{J-1}\rangle$ are respectively the orthogonal projections on the spaces V_{j-1} and W_{j-1}.

From the two scale relations we have

$$c_{J-1,k} = \sum_n \overline{h_{n-2k}} c_{J,n} = \overline{H} c_{J,n}$$

and

$$d_{J-1,k} = \sum_n \overline{g_{n-2k}} c_{J,n} = \overline{G} c_{J,n}$$

where $c_{J-1,k} = \langle f, \phi_{J-1,k}\rangle$ and $d_{J-1,k} = \langle f, \psi_{J-1,k}\rangle$

Again, applying the above formulas recursively on the coarse sequence of approximation coefficients $\{c_{j,k}\}$ we get an algorithm to decompose the finer approximation coefficients into coarser approximation and detail coefficients. The algorithm is illustrated in the diagram below.

$$\begin{array}{ccccccc} \{c\}_j & \xrightarrow{\widetilde{H}} & \{c\}_{j-1} & \xrightarrow{\widetilde{H}} & \{c\}_{j-2} & \cdots \\ & \searrow^{\widetilde{G}} & & \searrow^{\widetilde{G}} & & \searrow^{\widetilde{G}} \\ \{d\}_j & & \{d\}_{j-1} & & \{d\}_{j-2} & \cdots \end{array}$$

13.3.2 Wavelet Implicit Models

Wavelet-based models of implicit surfaces and solids were proposed in [Mur93]. Muraki's model is a direct extension of the orthogonal two-dimensional wavelet decomposition to three dimensions. Because it uses a tensor product formulation, the representation is given in terms of 7 different wavelet functions, which complicates the computations with the model.

The wavelet representation is very effective in describing the variations of a function. This makes it attractive from the standpoint of volumetric processing. The main reason is that the wavelet is designed to detect changes of the function value at different scales. What it really encodes is the location and scale of these transitions. Some examples of applications of wavelets in volume rendering and compression are [GLDH97], [LG95], [IP98].

13.4 The Laplacian Decomposition

From the point of view of geometric modeling, a representation in terms of the wavelet coefficients is not so desirable. We are looking for a multiscale representation that is constructive, compact, and expressible in terms of a simple function. Intuitively, we want a description of a function as a summation of "blobs" of different sizes. In that sense, we want a representation in terms of a scaling function associated with a wavelet function.

The projection of f onto the spaces V_j gives the multiresolution analysis of f. Unfortunately, we cannot use this representation directly. The scaling function coefficients represent approximations of the function f at each resolution 2^j. It is clear that, as a whole, they do not constitute a linear expansion of f in terms of the scaling function ϕ.

Instead, we would like to obtain a representation that is a multiscale decomposition of the function f. Such a representation can be constructed from the wavelet decomposition. Observe that $V_j = V_{j-1} \oplus W_{j-1}$ implies that $W_{j-1} \subset V_j$. Therefore, W_{j-1} can be represented in terms of a basis of V_j without any loss of information. All we need to do is to project the wavelet coefficients in the subspaces W_{j-1} back to the subspaces V_j.

This representation is called the Laplacian decomposition' [Bur83]. It is equivalent to the wavelet decomposition but is given in terms of the scaling function [VTG94].

13.4.1 The Laplacian Transform

The Laplacian transform exploits the fact that, since $W_{j-1} = V_j \ominus V_{j-1}$, the coefficients $\{e_j\}$ of the Laplacian decomposition can be computed by subtracting the coefficients of the approximations at V_j and V_{j-1}. The algorithm is illustrated in the diagram below.

$$
\begin{array}{ccccccc}
\{c\}_j & \xrightarrow{\widetilde{H}} & \{c\}_{j-1} & \xrightarrow{\widetilde{H}} & \{c\}_{j-2} & \cdots \\
- & \swarrow^{H} & - & \swarrow^{H} & - & \swarrow^{H} \\
\{b\}_j & & \{b\}_{j-1} & & \{b\}_{j-2} & \cdots \\
= & & = & & = & \\
\{e\}_j & & \{e\}_{j-1} & & \{e\}_{j-2} & \cdots
\end{array}
$$

where it is assumed that we have computed the coefficients $\{c\}_J = \langle f, \widetilde{\phi}_{J,k} \rangle$ of the representation of f in the basis of V_J, such that f is written as $f(x) = \sum_k c_{Jk} \phi_{J,k}(x)$.

13.5 The Multiscale Representation

The results in the previous two sections allow us to create a implicit multiscale representation of the function f based either on the wavelet decomposition or on the Laplacian decomposition. Both representations are adapted to the variations of f at different scales.

The representation contains the coefficients a_i of the multiscale decomposition. The indices $\gamma_i = (j_i, k_i)$ relate these coefficients to elements φ_{γ_i} of the dictionary $\mathcal{D}$. Note that, in the wavelet decomposition, $\mathcal{D} = \{\psi_{jk}\}$, and while in the Laplacian decomposition $\mathcal{D} = \{\phi_{jk}\}$.

In order to produce the representation, we compute the coefficients a_i using the appropriate transform (wavelet or laplacian) and convert them to a suitable data structure.

13.5.1 Data Structures

We can use three alternative data structures to encode the coefficients of the multiscale decomposition:

- A list of the coefficients $l = \{a_i, \gamma_i\}$.
  The list structure is simply an enumeration of the coefficients of the multiscale decomposition a_i and the indices $\gamma_i = (s_i, u_i)$ corresponding to the functions $\varphi_{2^j,k}$ of the dictionary $\mathcal{D}$.
- A pyramid data structure, $A_j = (a_{j,k})$.
  The pyramid structure contains all coefficients associated with the basis functions $\varphi_{(2^j,k)}$ of the approximating spaces V_j. This is essentially an enumeration of the coefficients corresponding to all elements in $\mathcal{D}$, such that the functions that are not in γ_i have a coefficient $a_{j,k} = 0$. Such scheme establishes a one-to-one correspondence between the elements of $\mathcal{D}$ and the coefficients of the decomposition, and eliminates the need for including the indices γ_i in the representation.
- A spatial hash table $H = \{\uparrow a_i, \gamma_i\}$.
  The coefficients a_i may also be associated with a spatial hash table in which each cell has pointers to the elements whose support is in the cell. This cell complex can be formed by any adaptive subdivision of space, such as an Octree or a BSP tree.

The list structure is the most compact, but it requires extra processing to find the elements that should be evaluated at a given point. The pyramid structure has the advantage of simplicity and direct access but, since the multiscale decomposition is usually sparse, it has the disadvantage of using more space than necessary. The hash table structure offers a good compromise between access time and space.

In practice, the choice of a particular data structure will be dictated by the characteristics of the implicit function, as well as by the requirements of the application. The list structure is the best for models with a small number of coefficients. The pyramid structure is the best for models with uniform spatial complexity. The hash table structure is the best for models with non-uniform spatial complexity.

13.5.2 Conversion of Implicit Objects

To use the multiscale representation we need apply a conversion to other types of implicit models. In this process the decomposition is computed from an intermediate volumetric description. The method takes as input an n-dimensional array of discrete samples and generates a multiscale implicit representation.

The example shows the use of the method in three dimensions. The input is a 3D sample array that is converted to the Laplacian multiscale representation.

Example 13.1. Hypertexture Object

The object is a "noisy sphere", a procedural implicit shape. It is defined by functional composition of an object density function with density modulation functions [PH89]. In this example, the object density function is of a soft sphere and the modulation function is a bandlimited noise function.

Figure 13.1 shows the volume density array generated by the hypertexture procedure mentioned above. Figure 13.2 shows the points in the volume corresponding to the boundary and the interior of the object. Figure 13.3 is a ray-traced image of the noisy sphere. It was produced by rendering the Laplacian pyramid description of the data in Figure 13.1. Almost all the information is contained in the bottom level of the pyramid, indicating that most of variations of the implicit function are at that scale.

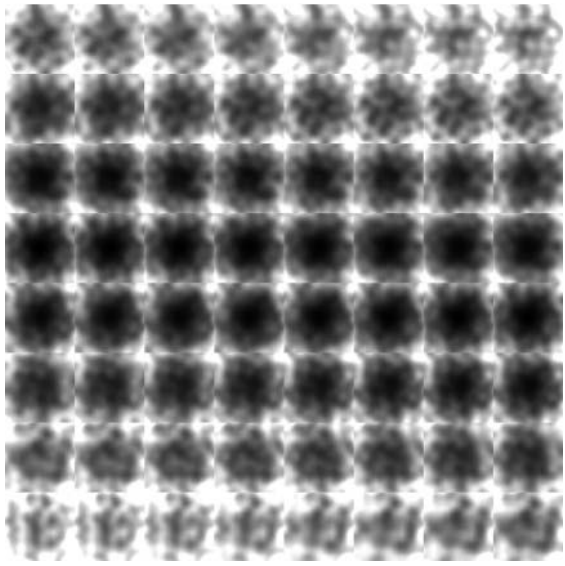


Fig. 13.1. Slices of the Volume Density Function for the Noisy Sphere

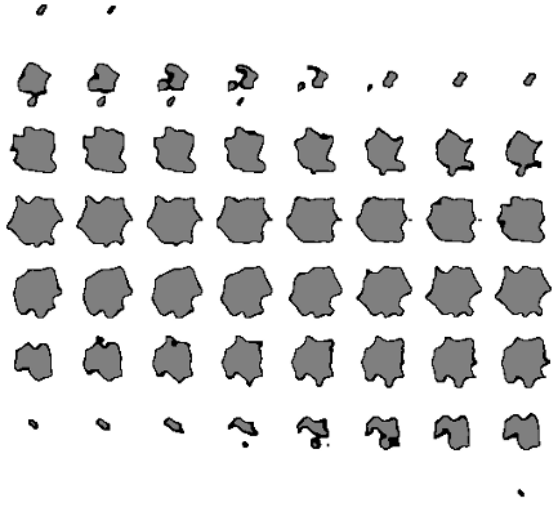


Fig. 13.2. Boundary and Interior Points of the Noisy Sphere

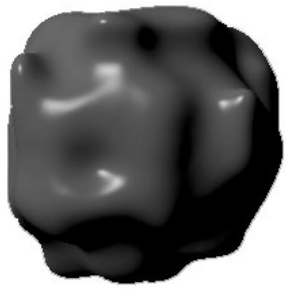


Fig. 13.3. Noisy Sphere, Raytraced from its B-spline Pyramid

Unstructured Decompositions

Structured multiscale decompositions have one property that may not be desirable in some situations. They are not invariant under fractional translations. This means that the representation of two identical shapes will be different if one is displaced relative to the other by a non-integer amount.

A possible approach to compute a non-structured multiscale decomposition that is translation invariant could be to employ a matching pursuit algorithm [MZ93]. This algorithm consists of an iterative procedure for minimizing the energy of the residual error between the input function and its linear expansion over a set of elements in the dictionary. In this case, the model would be formulated in terms of a dictionary consisting of functions that are the basis of "continuous" spaces in a dyadic scale sequence.

Unstructured wavelet decomposition have been used by [PZ90] to construct free-form implicit surfaces. Perlin's model uses 1-directional spline wavelet. His method employs an empirical procedure to construct the representation and the model is restricted to a particular level surface. This model, called surflet, consists of a set of wavelet functions which act as free-form splines approximating a localized piece of the surface. The sum of these surflets form the implicit function describing the whole surface.

