TM–54

BLENDING AND OFFSETTING SOLID MODELS

by

Jarosław R. Rossignac

PRODUCTION AUTOMATION PROJECT

July 1985

Technical Memorandum No. 54

PRODUCTION AUTOMATION PROJECT
Department of Electrical Engineering
College of Engineering and Applied Science
University of Rochester
Rochester, New York 14627

TM–54

BLENDING AND OFFSETTING SOLID MODELS

by

Jarosław R. Rossignac

Submitted in Partial Fulfillment
of the
Requirements for the Degree

DOCTOR OF PHILOSOPHY

Supervised by: Dr. Aristides A. G. Requicha

July 1985

The work reported in this paper was supported by the National Science Foundation under Grant ECS–81–04646, and by companies in the Production Automation Project's Industrial Associates Program. Any opinions, findings, conclusions or recommendations expressed or implied are the author's and do not necessarily reflect the views of the N.S.F. or the Industrial Associates of the P.A.P.

ABSTRACT

A new generation of systems for modelling three–dimensional parts and assemblies is emerging. These systems are called "solid modellers" and carry complete (i.e., unambiguous) representations of the modelled solids, from which the various properties of solids that are important in CAD/CAM can be computed automatically. The geometric domain of current modellers is limited by the types of surfaces they support, which usually are quadric and toroidal. Part surveys show that the large majority of functional (non–sculptured) mechanical parts could be represented in such modellers if facilities were provided for the representation of fillets and roundings, collectively called *blends*.

This thesis studies the semantics of blending operations and the geometric properties of blending surfaces, and develops a theory and algorithms that allow the harmonious integration of blending facilities in dual–representation solid modellers containing both Constructive Solid Geometry (CSG) and Boundary (Brep) representations.

Entire solids or sub–solids can be globally blended with constant radius blends through offsetting operations (shrinking and expanding). Offsetting is used as the primary tool for specifying and representing blends in an *extended* CSG scheme, but also has many important applications in other areas of Computer Aided Design and Manufacturing. Offset or blended solids are bounded by the standard surfaces supported by current modelers and by canal surfaces, which are envelopes of families of spherical surfaces.

To accommodate solids bounded by canal surfaces and use them like other solids in a dual–representation modeller, one must be able to perform set membership classification operations on such solids. The algorithms described in this thesis are based on explicit representations of canal surfaces and of their intersection–edges. To reduce the cost of computing and processing such representations, all edges are approximated by smooth piecewise–circular curves (PCC), and all canal surfaces by smoothly connected pieces of tori or cylinders. PCC approximations are sufficiently smooth for practical applications and very well suited for the required computations.

An experimental modeller was implemented to demonstrate that the proposed approach is viable.

VITA

Jarosław Roman Rossignac was born on June 26, 1955 in Warsaw, Poland and emigrated with his parents to France in 1965. He obtained a French "Baccalauréat" degree in Mathematics and Physics with Honors in 1974. He received the first part of the French Engineer's education at the Lycée Lakanal, Sceaux, studying advanced Mathematics and Physics. In 1976 he was admitted, through a national competitive examination to the "Ecole Nationale Supérieure d'Electrici-té et de Mé-ca-ni-que" in Nancy, France, where he received a standard education in Electrical and Mechanical Engineering. During the last year, he specialized in Systems and Computer Science and graduated *Summa Cum Laude* in 1979. That very year he came to the University of Rochester as a graduate student in the Department of Electrical Engineering. In August 1980, he was drafted in the French Navy for sixteen months and worked as a software engineer developing real–time systems. He resumed his doctoral work at the University of Rochester as a research assistant with the Production Automation Project. In 1982, he obtained his Master's Degree in Electrical Engineering, and fulfilled his teaching and courses requirements for the degree of Doctor of Philosophy.