Procedural implicit objects and voxel arrays have been successfully converted to the surflet representation, showing the potential of this model.

14. Modeling

This chapter is concerned with applications of implicit objects to modeling. We examine the various aspects related to the description and construction of these objects in a modeling system.

There are two main questions involved in the design of a modeling system: *representation* and *specification*. The model representation deals with the problems of how to characterize objects and convert this characterization into concrete structures. The model specification deals with the techniques employed to build these structures, as well as, with the user interface.

Figure 14.1 shows the structure of a modeling system.

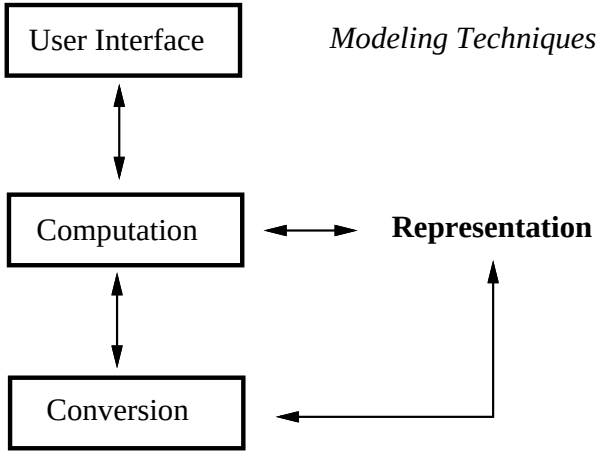


Fig. 14.1. Structure of a Modeling System

14.1 Representation Schemes

A representation scheme translates the abstract model of an object into a concrete description which allows the analysis of its properties.

Formally, a representation scheme is a relation $S : M \rightarrow R$ which maps models into representations. A model is a mathematical characterization of an object and a representation is a symbolic description of the object.

Let D be the domain of S, i.e., the set of all elements in M for which there is a corresponding representation in R, and let V be the image of S, i.e., the set of all elements of R which correspond to some element of M. Any representation in V is said to be a *valid representation*. We point out that, in general $D \neq M$ and $V \neq R$. That is, some objects in the space of models M are not representable, and there are some syntactically correct representations in R that are not valid. The *expression power* of a representation scheme S is the size of the subset D of models that can be represented by S. (See Figure 14.2).

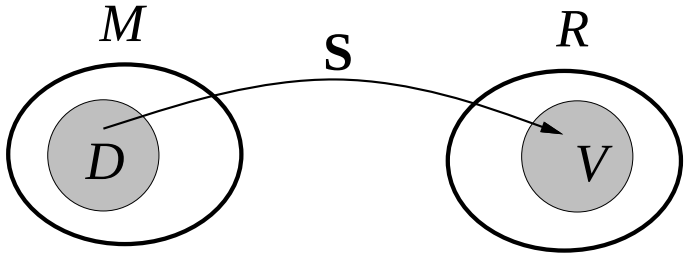


Fig. 14.2. Representation Scheme

14.1.1 Properties of a Repesentation Scheme

A representation scheme has formal and informal properties that should be taken into account when designing a modeling system [Req80]. The formal properties are *uniqueness* and *completeness*.

A representation scheme has the uniqueness property if each element in D has only one representation (i.e., S is not a one-to-many relation). A representation scheme is complete (or unambiguous) if

the inverse relation S^{-1} is not a one-to-many relation, that is, each valid representation in V represent only one model in the domain D (we remark that the inverse relation S^{-1} is very important, because it defines the correspondence between representations and models and therefore, it encapsulates the semantics of the representation).

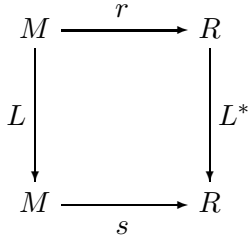
In most applications completeness and uniquenes are desirable properties of a representation scheme. However, uniqueness is very difficult, if not impossible, to attain, and therefore, we must still face the problem of *comparison of representations*, i.e., decide whether two given representations in some representation scheme correspond to the same model.

Other properties of a representation scheme are less formal but also relevant in modeling applications because they are related computational issues. Some examples of these properties are: *robustness*, i.e., sensitiveness to numerical errors; *conciseness*, i.e., size of the representation; *simplicity*,i.e., complexity of operations with the representation; and *precision*, i.e., whether the representation is exact or approximate.

14.1.2 Algebraic Structure of a Representation

When operations in the model space M have some invariance in relation to a representation space R, there is a well defined correspondence between these two spaces. Then, it is possible to create an algebraic structure for the representation scheme $S : M \to R$.

Let L be an operation in M, s be a representation of a model $m \in M$, and r be a representation of the model $L(m) \in M$. The representation is invariant under L if there exists a corresponding operation L^* in the representation space R, such that $L^* \circ r = s \circ L$. This means that the diagram below should commute.



Such operations L form an algebra that we can incorporate into the representation scheme.

For example, the universe P of point sets, with the operations of union $\cup$, intersection $\cap$ and complement $\neg$, define a Boolean algebra $(P, \cup, \cap, \setminus)$. In the context of implicit models, these operations on point sets have correspondent operations (i.e., $\min(a, b)$, $\max(a, b)$ and $\text{neg}(a)$) on implicit representations $F(x)$. Thus, they lead to a representation scheme with an algebraic structure.

Under this scheme, the representation of a model is an expression of the point set in this algebra. The implementation of such representation in a modeling system is usually done through a data structure called *parse tree* of the expression. It is an ordered binary tree where the terminal nodes represent primitive point sets and the non-terminal nodes represent point set operations. Figure 14.3 shows a parse tree and the corresponding algebraic expression:

$$\left(\bigcup \quad \left(- \quad \text{BLOCK} \left(\bigcap \quad \text{BLOCK} \quad \text{QUADRIC}\right)\right) \quad \text{BLOCK}\right)$$

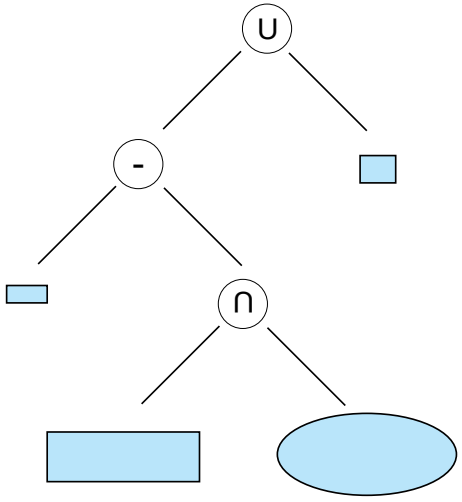


Fig. 14.3. Parse tree and algebraic expression:

14.1.3 Universal Representation

A *universal representation* is a pair $(\pi, \bar{R})$, where $\bar{R}$ is a representation space and such that for any representation space R, and representation scheme $S : M \to R$, $\pi : \bar{R} \to R$ is a surjective map satisfying $\pi \circ R = S$. (see Figure 14.4).

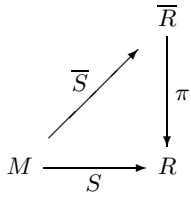


Fig. 14.4. Universal Representation

The representation $\bar{S}$ is called the *universal representation scheme* associated with the representation S. Note that the universal representation associated to some representation scheme is not necessarily unique, and, ideally, it should be possible to define a *canonical universal representation*.

Intuitively, the universal representation merges different representation spaces into the same space $\bar{S}$ in such a way that any representation scheme can be associated to a representation scheme on this space. This merging operation allows us to consider all representation schemes as being defined in the same representation space, and therefore problems that relate two or more representation schemes, such as representation conversion, can be well posed in this setting.

In the case of implicit objects, both the Medial Axis and the Distance Transform, discussed in Chapter 5, constitute universal representations. Moreover, these representation schemes are unique and complete.

14.2 The Implicit Representation

To study the shape and other material properties of physical three-dimensional objects, we characterize them as point sets in 3-space. In the context of implicit models, the geometry and topology of these point sets are defined mathematically by n-dimensional manifolds given in implicit form by a scalar function $F(x)$, as discussed in Chapter 6. The manifold condition can be relaxed in some cases when appropriate. Nonetheless, manifolds still constitute our fundamental geometric entity and the implicit form our basic building block.

The implicit representation is a functional constructive representation scheme, based on primitive point-membership classification functions, binary or n-ary composition functions, and unary modification functions.

Primitive objects, as defined in Chapter 11 can be analytical, procedural or sample-based. Modification operations, as presented in Chapter 7, can be spatial transforms, such as affine mappings and warpings, can be modulation functions. Composition operations, as discussed in Chapter 8 can be boolean set operations or blends.

Additionally, we can include special structures, such as blending graphs, for certain types of implicit objects like skeletons, discussed in Chapter 12.

14.2.1 Primitive Implicit Objects

Primitive solid objects are point sets defined as

$$\{(x, y, z) : F(x, y, z) \leq 0\}$$

The boundary of the primitive is a surface defined by $F(x, y, z) = 0$ and the interior of the primitive is the set of points satisfying $F(x, y, z) < 0$.

A family of objects is described as a *generic primitive* $\mathcal{O}(p_1, \ldots, p_n)$ that is instantiated to create individual objects. Most primitives are defined in terms of parameters, (p_i), which control their shape and other properties.

Primitives are often defined in a canonical coordinate system called local object space.

A characterization of the various families of implicit primitives was presented in Chapters 11 and 12.

14.2.2 Composite Implicit Objects

Composite objects are generated from primitive objects using either boolean (see Section 8.2) or blend (see Section 8.3) operations. They are defined by expressions involving primitives and the CSG operations of union, intersection and difference.

The standard representation of a composite object is a CSG expression implemented through an ordered binary tree as we discussed in the previous section.

14.2.3 Shape Modifiers

Unary operations, in principle, can be applied to any type of implicit objects (primitive or composite) to modify their properties.

These operations can change the domain or the image of the function F. Operations that change the domain are geometric transformations that deform the local ambient space where the object is embedded. Operations that change the image are modulate the value of the function to alter either boundary level set or the pseudo-metric induced by the function.

14.2.4 Groups of Objects

Objects can be grouped into different types of structures in order to make explicit relationships and constraints among them.

The most common grouping method is through a hierarchy which defines positional and orientation constraints between objects. It subordinates recursively the local coordinate frames of subgroups to the coordinate frame of the group. This type of structure is suited to describe complex objects built from simple parts, as well as, articulated objects connected by rotational or translational joints.

Arbitrary constraints can be represented by a directed graph. This structure is maybe too general. Cycles in the graph may cause problems. A good compromise is a directed acyclic graph (DAG). In a number of cases, it is possible to break the cycles and convert a general directed graph into a DAG. A specific type of constraint is the blending of primitives that belong to an skeleton. In this case, the structure is called *blending graph*.

Note that the group structure associated with unary modification operations and n-ary composition operations to be incorporated in a single graph structure that represents the object.

14.3 Auxiliary Representations

Auxiliary representations are used in a modeling system to complement the main representation. These secondary structures can provide either an approximate description of the objects or a means to perform computations more efficiently.

Two natural complementary representations associated with implicit objects are the space subdivision enumeration, discussed in Chapter 4, and the polygonal approximation discussed in Chapter 10.

14.3.1 Space Subdivision Enumeration

The space subdivision enumeration provides an appropriate structure to perform local computation with implicit objects. The structure can be uniform or adaptive. It can be use a single or multiple resolutions. Common structures are Voxel Arrays, Octrees and BSP trees.

14.3.2 Polygonal Approximation

The polygonization of an implicit object provides an approximate piecewise parametric representation that can be used for visualization purposes. Higher order approximations could be explored as a means to convert from implicit to parametric representation.

14.4 Conversion

Conversion between representations is a desirable capability in a modeling system. It allows to exploit the best characteristics of each representation, as well as, to import objects from other systems.

We distinguish between exact and approximate conversions. An *exact conversion* produces model of the original object, while an *approximate conversion* produces a model which approximates the original object.

Exact conversion between representations are usually not practical, except in some special cases [Mil89].

14.4.1 Implicit to Parametric

In general, is not possible to convert exactly an implicit to a parametric representation. This can be proved for the case of implicit algebraic surfaces and rational parametric surfaces.

Some types of implicit objects that can be converted exactly to a parametric form are the conics and quadrics.

However, it should be mentioned that polygonal (or higher order) approximations constitute a parametric model that approximates the implicit object.

14.4.2 Parametric to Implicit

Theoretically it is possible to do an exact conversion of an algebraic parametric representation into an implicit representation. This process is called implicitization and is based on classical elimination theory. A rational parametric description can be converted to implicit form using resultants [MC92] or Groebner bases techniques [SA84],

In practice, this type of conversion is not really used, because it produces algebraic implicit forms of very high degree for most cases of interest. For example, consider a rational parametric tensor product surface given by polynomials in u and v of degree m and n. The corresponding implicit surface is given by a polynomial equation of degree $2mn$. Thus, a bilinear patch leads to an implicit surface of degree 2, a biquadratic patch to an implicit surface of degree 8, and a bicubic patch to an implicit surface of degree 18.

A method for approximate conversion of parametric to implicit forms is presented in [VG96], The conversion is performed by first producing a volumetric representation of the characteristic function of the solid region enclosed by the parametric surface.

The volumetric representation is produced in two steps:

1. Rasterization the parametric surface.
2. Filling the volume enclosed by the surface.

There are standard algorithms for these two tasks [Kau87], [Pav78].

After the discrete characteristic function has been computed, a smoothing procedure based on multiscale edge detection is applied to generate the implicit function. The procedure employs a dyadic wavelet analysis and allows the control extent of smoothess relative to the tubular neighorhood of the shape.

14.5 Model Specification

A geometric modeling system must provide the means for creation, modification, and access to the representations of objects. Objects are specified through an input description that can be procedural, interactive, or a combination of both. Procedural schemes are essentially programming languages based on modeling commands. Such languages may include advanced algorithmic structures, such as flow control and functions. Interactive schemes are graphic programs that present the user with a set of operations for design and modification of objects. These two schemes are complementary and can be integrated under a single user interface, providing a powerful model specification mechanism.

Another important aspect of the model specification problem refers to paradigms for model construction. The *modeling techniques* constitute the basic methodology for object creation. Below we discuss some of these techniques in the context of modeling of implicit objects.

14.5.1 Constructive Techniques

Constructive modeling techniques are used to construct objects by fitting together basic parts that form a composite structure.

This method is closely related to the CSG scheme [WK85]. Examples of systems that are based on constructive functional techniques are the F-Rep system [PS95, PASS95], and the Blob system [WGG99]. They both use a textual language interface, as well as, interactive graphic model techniques. F-Rep works with R-functions and has its own language. The Blob system uses the Python extension language.

14.5.2 Free-Form Techniques

Free-form modeling techniques are used to build objects by glueing pieces and adapting them to fit the desired shape.

This method is normally associated with the parametric form, but it can applied to the implicit form as well. Implicit objects can be defined by functions of space in a piecewise manner allowing a large degree of local control [Las85] [Vel90b].

A very effective way to develop free-form implicit modeling techniques is through a sample-based implicit primitives, where samples on the surfaces provide a handles to model the shape [TDOY01].

Another method is to specify the implicit object through a parametrized model, where the parameters control the shape locally and in a differential manner [WH94].

14.5.3 Physically-Based Techniques

Physically-based techniques can be used in various ways for modeling purposes. It has been emploied to reconstruct real objects from acquired three-dimensional data [SP91]. Many other applications of physically-based techniques are possible in this context making this a fruitful area of research.

This page intentionally left blank

15. Visualization

This chapter discusses the visualization of implicit objects. It describes the various techniques available to render implicit objects.

Visualization plays a fundamental role in most applications. Practically every system that deals with implicit objects needs to use it.

There are several aspects to consider: different rendering styles, from simple drawings to photorealistic simulation; different modes of usage, from interactive display to off-line rendering.

Also, visualization of implicit objects integrates many geometric representations, such as points, lines, surfaces and volumes.

15.1 Points

The simplest and most efficient visualization technique is to use points. Implicit objects can be displayed as a cloud of particles distributed over its surface.

This technique consists of the following steps:

Algorithm 1 : Point Rendering

1. Generate a set of particles in space near the object.
2. Move particles towards the object's boundary until they reach the surface.
3. Display the particles as a set of points.

The spatial subdivision enumeration associated with the implicit object can be instrumental in generating particles near the boundary of the object, and, consequently, supplying good initial values for the subsequent step. If that is not available, the object's bounding box could be employed.

In order to make particles migrate to the boundary of the object a dynamical particle system can be used, as described in Chapter 9. Particles move along a modified gradient field associated with the implicit object to find an equilibrium on its surface.

When particles are projected on a two dimensional screen, the density of points is usually higher along singularities and silhouette curves, helping shape perception. Additionally, a depth cueing effect is obtained if closer particles are rendered as brighter points. This contributes to enhance the three dimensionality of the image.

This visualization technique is particularly well suited to interactive modeling tasks because of its simplicity and efficiency [Blo90]. In this case, the dynamical particle system may be permanently active so that when objects are modified the points are automatically updated.

Stochastic point sampling can also be used to produce point rendering [TSYK01].

Figure 15.1 shows an example of visualization using points.

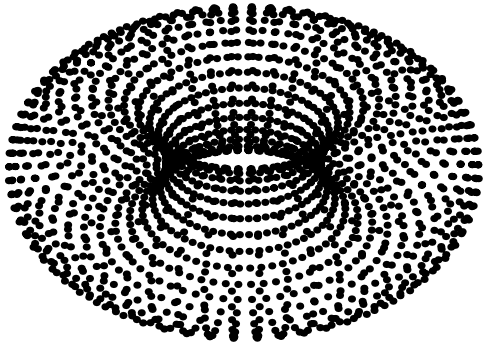


Fig. 15.1. Point display

15.2 Curves

An effective technique to display shape information is using line drawings. If lines are carefully selected, the resulting drawings convey the main features of the object with few strokes.

The problem is how to determine which curves can best depict the important geometric aspects of an object. Two classes of such lines are silhouette and contour lines.

15.2.1 Silhouette Curves

A silhouette point on a surface M is a point $p \in M$ such that the line from the observer's eye to p is tangent to M. A *silhouette curve* on M is a curve constituted by silhouette points. The most common silhouette curves delimit the boundary between the object and the background. They are defined as the set curves in which either the dot product between the surface normal and the vector in the viewing direction is equal to zero, or the surface normal is discontinuous.

These curves can be computed from a polygonization of the surface or using ray casting [Rot82], [RO87].

Silhouette curves can be complemented by internal hatching lines [BH98].

15.2.2 Contour Curves

Contour lines are obtained by intersecting the boundary of the object with a series of planes perpendicular to the line of sight and receding from the viewpoint. It has been suggested that these lines provide an exact unambiguous geometric interpretation, and are among the most useful in determining a shape.

[Blo90] describes a procedure to compute the contour lines corresponding to an implicit surface. Given an implicit object, its associated spatial decomposition enumeration and a set of parallel planes L_i, the contour curves are computed according to algorithm 2:

Algorithm 2 : Contour Rendering

for (each transverse cell c_j that intersects a plane L_i) **do**
Find the intersection points p_1 and p_2 between the affine approximation of F on the faces of c_j and the plane L_i.

Refine the line $\overline{p_1 p_2}$, using binary subdivision, according to the local curvature of the surface.

Line drawing techniques are very attractive for illustration purposes. Figure 15.2 shows a line drawing of an implicit object using silhouette curves.

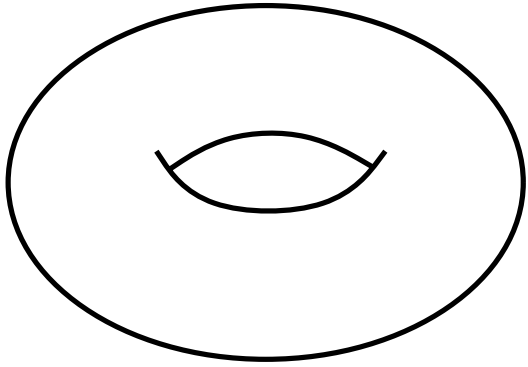


Fig. 15.2. Silhouette curve display

15.3 Surfaces

A realistic rendition of an object is achieved through a visualization of its surface using an illumination model. For this purpose, local or global shading methods can be applied directly to the implicit description of the object or to a parametric approximation of it.

15.3.1 Scan Line Methods

[SZ89] developed a visible surface algorithm for implicit algebraic surfaces. The basic structure of the method is shown in algorithm 3.

Algorithm 3 : Scan Line Rendering

for (each scan line s_i of the image) **do**

Compute the span of visible points by intersecting the surface with a plane defined by the viewpoint and s_i.

Interpolate points within the span.

Perform the shading calculations (usually local).

In the case of quadric surfaces, the above equation has a closed form solution. In the case of more general surfaces, the solution must be obtained numerically. A similar method is described in [Bli82].

15.3.2 Polygonal Rendering

A piecewise linear approximation of the boundary of the implicit object can be used with polygonal rendering algorithms to visualize the surface of the object. This approach makes possible to incorporate implicit objects into polygon-based systems. It also allows us to take advantage of special purpose display hardware available on graphics workstations.

Figure 15.3 shows the rendition of a polygonal approximation of the implicit surface using flat shading.

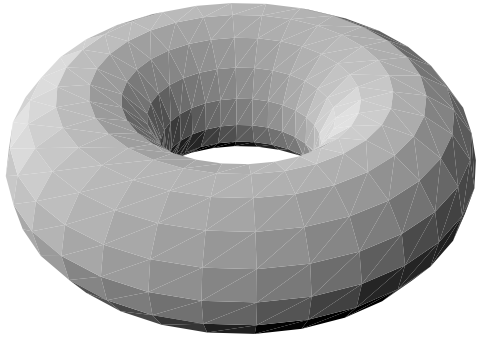


Fig. 15.3. Rendering of a polygonal approximation of the implicit surface

15.3.3 Ray Tracing

Ray tracing is the traditional method to render implicit objects [Mat68]. The basic ray tracing method is shown in algorithm 4.

A survey of the ray-surface intersection computation methods can be found in [Han88], [Bar86].

A ray tracing method with anti-aliasing is presented in [Har96]. The method is called *sphere tracing*.

Algorithm 4 : Ray Tracing Rendering

for (each point $[x, y]$ of the image) **do**

Compute the closest intersection of a ray from the viewpoint with the surface.

Perform the shading calculations (normally global, using recursive ray tracing).

The space subdivision enumeration associated with the implicit object can be used to accelerate ray tracing computations [AK89, Arv90].

Figure 15.4 shows the rendition of an implicit surface using ray tracing and Phong shading.

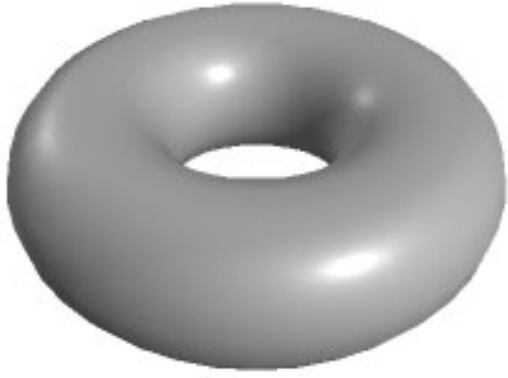


Fig. 15.4. Surface display with ray tracing

15.4 Volumes

There are two situations in which the visualization techniques described above are not adequate. The first is when the implicit object becomes so complex that the notion of a surface breaks up. The second is when it becomes necessary to display some spatial property of the implicit function F. In both cases, the solution is to use volume visualization techniques. They deal with the problem by rendering volumes

as textured density functions. In this way, whenever a surface exhibits geometric variations bellow a certain level of detail, it is treated as texture volume representing the collection of surface elements contained in that region. This also permits to depict continuously varying space functions.

15.4.1 Slice Rendering

Volume data, sometimes, is difficult to interpret if rendered as a whole. A simple technique to help inspect the properties of a space function consists in display its cross sections as planar images. A slicing plane can be moved through the volume revealing a local view of the volume.

Figure 15.5 shows an example of slice visualization.

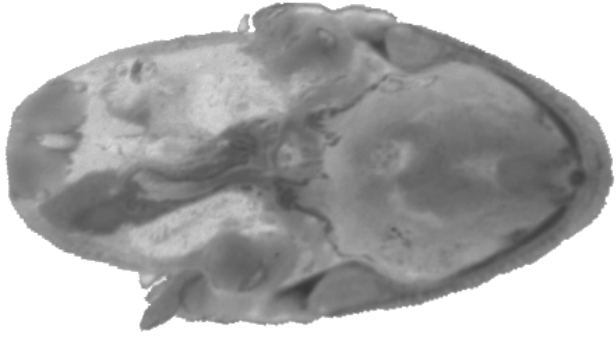


Fig. 15.5. Slice Visualization

15.4.2 Volume Rendering

The two main approaches used for volume rendering are: projection methods and ray casting [WG91].

Projection Methods. In projection methods the volume is decomposed into a three dimensional array of voxels. Each voxel is projected onto the image plane and its contribution to the image is calculated [DCH88].

Volumetric Ray Casting. In ray casting methods, rays from the viewpoint through each pixel are cast out into the volume. The contribution to the pixel is calculated integrating the density function along the ray [Lev90].

Figure 15.6 shows an example of volume the visualization techniques.

Fig. 15.6. Volume visualization

15.5 Visualization Modes

There are two basic visualization modes:

- interactive;
- non-interactive;

Interactive visualization requires that rendering is done in real-time to provide visual feedback to some interactive task. In this case, the most important aspect is the synchronization between user control and visual response.

Non-interactive visualization emphasizes high-quality rendering. It often employs sophisticated lighting models, as well as, antialiasing.

15.5.1 Progressive Refinement

Progressive refinement techniques offer the best compromise between rendering efficiency and image quality. This visualization strategy consists in starting with a simple, and fast to compute, rendition of the scene that is gradually improved by adding more features and detail [BFGS86]. This technique is particularly effective in interactive modeling situations, where idle cpu cycles can be used to refine the image.

15.6 Texture Mapping

Because implicit objects are not defined through a parametrization, texture mapping is not as straightforward as with surface patches.

One strategy to apply 2D textures onto implicit surfaces focus on implicit to parametric conversion techniques. Athough there isn't a generalized conversion method, it is reasonable to follow this approach, since applying bidimensional textures onto parametric surfaces is a "solved problem". Pedersen [Ped95, Ped96] introduced a method which estimates geodesics on an implicit surface and performs local parameterizations creating patches over it.

At first sight it is unclear how to globally assign texture coordinates to an implicitly defined surface. But, fortunately there are a few natural solutions for this problem. One is based on solid texture and the other on projection mappings. A hybrid between these solutions is particle based texturing.

15.6.1 Solid Texture

This method uses a three dimensional function to define the texture. For this reason, the generation of texture coordinates is a trivial problem – usually is the identity map. Most of the effort goes into constructing the texture itself. It can be described by a procedural function or by a voxel array. Perlin [Per85b] and Peachey [Pea85] introduced the idea of solid textures. Although limited to materials that have a 3D structure, this technique is very usefull for texture mapping implicit surfaces. [WMW87] discuss some of the aspects involved in solid texturing implicit primitives.

Figure 15.7 shows an example of solid texture.

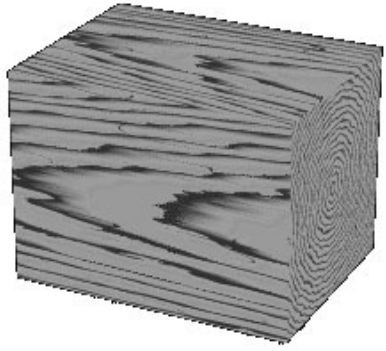


Fig. 15.7. Solid texture

15.6.2 Projection Mapping

This method uses bidimensional textures, normally given as images. It associates points in texture space with points on a surface in three space using projection mappings [Bar83]. Intuitively, a projection mapping can be understood in the following way: Associate the texture to a standard surface which has a simple parametrization. Then, project rays from each point of the surface into a three dimensional space where the object to be textured is defined. The intersection of these

rays with the object gives the texture coordinates. This is how a slide projector works. Note that all points along a ray have the same texture coordinate. This approach presents a critical problem: the rays as a whole may not reach all surface points from the outside, making impossible the idea of wrapping the surface with the texture map.

The texture mapping function is defined by choosing different standard surfaces and projection methods. Some simple examples are: the orthogonal, cylindrical and spherical projections. Figure 15.8 shows an example of texture mapping using orthogonal projection.

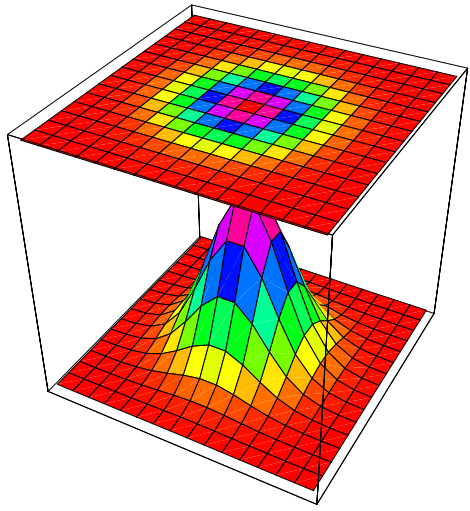


Fig. 15.8. Projection mapping

15.6.3 Particle-Based Texturing

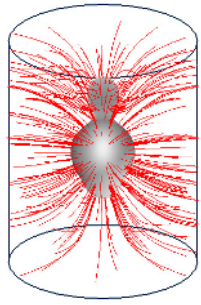
Particle-based texturing allows the application of a 2D texture to implicit surfaces in a natural way [ZGVdF98]. Moreover, it can be applied equally well to primitive or composite implicit objects with blending control while maintaining texture consistency [ZGV$^+$98, TW99].

The method establishes a correspondence between points on the implicit surface and texture attributes with the help of a dynamic particle system. The implicit surface's gradient is interpreted as a force field and used to govern the particle system. Points on the implicit

surface are treated as particles that move in the direction of a support object where the texture is defined beforehand. The texture attribute at each point of the surface is associated with the location of the corresponding particle on the support surface.

Figure 15.9 illustrates the method.

(texture)



(particle texturing)

(textured object)

Fig. 15.9. Particle Texture Mapping

s

16. Animation

This chapter investigates some aspects of animating with implicit objects. We are mainly concerned with how intrinsic characteristics of these objects can be applied to motion specification problems. For this purpose, the chapter starts with a short introduction to animation. Subsequent sections situate implicit objects into this context.

16.1 Animation Concepts

Animation is the study of time-varying phenomena. This discipline deals with the modeling, specification, control, and simulation of temporal behaviour.

The animation problem, in general, involves a system

$$D(q_1(t), q_2(t), \ldots, q_n(t)) \tag{16.1}$$

where the parameters q_i change with time $t \in \mathbb{R}_+$.

The system D may be very complex, comprising several objects O_i interacting in some environment. The system incorporates also a set of rules, R_j which govern the dynamic interaction between the objects. The parameters q_i are related to dynamic properties of the objects in the system.

We can classify animation systems based on three criteria: how the objects O_i are described; what class of rules R_j are used in the system; and which mechanism is adopted to control the parameters q_i.

16.1.1 Geometric Description

One of the main properties of an object is the geometric description of its shape. For animation, we are interested in the relation between shape and motion parameters. According to this criterium, we can distiguish three types of objects.

- **Particles**: are infinitesimal objects associated with a position in space. Geometrically, a particle is defined as a point $p \in \mathbb{R}^3$. The motion of a particle can be specified by a time-varying translation vector $u(t)$.
- **Rigid Bodies**: are solid objects whose shape cannot be deformed. The geometry of a rigid body is defined as a fixed compact point set $S \subset \mathbb{R}^3$ in space. The motion of a rigid body can be specified by a time-varying rotation and translation $\{R(t), u(t)\}$.
- **Deformable Bodies**: are objects whose shape changes over time. The geometry of a deformable body is more complex to define since deformation parameters have to be incorporated into de model. Nonetheless, it is possible to decompose the animation of a deformable body into a rigid motion $\{R(t), u(t)\}$ that changes only its local coordinate system and a warping $W(p, t)$ that acts on points $p \in O$ of the object.

16.1.2 Animation Rules

Animation rules determine how the interaction between the objects and the environment is simulated in the system. They are related to the nature of the control parameters. There are three types of simulation levels:

- **Kinematic**: this type of simulation controls directly the change of the parameters by specification of velocities.
- **Dynamic**: this type of simulation controls the rate of change of the parameters, by the specification of accelerations and torques.
- **Behavioral**: this type of simulation controls higher level parameters that cause self-motivated autonomous motion. These controls can be specified in the form of tasks or goals.

Note that these three types of simulation form a hierarchy of control levels.

16.1.3 Object Properties

A dynamic system models the effects of time change by applying the animation rules to the objects in the environment. Depending on the object description and type of simulation level, different properties of the objects are considered, as shown in the table below.

	kinematic	*dynamic*	*behavioral*
particles	position	mass	relation
rigid bodies	position, orientation	mass inertia	pose
deformable bodies	position, orientation, displacement	mass inertia plasticity	emotion

16.1.4 Composite Objects

Complex objects can be built from simpler shapes that are associated with each other by a structural relations. Composite objects are classified according to the nature of relationship and geometric description.

- **Particle System**: consists of a finite set of particles whose behaviour is governed by a function of time. In a physical particle system, the particles have masses and the Newtonian mechanics dictates their dynamical behaviour. The motion of a particle depends on its mass, position, velocity, and also on the forces acting on it, either by other particles or by the ambient medium.
- **Articulated Structure**: consists of a set of rigid body parts linked together by mechanical joints that restrict the motion of one part relative to the other.
- **Spring-Mass Model**: consists of a physical system of particles that are connected in pairs with springs. These springs impose internal forces that depend on the distance between the pair of particles, and drive the global behaviour of the system. The resulting structure can be represented as a graph where each particle is a node and two nodes are connected when there is a spring joining the corresponding particles. This type of model offer an alternative for the description of deformable shapes [TPF89b], [DG95, DCG98].

16.1.5 Constraints and Interference

Constraints can be used to restrict the motion of a shape relative to the others in the environment or in a composite object. In that way, they define degrees of freedom that constitute new animation parameters.

There are two categories of constraints:

- **Kinematic Constraints**: which impose some geometric relationship between the local coordinate systems of different objects. Usually these are hard constraints defined by a parametrized rigid motion.
- **Dynamic Constraints**, which impose forces that act objects on the environment. Usually these are soft constraints.

An additional problem in animation is the treatment of interference among objetcs in the environment. This leads to collision detection and the computation of reaction forces.

16.1.6 Control Modes and Simulation

The control mode determines how the animation parameters are specified and also how the system computes the animation. There are two basic control modes:

- **Indirect Control**: in this mode, the simulation formulated as an initial value problem. The initial state of the system (e.g. the value $p_i(t_0)$ of the parameters at $t = 0$) is specified and the temporal evolution of the system is calculated to the subsequent times $t = t_1, \ldots t_n$. The animation system employs a *forward* simulation, which can be kinematic or dynamic. It is also possible to have some parameters that are used to control the system as the simulation progresses.
- **Direct Control**: in this mode, the simulation is formulated as a boundary value problem. The configuration of the system is specified at some time instants, and the proper change of the parameters must be calculated for the in-between times, such that the temporal evolution reaches the desired configurations at the right times. The animation system employs a *inverse* simulation, which can be kinematic or dynamic.

16.2 Animated Implicit Skeletons

Skeleton based implicit primitives provide a natural way to control animation. They are particularly suited to the description of amorphous time-varying shapes, such as liquids, as well as, moving articulated objects, such as 3D cartoon characters and human models.

16.2.1 Particle Systems

Point-skeleton primitives can be used to represent objects constituted by particle aggregates. The model parameters determine how particles blend together, allowing the description of different types of material. Animation is generated by simply updating the parameters of all particles at each time step. The motion of particles may be controlled in a variety of ways to achieve the desired effect. Animation techniques include: spline interpolation, particle systems and dynamic simulation [WMW86a] [TPF89a].

16.2.2 Articulated Objects

Articulated objects can be modeled as a hierarchy of skeleton primitives. The implicit representation unifies the motion control and shape description of this kind of objects. The global blend capability ensures that a smooth continuous surface covers the whole object. Additional control over the blending of primitives is provided by organizing them into groups. Each group is designated a list of which other groups to blend with. Animation is generated by controlling the parameters of the skeleton structure at each joint. Then, local transformations are determined and applied to primitives.

16.3 Dynamic Simulation

In spite of the large body of work in the area of physically based animation, very few attempts have been made to apply dynamic simulation techniques to implicit objects. In [PW89] modal analysis is used to compute the dynamics of objects, generating polynomial deformation mappings that are coupled to volumetric models. In [VdMG91a] spring-mass models are used to determine piecewise deformation mappings that are applied to implicit objects.

Collision detection and avoidance is an important component of dynamic simulation. This problem can be posed as one of intersection determination in four dimensional space. This will be discussed in more detail in Chapter 17.

A different approach to collision detection and modeling of contact surfaces is proposed in [CG98], [CGD97]. The implicit model is used in an integrated way for the computation of collision, contact and reaction forces.

16.4 Metamorphosis

Metamorphosis is the transformation of one shape into another. Since this transformation is time-varying it can be posed as animation problem. This is a hard problem mainly due to the difficulty of finding correspondences between local features in both shapes. Not to mention the aspect of controlling the transition between them.

The implicit formulation is specially adequate for animating metamorphosis because objects are defined by a global function of space, which provides a straightforward mechanism correlation and interpolation.

16.4.1 Correspondence

When objects are composed of more than one element, a correspondence between shape elements must be found before the interpolation is applied. Depending on the type of the objects involved, different set of criteria should be used to associate pairs of elements. A general rule is based on the spatial location of the elements in each configuration. In the case of articulated objects, the topology of their hierarchical structures provide a good starting point. [Wyv90] gives a number of heuristics for correspondence determination.

16.4.2 Interpolation

The metamorphosis of an implicit object can be animated by interpolating between its initial and final shapes. These two shapes are blended together and mixed in different proportions. The transformation is accomplished by decreasing the contribution of the first shape while increasing the contribution of the second.

Since the implicit form is closed under functional composition the characteristic functions of various shapes can be combined very easily.

Figure 16.1 shows the metamorphosis of one implicit object into another using interpolation techniques.

Fig. 16.1. Metamorphosis using interpolation

Some examples of the research in this area are [TO99], [LGL95], [Hug92], [BW01a].

This page intentionally left blank

17. n-dimensional Implicit Problems

An implicit equation is defined by $F = 0$, where F is a function. The importance of implicit equations for geometric problems, and in particular for geometric modeling, is that it is a very natural and simple mathematical model to express geometric constraints. Geometric constraints are used to describe point sets, i.e. geometric objects. Sometimes we are interested in the objects by themselves and sometimes we use them as an aid to solve some geometric problem.

As an example the constraint that a point $P = (x, y)$ should be a a fixed distance R from the origin is expressed by the implicit equation

$$x^2 + y^2 = R^2.$$

We might be interested in the circular shape, or we just need that some point stay in a circular trajectory.

A problem that can be stated using an implicit equation $F = 0$ is called an *implicit problem*. The implicit problem is n-dimensional if the function F is defined in $\mathbb{R}^n$.

In order to solve an implicit problem, we need methods and algorithms to deal with implicit equations $F = 0$. A substantial effort has been done by the Purdue group in order to obtain robust solutions for implicit problems. The expression "dimensionality paradigm" has been coined in order to describe the techniques and algorithms developed to get robust solutions to implicit problems.

17.1 Example of Implicit Problems

In this section we will describe some examples of implicit problems that arise in different applications.

17.1.1 Offset surfaces

Given a surface $f = 0$, its r-offset consists of the points

$$\mathrm{Off}(f, r) = \{p : d_f(p) = r\},$$

where $d_f(p)$ is the Euclidean distance of the point p from the surface $f = 0$.

17.1.2 Voronoi Surfaces

Given two surfaces $f = 0$ and $g = 0$, the *Voronoi surface* of f and g is the set of all points in the space that have equal distance from either surface. Formally the Voronoi surface is defined by

$$\mathrm{Vor}(f, g) = \{p : d_f(p) = d_g(P)\}.$$

Voronoi surfaces are closey connected with the computation of the Medial Axis Transform associated to a givem model.

17.1.3 Variable Radius Blend

Given two surfaces $f = 0$ and $g = 0$, a *blending surface* is a surface F that intersects both f and g tangentially along some curves. In [Hof90] it is shown how the problem of surface blending can be reduced to an implicit problem, using Voronoi surfaces.

17.1.4 Shadow Computation

The shadow silhouette produced by a point light source l and cast by one sphere s_1 of radius r_1 onto another sphere s_2 of radius r_2, can be expressed as a level set of the function $F : \mathbb{R}^6 \to \mathbb{R}^5$ [DLTW90]. The components of $F = [f_1, f_2, f_3, f_4, f_5]$ are:

$$f_1(\mathbf{p}_1, \mathbf{p}_2) = (x_1 - cx_1)^2 + (y_1 - cy_1)^2 + (z_1 - cz_1)^2$$
$$f_2(\mathbf{p}_1, \mathbf{p}_2) = (x_2 - cx_2)^2 + (y_2 - cy_2)^2 + (z_2 - cz_2)^2$$

where f_i is the square distance from the point $\mathbf{p}_i = [x_i, y_i, z_i]$ to the center $\mathbf{c}_i = [cx_i, cy_i, cz_i]$ of the sphere s_i;

$$f_3(\mathbf{p}_1, \mathbf{p}_2) = (x_1 - cx_1)(x_1 - lx) + (y_1 - cy_1)(y_1 - ly) + (z_1 - cz_1)(z_1 - lx$$

where f_3 is equal 0 only if the vector from $\mathbf{p}_1$ to $\mathbf{c}_1$ is perpendicular to the vector from $\mathbf{p}_1$ to the light source $l = [lx, ly, lz]$, and

$$f_4(\mathbf{p}_1, \mathbf{p}_2) = (x_1 - lx)(y_2 - ly) - (x_2 - lx)(y_1 - ly)$$
$$f_5(\mathbf{p}_1, \mathbf{p}_2) = (x_1 - lx)(z_2 - lz) - (x_2 - lx)(z_1 - lz)$$

where f_4 and f_5 are equal 0 only if p_1 and p_2 are collinear with the light source.

The level set of $F(\mathbf{p}_1, \mathbf{p}_2) = [r_1^2, r_2^2, 0, 0, 0]$ gives all pairs of points such that the first is on s_1 and the second corresponds to its shadow on s_2. The shadow silhouette is the $\mathbf{p}_2$-component of this level set.

17.1.5 Collision Detection

Collision detection can be formulated as an intersection problem in four dimensions. The analysis of interference between moving objects is based on the three-dimensional volumes swept through time along their paths of motion [MS90], [Cam90]. The set of all points in the trajectory of an object can be formulated implicitly as follows.

By allowing a surface $F(x, y, z) = 0$ to move, a family of surfaces $F(x, y, z, t) = 0$ is generated. The envelope of $F(x, y, z, t) = 0$ encloses all the members of this family. An envelope surface is a surface which is tangential to all members of a family of surfaces. This means that, at time t, the corresponding member of the family will touch the envelope surface in some curve.

This hypersurface in four dimensions can projected back into three dimensions maintaining its implicit form. Consider two members of this family at times t and $t+\Delta t$. The first has equation $F(x, y, z, t) = 0$, while the second $F(x, y, z, \Delta t) = 0$. In order to meet both surfaces, the envelope must satisfy both equations. Expanding the latter in terms of Δt and letting $\Delta t \to 0$, we get

$$\frac{\partial F(x, y, z, t)}{\partial t} = 0$$

The implicit equation of the envelope surface in three dimensions is obtained by eliminating t from the these two equations.

If F is algebraic we can use symbolic methods to get the equation above.

The path of a point $\mathbf{p}$ belonging to the impict object can be tested for collisions by expressing the sweep volume as

$$F(x, y, z, t) = R(t)F(x, y, z) + T(t)$$

where R is a rotation and T is a translation.

17.2 Dimensionality Paradigm

When we pose no dimension restrictions on the formulation of a geometric problem we get easier ways to rewrite geometric constraints. We have seen in chapter 3.4 that every parametric surface in $\mathbb{R}^3$ can be easily defined implicitly as a surface embedded in $\mathbb{R}^5$.

The expression in the title of this section was coined by Chris Hoffmann in order to describe a systematic approach to solve geometric problems with constraints by embedding the problem in some higher-dimensional space, that is, formulate and equivalent problem in higher-dimensional space, using more variables and more equations.

When restating the problem in higher dimensions we get a new problem with more variables and more equations, but the equations are simpler. In order to put the dimensionality paradigm to work, we need to devise algorithms for implicit surface interrogations in higher dimensions.

The Purdue group has obtained several algorithms that solve some interrogation problems that are very important in the solution of n-dimensional implicit problems. Among these algorithms we could mention:

- Evaluate the intersection of two implicit surfaces in higher-dimensional spaces;
- Evaluate the curvature of an implicit surface at a given point;
- Obtain global approximations to an implicit surface in higher dimensions;

The reader should consult [HV89] and [Hof90] for more details and references.

18. Conclusions

This chapter summarizes the main characteristics of implicit objects and gives some pointers to future research in the area.

18.1 Review

The intrinsic properties of the implicit formulation make it the best model to describe the spatial attributes of objects.

The characteristic function F associated with an implicitly defined object divides space into regions — inside, boundary and outside — representing uniformly both surfaces and solids.

Because the implicit form is defined in terms of a function of space, it can take into account very naturally different levels of detail. This makes possible to establish a connection between abstract geometric descriptions and the physical world, which is an essential requirement in some applications.

The implicit representation has the potential to unify the methods for describing the various classes of real physical objects. The interpretation of the function F as a measure of spatial density gives a definite meaning to models of solids, liquids and gases [Koe89].

Implicit objects constitute the ideal choice for CAD/CAM applications. They fulfill the basic (and apparently incompatible) demands imposed by the three main phases of the industrial production process: design, engineering and manufacturing. At the initial stages of the design, while concepts are still unclear, there is a need for fuzzy sketches. During product development, a precise geometric description is necessary. For production, machine controls must be derived from shape information. Finally, quality control requires that measurements of the real object to be compared with the abstract model. Implicit models provide nice primitives which can be roughed out and refined

incrementally using the same coherent representation. The blend capability of implicit objects plays an important role in this type of application, serving both as an aesthetic feature and as an engineering solution. The distance function intrinsic to implicit objects is well suited to generate offset surfaces, as well as cutting paths sequences for NC machining [1]. Scattered data acquired from the production line can be used to generate an implicit model reconstructing the real object. This makes possible a direct comparison between the final product and its specification.

The implicit form is the appropriate mathematical model for describing volumetric data. This puts implicit objects in a special position, serving as a common thread between computer graphics and many scientific fields. In fact, most of the recent development in modeling with implicit objects is due to scientific visualization applications.

Many complex problems in computer graphics can be formulated implicitly in higher dimensional spaces. Again, implicit objects fit well into this approach, in the sense that they arise naturally as the required solutions, as well as, in the sense that these methods can be incorporated directly into implicit based systems.

18.2 Research Topics

Some topics for future research in the area of implicitly defined objects for computer graphics include:

The study of local and global transformations as a deformation technique for implicit objects.

The development of physically based methods compatible with implicitly defined objects.

The analysis of requirements for a testbed modeling and animation system based on implicit objects that can be used as an environment to implement and test new concepts in the area.

A theoretical investigation of general models for implicit objects, and the role of normal forms in this context.

The applications of implicit models in reconstruction problems, particularly in surface reconstruction from scattered data.

The implications of implicit models for tolerance theory.

[1] numerical controlled machines

dFdMGTV92

[AE98] Ulrike Axen and Herbert Edelsbrunner. Auditory morse analysis of triangulated manifolds. *Mathematical Visualization*, pages 223–236, 1998. Held in Heidelberg.

[AG87] E. L. Allgower and S. Gnutzmann. An algorithm for piecewise linear approximation of implicitly defined two-dimensional surfaces. *SIAM Journal of Numerical Analysis*, 24(2):2452–2469, April 1987.

[AG90] E. L. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. Springer-Verlag, Berlin, Heildelberg, 1990.

[AG91] E. L. Allgower and K. Georg. Simplicial pivotiong for mesh generation of implicitly defined surfaces. *Comp. Aid. Geom. Des.*, 1991.

[AG01] Samir Akkouche and Eric Galin. Adaptive implicit surface polygonization using marching triangles. *Computer Graphics Forum*, 20(2):67–80, 2001. ISSN 1067-7055.

[AK89] James Arvo and David Kirk. A survey of ray tracing acceleration techniques, 1989.

[Alf89] P. Alfeld. Scattered data interpolation in three or more variables. *Mathematical Methods in Computer Aided Geometric Design*, pages 1–34, 1989.

[Arv90] James Arvo. Ray tracing with meta-hierarchies, August 1990.

[AS85] E. L. Allgower and P. H. Schmidt. An algorithm for piecewise linear approximation of an implicitly defined manifold. *SIAM Journal of Numerical Analysis*, 22:322–346, 1985.

[Bar81] A. Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1081.

[Bar83] A. Barr. Decals. ACM Siggraph Course Notes, 1983. State-of-the-Art in Image Synthesis.

[Bar84] A. Barr. Global and local deformations of solid primitives. *Computer Graphics*, 17(3):21–30, 1984.

[Bar86] A. Barr. Ray tracing deformed surfaces. *Computer Graphics*, 20(4):287–296, 1986.

[Bar01] Moacyr Alvim Horta Barbosa. *Medial Axis Reprsentations*. PhD thesis, IMPA, 2001. (in preparation).

[BBB+97] Chandrajit Bajaj, Jim Blinn, Jules Bloomenthal, Marie-Paule Cani-Gascuel, Alyn Rockwood, Brian Wyvill, , and Geoff Wyvil. *Introduction to Implicit Surfaces*. Morgan-Kaufmann, 1997.

[BCX94] Chandrajit L. Bajaj, Jindon Chen, and Goulaing Xu. Free form surface design with a-patches. *Graphics Interface '94*, pages 174–181, May 1994. Held in Banff, Alberta, Canada.

[BCX95a] Chandrajit L. Bajaj, Jindon Chen, and Guoliang Xu. Modeling with cubic a-patches. *ACM Transactions on Graphics*, 14(2):103–133, April 1995. ISSN 0730-0301.

[BCX95b] Chandrit Bajaj, J. Chen, and G. Xu. Interactive shape control and rapid display of a-patches. *Implicit Surfaces '95*, April 1995.

[BdF90] E. Bruzzone and L. de Floriani. Two data structures for building tetrahedralizations. *The Visual Computer*, 6:266–283, 1990.

[Bei90] T. Beier. Practical uses for implicit surfaces in animation. ACM Siggraph Course Notes, 1990. Modeling and Animating with Implicit Surfaces.

[BFGS86] L. Bergman, H. Fuchs, F. Grant, and S. Spach. Image rendering by adaptive refinement. *Computer Graphics*, 20(4):28–38, 1986.

[BH97] Brandon Burch and John Hart. Linear fractal shape interpolation. *Graphics Interface '97*, pages 155–162, May 1997. ISBN 0-9695338-6-1 ISSN 0713-5424.

[BH98] David Bremer and John F. Hughes. Rapid approximate silhouette rendering of implicit surfaces, June 1998.

[BI89] C. L. Bajaj and I. Ihm. Hermite interpolation of rational space curves using real algebraic surfaces. In *Proceedings of 5th Annual Symposium on Computational Geometry*, pages 94–103, 1989.

[BI92] C. L. Bajaj and I. Insung. Smoothing polyhedra using implicit algebraic splines. *Computer Graphics*, 26(2):79–88, 1992.

[Bin57] R. H. Bing. Approximating surfaces with polyhedral ones. *Ann. of Math.*, 1(65):456–483, 1957.

[Bli82] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[Bli84] J. F. Blinn. The algebraic properties of homogeneous second order surfaces, 1984.

[Blo88] J. Bloomenthal. Polygonization of implicit surfaces. *Comp. Aid. Geom. Des.*, 5(4):341–355, 1988.

[Blo90] J. Bloomenthal. Techniques for implicit modeling. *ACM Siggraph Course Notes*, 1990. Modeling and Animating with Implicit Surfaces.

[Blo97] Jules Bloomenthal. Bulge elimination in convolution surfaces. *Computer Graphics Forum*, 16(1):31–41, 1997. ISSN 0167-7055.

[Blu67] H. Blum. A transformation for extracting new descriptors of shape. In W. Whaten-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, MA, 1967.

[BS91] J. Bloomenthal and K. Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–255, 1991.

[BS95] Carole Blanc and Christophe Schlick. Extended fiels functions for soft objects. In *Implicit Surfaces'95*, pages 21–32, Grenoble, France, April 1995. is95.

[Bur83] Peter J. Burt. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540, April 1983.

[BW01a] David E. Breen and R. T. Whitaker. A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):173–192, April - June 2001. ISSN 1077-2626.

[BW01b] Ken Brodlie and Jason Wood. Recent advances in volume visualization. *Computer Graphics Forum*, 20(2):125–148, 2001. ISSN 1067-7055.

[Cai34] S. S. Cairns. On the triangulation of regular loci. *Annals of Math.*, 35:579–587, 1934.

[Cam90] S. A. Cameron. Collision detection by four-dimensional intersection testing. *IEEE Trans. Robotics and Automation*, 6(3):291–302, 1990.

[CBC+01] Jonathan C. Carr, Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. Reconstruction and representation of 3d objects with radial basis functions. *Proceedings of SIGGRAPH 2001*, pages 67–76, August 2001. ISBN 1-58113-292-1.

[CFT88] A. Castelo, S. R. Freitas, and G. Tavares. Simplicial approximation of implicitly defined manifolds, 1988. (unpublished manuscript).

[CG98] Marie-Paule Cani-Gascuel. Layered deformable models with implicit surfaces. *Graphics Interface '98*, pages 201–208, June 1998. ISBN 0-9695338-6-1.

[CGD97] Marie-Paule Cani-Gascuel and Mathieu Desbrun. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):39–50, January - March 1997. ISSN 1077-2626.

[CGMS00] P. Cignoni, F. Ganovelli, C. Montani, and R. Scopigno. Reconstruction of topologically correct and adaptive trilinear isosurfaces, 2000.

[CGV92] P. C. Carvalho, J. M. Gomes, and L. Velho. Space decompositions: Theory and practice. *IMPA (preprint)*, 1992.

[CH89] J. H. Chuang and C. M. Hoffman. On local implicit approximation and its applications. *ACM Transactions on Graphics*, 8(4):298–324, 1989.

[CHF98] Wayne O. Cochran, John C. Hart, and Patrick J. Flynn. On approximating rough curves with fractal functions. *Graphics Interface '98*, pages 65–72, June 1998. ISBN 0-9695338-6-1.

[Chu90] J. H. Chuang. *Surface Approximations in Geometric Modeling*. PhD thesis, Purdue University, 1990.

[CLH01] Wayne O. Cochran, Robert R. Lewis, and John C. Hart. The normal of a fractal surface. *The Visual Computer*, 17(4):209–218, 2001. ISSN 0178-2789.

[Cox63] H. Coxeter. *Regular Polytopes*. Macmillan, New York, 1963.

[CS96] F. Chen. and D. Suter. Multiple order laplacian spline - including splines with tension. Technical report, Monash University - MECSE 1996-5, 1996.

[Dah89] W. Dahmen. Smooth piecewise quadric surfaces. *Mathematical Methods in Computer Aided Geometric Design*, pages 181–194, 1989.

[Dau92] I. Daubechies. *Ten Lectures on Wavelets*. Number 61 in CBMS-NSF Series in Applied Mathematics. SIAM Publications, Philadelphia, 1992.

[dC74] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1974.

[DCG98] Mathieu Desbrun and Marie-Paule Cani-Gascuel. Active implicit surface for animation. *Graphics Interface '98*, pages 143–150, June 1998. ISBN 0-9695338-6-1.

[DCH88] R. A. Debrin, L. Carpenter, and P. Hanrahan. Volume rendering. *Computer Graphics*, 22(4):65–74, 1988.

[dF92] L. H. de Figueiredo. *Computational Morphology of Implicit Curves*. PhD thesis, IMPA - Instituto de Matematica Pura e Aplicada, 1992.

[dFdMGTV92] L. H. de Figueiredo, J. de M. Gomes, D. Terzopoulos, and L. Velho. Physically-based methods for polygonization of implicit surfaces. In *Proceedings of Graphics Interface 92*, 1992.

[DG95] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. *Proceedings of SIGGRAPH 95*, pages 287–290, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

[DLTW90] D. Dobkin, S. Levy, W. Thurston, and A. Wilks. Countour tracing by piecewise linear approximations. *ACM Transactions on Graphics*, 9(4):389–423, 1990.

[Duf92] T. Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. *Computer Graphics*, 26(2):131–138, 1992.

[FPRJ00] Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. *Proceedings of SIGGRAPH 2000*, pages 249–254, July 2000. ISBN 1-58113-208-5.

[Gas93] Marie-Paule Gascuel. An implicit formulation for precise contact modelling between flexible solids. *Computer Graphics*, 27:313–320, 1993.

[GLDH97] M. H. Gross, L. Lippert, R. Dittrich, and S. Häring. Two methods for wavelet-based volume rendering. *Computers & Graphics*, 21(2):237–252, March 1997. ISSN 0097-8493.

[Guo91a] Baining Guo. Shape control in implicit modeling. *Graphics Interface '91*, pages 230–235, June 1991.

[Guo91b] Baining Guo. Surface generation using implicit cubics. *Scientific Visualization of Physical Phenomena (Proceedings of CG International '91)*, pages 485–503, 1991.

[GV98] Jonas Gomes and Luiz Velho. *From Fourier Analysis to Wavelets*. SIGGRAPH'98 Course Notes, SIGGRAPH-ACM publication, Orlando, Florida, july 1998.

[Han88] P. Hanrahan. Survey of ray-object intersections. In A. Glassner, editor, *Introduction to Ray Tracing*. Academic Press, 1988.

[Har89] John C. Hart. Image space algorithms for visualizing quaternion julia sets, 1989.

[Har92] John C. Hart. The object instancing paradigm for linear fractal modeling. *Graphics Interface '92*, pages 224–231, May 1992.

[Har96] John C. Hart. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(9):527–545, 1996. ISSN 0178-2789.

[Har98] Erich Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14(3):95–108, 1998. ISSN 0178-2789.

[HD91] John C. Hart and Thomas A. DeFanti. Efficient anti-aliased rendering of 3d linear fractals. *Computer Graphics (Proceedings of SIGGRAPH 91)*, 25(4):91–100, July 1991. ISBN 0-201-56291-X. Held in Las Vegas, Nevada.

[Hen93] Michael Henderson. Computing implicitly defined surfaces: two paramater continuation. Technical report, IBM - Research Division, 1993.

[HH95] Daryl H. Hepting and John C. Hart. The escape buffer: Efficient computation of escape time for linear fractals. *Graphics Interface '95*, pages 204–214, May 1995. ISBN 0-9695338-4-5.

[HJ99] K. C. Hui and Z. H. Jiang. Tetrahedra based adaptive polygonization of implicit surface patches. *Computer Graphics Forum*, 18(1):57–68, March 1999. ISSN 1067-7055.

[Hof89] C. M. Hoffmann. *Solid and Geometric Modeling: An introduction*. Morgan Kauffmann, 1989.

[Hof90] C. M. Hoffmann. A dimensionality paradigm for surface interrogations. *Computer Aided Geometric Design*, 7:517–532, 1990.

[Hof91] C. M. Hoffmann. Skeletons, cyclographic maps, and shock waves. manuscript, 1991.

[Hof92] C. M. Hoffmann. Computer vision, descriptive geometry, and classical mechanics. In B. Falcidieno and I. rman, editors, *Proc. Eurographics Workshop on Computer Graphics and Mathematics*, Eurographics Series, pages 229–244. Springer Verlag, 1992.

[Hug92] John F. Hughes. Scheduled fourier volume morphing. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):43–46, July 1992. ISBN 0-201-51585-7. Held in Chicago, Illinois.

[HV89] C. M. Hoffmann and George Vanecek. Fundamental techniques for geometric and solid modeling. Technical report, Purdue University, 1989.

[HW90] M. Hall and J. Warren. Adaptive polygonization of implicitly defined surfaces. *IEEE Computer Graphics and Applications*, 10(6):33–43, 1990.

[IP98] Insung Ihm and Sanghun Park. Wavelet-based 3d compression scheme for very large volume data. *Graphics Interface '98*, pages 107–116, June 1998. ISBN 0-9695338-6-1.

[IPZ79] S. Incerti, V. Parisi, and F. Zirilli. A new method for solving nonlinear simultaneous equations. *SIAM Journal on Numerical Analysis*, 16:779–789, 1979.

[Joh83] F. E. A. Johnson. On the triangulation of stratified sets and singular varieties. *Transactions of the American Mathematical Society*, 275(1), 1983.

[Kau87] Arie Kaufman. Efficient algorithms for 3d scan-conversion of parametric curves, surfaces, and volumes. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):171–179, July 1987.

[KBSS01] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature-sensitive surface extraction from volume data. *Proceedings of SIGGRAPH 2001*, pages 57–66, August 2001. ISBN 1-58113-292-1.

[Kle89] J. Kleck. Modeling using implicit surfaces. Master's thesis, University of California at Santa Cruz, 1989.

[Koe89] K. Koendrick. *Solid Shape*. MIT Press, 1989.

[KS01] T. Karkanis and A. J. Stewart. Curvature-dependent triangulation of implicit surfaces. *IEEE Computer Graphics & Applications*, 21(2):60–69, March / April 2001. ISSN 0272-1716.

[Las85] D. Lasser. Bernstein-Bezier representation of volumes. *Comp. Aid. Geom. Des.*, 2:145–149, 1985.

[LC87] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[Lev90] M. Levoy. Efficient ray tracing of volume rendering. *ACM Transactions on Graphics*, 9(3):245–261, 1990.

[LG95] L. Lippert and M. H. Gross. Fast wavelet based volume rendering by accumulation of transparent texture maps. *Computer Graphics Forum*, 14(3):431–444, August 1995. ISSN 1067-7055.

[LGL95] Apostolos Lerios, Chase D. Garfinkle, and Marc Levoy. Feature-based volume metamorphosis. *Computer Graphics*, 29(Annual Conference Series):449–456, 1995.

[Mar82] D. Marr. *Vision*. Freemann, 1982.

[MAS91] 0. Monga, N. Ayache, and P. Sander. From voxel to curvature. In *Proc. IEEE Computer Vision and Pattern Recognition*, 1991.

[Mat68] Mathematical Applications Group Inc. 3D simulated graphics offered by service bureau. *Datamation*, 13(1):69, 1968.

[MC92] D. Manocha and J. F. Canny. Algorithm for implicitizing rational parametric surfaces. *Computer Aided Geometric Design*, 9(1):25–51, 1992.

[Mil65] J. Milnor. *Topology from the Differentiable Viewpoint*. The University Press of Virginia, 1965.

[Mil89] J. R. Miller. Architecture issues in solid modelers. *IEEE Computer Graphics and Applications*, 9(5):72–87, 1989.

[Moo79] R. Moore. *Methods and Applications of Interval Analysis*. SIAM - Society for Industrial and Applied Mathematics, 1979.

[MS90] R. R. Martin and P. C. Stephenson. Sweeping of three dimensional objects. *Computer Aided Design*, 22(4):223–234, 1990.

[MS98] Jon McCormack and Andrei Sherstyuk. Creating and rendering convolution surfaces. *Computer Graphics Forum*, 17(2):113–120, 1998. ISSN 1067-7055.

[MSS94] C. Montani, R. Scateni, and R. Scopigno. A modified look-up table for implicit disambiguation of marching cubes, 1994.

[MTV86] Y. De Montaudoin, W. Tiller, and H. Vold. Applications of power series in computational geometry. *Comp. Aid. Geom. Des.*, 10(18), 1986.

[Mun66] J. Munkres. *Elementary Differential Topology*. Princeton University Press, 1966.

[Mur93] S. Muraki. Volume data and wavelet transform. *IEEE Computer Graphics and Applications*, 13(4):50–56, July 1993.

[MW91] Doug Moore and Joe Warren. Bounded aspect ratio triangulation of smooth solids. *SMA '91: Proceedings of the First Symposium on Solid Modeling Foundations and CAD/CAM Applications*, pages 455–464, June 1991. ISBN 0-89791-427-9. Held in held June 5-7, 1991 in Austin, Texas, USA. .

[MZ93] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. Technical report, Courant Institute, New York University, 1993.

[Nat94] B. Natarajan. On generating topologically consistent isosurfaces from uniform samples, 1994.

[Nav89] I. Navazo. Extended octree representation of general solids with plane faces: Model structure and algorithms. *Computer & Graphics*, 13(1):5–16, 1989.

[NHK+85] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Object modeling by distribution function and a method of image generation. *Japan Electronics Communication Conference 85*, J68-D(4):718–725, 1985.

[Nor82] A. Norton. Generation and display of geometric fractals in 3D. *Computer Graphics*, 16(3):61–66, 1982.

[NP85] L. R. Nackman and S. M. Pizer. Three-dimensional shape description using the symmetric axis transform I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI 7:187–205, 1985.

[PASS95] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995. ISSN 0178-2789.

[Pav78] T. Pavlidis. Filling algorithms for raster graphics. *Computer Graphics (SIGGRAPH '78 Proceedings)*, 12(3):161–166, August 1978.

[Pea85] Darwyn R. Peachey. Solid texturing of complex surfaces. *Computer Graphics (Proceedings of SIGGRAPH 85)*, 19(3):279–286, July 1985. Held in San Francisco, California.

[Ped95] Hans Køhling Pedersen. Decorating implicit surfaces. *Proceedings of SIGGRAPH 95*, pages 291–300, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

[Ped96] Hans Køhling Pedersen. A framework for interactive texturing operations on curved surfaces. *Proceedings of SIGGRAPH 96*, pages 295–302, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[Per85a] K. Perlin. An image synthesizer. *Computer Graphics*, 19(3):287–293, 1985.

[Per85b] Ken Perlin. An image synthesizer. *Computer Graphics (Proceedings of SIGGRAPH 85)*, 19(3):287–296, July 1985. Held in San Francisco, California.

[PH89] K. Perlin and E. Hoffert. Hypertexture. *Computer Graphics*, 23(3), 1989.

[PP88] A. Pasko and V. Pilyugin. Geometric modeling in the analysis of trivariate functions. *Computer and Graphics*, 12(3-4):457–465, 1988.

[PS95] Alexander Pasko and Vladimir Savchenko. Constructing functionnally-defined surfaces. *Implicit Surfaces '95*, April 1995.

[PW89] A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics*, 23(3):215–222, 1989.

[PZ90] K. Perlin and B. Zhu. Surflets. ACM Siggraph Course Notes, 1990. Photorealistic Volume Modeling and Rendering Techniques.

[Req80] A. Requicha. Representation for rigid solids: Theory, methods, and systems. *ACM Computing Surveys*, 12(4), 1980.

[Rhe87] W. C. Rheinboldt. On a moving frame algorithm and the triangulation of equilibrium manifolds. In T. Kupper, R. Seydel, and H. Trogger, editors, *Bifurcation: Analysis, Algorithms, Applications*, pages 256–267. Birkhauser-Verlag, 1987.

[Ric73] A. Ricci. A constructive solid geometry for computer graphics. *The Computer Journal*, 16(2):157–160, 1973.

[RO87] A. P. Rockwood and J. Owen. Using implicit surfaces to blend arbitrary solid models. In G. Farin, editor, *Geometric Modeling: Algorithms and Trends*. SIAM, 1987.

[RO89] J. Rossignac and M. O'Connor. SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries. Technical report, IBM Research Division, 1989.

[Roc89] A. P. Rockwood. The displacement method for implict blending surfaces in solid models. *ACM Transactions on Graphics*, 8(4):279–297, 1989.

[Rot82] S. Roth. Ray casting as a method for solid modeling. *Computer Graphics and Image Processing*, 18(2):109–144, 1982.

[SA84] T. Sederberg and D. Anderson. Implicit representation of parametric curves and surfaces, 1984.

[Sed90a] T. W. Sederberg. Techniques for cubic algebraic surfaces, part 1. *IEEE Computer Graphics and Applications*, 10(4):14–25, 1990.

[Sed90b] T. W. Sederberg. Techniques for cubic algebraic surfaces, part 2. *IEEE Computer Graphics and Applications*, 10(5):12–21, 1990.

[SH97] Barton T. Stander and John C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. *Proceedings of SIGGRAPH 97*, pages 279–286, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California.

[Sha88] Vadim Shapiro. Theory of r-functions and applications: A primer. Technical report, Cornell University, November 1988.

[Sha91] V. Shapiro. Representations of semi-algebraic sets in finite algebras generated by space decompositions. Technical report, Cornell University, 1991.

[She99] Andrei Sherstyuk. Kernel functions in convolution surfaces: a comparative analysis. *The Visual Computer*, 15(4):171–182, 1999. ISSN 0178-2789.

[Sni92] J. M. Snider. Interval analysis for computer graphics. *Computer Graphics*, 26(2):121–130, 1992.

[SP86] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. *Computer Graphics*, 20(4):151–160, 1986.

[SP91] S. Sclaroff and A. Pentland. Generalized implicit functions for computer graphics. *Computer Graphics*, 25(4):251–256, 1991.

[Spi65] M. Spivak. *Calculus on Manifolds*. W. A. Benjamin, 1965.

[ST99] Vadim Shapiro and Igor Tsukanov. Implicit functions with guaranteed differential properties. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, 1999.

[SW97] G. Sealy and G. Wyvill. Representing and rendering sweep objects using volume models. *Computer Graphics International 1997*, June 1997. Held in Hasselt/Diepenbeek, Belgium.

[SZ89] T. W. Sederberg and A. K. Zundel. Scan line display of algebraic surfaces. *Computer Graphics*, 23(3):147–156, 1989.

[TdMG89] G. Tavares and J. de M. Gomes. Concordance operations for implicitly-defined manifolds. In *Proceedings of SIAM Conference on Geometric Design*, 1989.

[TDOY01] Greg Turk, Huong Quynh Dinh, James O'Brien, and Gary Yngve. Implicit surfaces that interpolate. *Shape Modelling International*, pages 62–71, May 2001.

[Tei98] Ralph Costa Teixeira. *Curvature Motions, Medial Axes and Distance Transforms*. PhD thesis, Harvard University, Cambridge, Massachusetts, June 1998.

[Ter01] Bryan Morse Terry. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions, 2001.

[TL93] Takashi Totsuka and Marc Levoy. Frequency domain volume rendering. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 271–278, August 1993.

[TO99] Greg Turk and James O'Brien. Shape transformation using variational implicit functions. *Proceedings of SIGGRAPH 99*, pages 335–342, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

[TPF89a] D. Terzopoulos, J. Platt, and K. Fleischer. Heating and melting deformable objects. In *Proceedings of Graphics Interface 89*, pages 219–226, 1989.

[TPF89b] Demetri Terzopoulos, John Platt, and Kurt Fleischer. Heating and melting deformable models (from goop to glop). *Graphics Interface '89*, pages 219–226, June 1989.

[TSYK01] S. Tanaka, A. Shibata, H. Yamamoto, and H. Kotsuru. Generalized stochastic sampling method for visualization and investigation of implicit surfaces. *Computer Graphics Forum*, 20(3), 2001.

[TW99] Mark Tigges and Brian Wyvill. A field interpolated texture mapping algorithm for skeletal implicit surfaces. *Computer Graphics International '99*, June 1999. ISBN ISBN 0-7695-0185-0.

[TWM85] J.F. Thompson, Z. U. Z. Waisi, and C. W. Mastin. *Numerical Grid Generation, Foundations and Applications*. North Holland, New York, 1985.

[VdFG99] Luis Velho, Luiz Henrique de Figueiredo, and Jonas Gomes. A unified approach for hierarchical adaptive tesselation of surfaces. *ACM Transactions on Graphics*, 18(4):329–360, October 1999. ISSN 0730-0301.

[VdMG91a] L. Velho and J. de M. Gomes. A dynamical simulation environment for implicit objects using discrete models. In *Proceedings of 2nd Eurographics Workshop on Animation and Simulation*, 1991.

[VdMG91b] L. Velho and J. de M. Gomes. Regular triangulations of implicit manifolds using dynamics. In *Proceedings of Compugraphics 91*, 1991.

[Vel90a] L. Velho. Adaptive polygonization of implicit surfaces using simplicial decomposition and boundary constraints. In *Proceedings of Eurographics 90*. Elsevier Science Publisher, 1990.

[Vel90b] L. Velho. Interactive modeling of soft objects. In *Proceedings of Ausgraph 90*, 1990.

[Vel96] Luiz Velho. Simple and efficient polygonization of implicit surfaces. *Journal of Graphics Tools*, 1(2):5–24, 1996. ISSN 1086-7651.

[VG96] Luiz Velho and Jonas Gomes. Approximate conversion of parametric to implicit surfaces. *Computer Graphics Forum*, 15(5):327–338, 1996. Elsevier Science Publishers.

[vHB87] B. von Herzen and A. Barr. Accurate triangulations of deformed intersecting surfaces. *Computer Graphics*, 21(4):103–110, 1987.

[VTG94] Luiz Velho, Demetri Terzopoulos, and Jonas Gomes. Multiscale implicit models. In *Proceedings of SIBGRAPI '94*, pages 93–100, Curitiba, Novembro 1994.

[Wal75] C. T. C. Wall. Regular stratifications. *Lecture Notes on Mathematics*, 468:332–344, 1975.

[Wei86] K. J. Weiller. *Topological Structures for Geometric Modeling*. PhD thesis, Rensselaer Polytechnic Institute, 1986.

[WG91] J. Wilhelms and A. Van Gelder. A coherent projection approach for direct volume rendering. *Computer Graphics*, 25(4):275–284, 1991.

[WGG99] Brian Wyvill, Andrew Guy, and Eric Galin. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum*, 18(2):149–158, June 1999. ISSN 1067-7055.

[WH94] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. *Proceedings of SIGGRAPH 94*, pages 269–278, July 1994. ISBN 0-89791-667-0. Held in Orlando, Florida.

[Whi57] H. Whitney. Elementary structure of real algebraic varietes. *Annals of Math.*, 66(3):545–556, 1957.

[Wis90] P. Wisskirchen. Object-oriented graphics. *ACM Siggraph Course Notes*, 1990.

[WK85] Geoff Wyvill and Tosiyasu L. Kunii. A functional model for constructive solid geometry. *The Visual Computer*, 1(1):3–14, July 1985.

[WMW86a] B. Wyvill, C. McPheeters, and G. Wyvill. Animating soft objects. *The Visual Computer*, 2(4):235–242, 1986.

[WMW86b] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

[WMW87] G. Wyvill, C. McPheeters, and B. Wyvill. Solid texture of soft objects. *IEEE Computer Graphics and Applications*, 7(12):20–26, 1987.

[WW89] Brian Wyvill and Geoff Wyvill. Field functions for implicit surfaces. *The Visual Computer*, 5(1/2):75–82, March 1989.

[Wyv90] B. Wyvill. Metamorphosis of implicit surfaces. *ACM Siggraph Course Notes*, 1990. Modeling and Animating with Implicit Surfaces.

[XHB01] Guoliang Xu, Hongci Huang, and Chandrajit Bajaj. C1 modeling with a-patches from rational trivariate functions. *Computer Aided Geometric Design*, 18(3):221–243, April 2001. ISSN 0167-8396.

[ZGV+98] Ruben Zonenschein, Jonas Gomes, Luiz Velho, Luiz Henrique de Figueiredo, Mark Tigges, and Brian Wyvill. Texturing composite deformable implicit objects. In *Proceedings of SIBGRAPI 98*, pages 346–353. SBC - Sociedade Brasileira de Computacao, IEEE Press, October 1998.

[ZGVdF98] Ruben Zonenschein, Jonas Gomes, Luiz Velho, and Luiz Henrique de Figueiredo. Controlling texture mapping onto implicit surfaces with particle systems. In Jules Bloomenthal and Dietmar Saupe, editors, *Proceedings of the Third International Workshop on Implicit Surfaces*, pages 131–138, Seattle, USA, July 1998. Eurographics - ACM SIGGRAPH, Eurographics.

[ZJ91] C. Zahlten and H. Jurgens. Continuation methods for approximating isovalued complex surfaces. In *Proceedings of Eurographics 91*, pages 5–19, 1991.

This page intentionally left blank

Index