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to Professors Herbert B. Voelcker and Aristides A. G. Requicha who provided excellent advice on the matter of this thesis and other professional concerns. Their dedication to science and their high standards of professional excellence were a constant source of encouragement and inspiration. Professor Requicha, as the author's advisor for this research, should be specially thanked for his patience in repeatedly proof–reading this dissertation and for his advice and help in tuning the finest details. The author's thanks go also to Professors Alexander Albicki and Christopher Brown, who provided help and showed great interest in the progress of this research, to Doctor Robert Tilove, who is responsible for the author's interest in the blending problem, and to the staff and students of the Production Automation Project for their assistance and continual encouragement. And finally, the author would like to thank his parents for their concern and support and Catherine for her dedication in proof–reading this manuscript and in drawing many of the figures.

NOTATION

$\overline{S}$	complement of the set S
$A \bigcup B$	Boolean union of the two sets A and B
$A \bigcap B$	Boolean intersection of the two sets A and B
$A - B$	Boolean difference of the two sets A and B: $A - B = A \bigcap \overline{B}$
$iS, kS, \partial S$	interior, closure, and boundary of the set S: $\partial S = kS \bigcap k\overline{S}$
$\overline{S}^*$	regularized complement of the set S: $\overline{S}^* = ki\overline{S}$
$A \bigcup^* B$	regularized Boolean union: $A \bigcup^* B = ki(A \bigcup B)$
$A \bigcap^* B$	regularized Boolean intersection: $A \bigcap^* B = ki(A \bigcap B)$
$A -^* B$	regularized Boolean difference: $A -^* B = ki(A - B)$
$S\uparrow r$	S expanded by r: $S\uparrow r = \{P : \exists Q \in S, \|P - Q\| \leq r\}$
$S\downarrow r$, $S\downarrow^* r$	shrinking: $S\downarrow r = \overline{\overline{S}\uparrow r}$, regularized shrinking: $S\downarrow^* r = \overline{\overline{S}^*\uparrow r}^*$
$d(P,S)$	distance from point P to set S: $d(P,S) = \inf_{Q \in S} \|P - Q\|$
$R_r(S)$	S rounded by r: $R_r(S) = (S\downarrow r)\uparrow r = \bigcup \{B(Q,r) \subset S\}$
$F_r(S)$	S filleted by r: $F_r(S) = \overline{R_r(\overline{S})} = (S\uparrow r)\downarrow r$
$F'_r(S)$	modified filleting: $F'_r(S) = \overline{R_r(\overline{S}^*)}^* = k\big(F_r(iS)\big)$
$R^*_r(S)$	regularized rounding: $R^*_r(S) = (S\downarrow^* r)\uparrow r$
$F^*_r(S)$	regularized filleting: $F^*_r(S) = \overline{R^*_r(\overline{S}^*)}^* = (S\uparrow r)\downarrow^* r$
$\widehat{R}(S)$	radius of maximum convexity of S: $\widehat{R}(S) = \sup_{S = R_r(S)} r$
$\breve{R}(S)$	radius of maximum concavity of S: $\breve{R}(S) = \sup_{S = F_r(S)} r$
$B_r(S)$	constant distance set $B_r(S) = \{P : d(P,S) = r\}$
$F\|_r$	parallel faces, n-offset by r from the face F
$\{P \rightarrowtail S\}$	closest projection set of point P on the set S

TABLE OF CONTENTS

CHAPTER IV
BOUNDARIES OF OFFSET SOLIDS

CHAPTER V
REPRESENTATIONAL REQUIREMENTS

CHAPTER VI
COMPUTATION IN CSGO

CHAPTER VII
APPROXIMATION OF EDGES

CHAPTER VIII
PIECEWISE CIRCULAR CURVES

CHAPTER IX
COMPUTATION WITH PCC'S

CHAPTER X
EXPERIMENTAL IMPLEMENTATION

CHAPTER XI
CONCLUSION

APPENDIX A
CANAL SURFACES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

This thesis reports research on two closely related operations, called *blending* and *offsetting*, which map solids onto solids. Blending is a catch–all term for the creation of fillets, rounds, and similar smooth, localized transitions between large–scale surface features of a solid object. Figure 1.1 shows that blending can be viewed as an addition or a subtraction of material. Offsetting amounts to growing or shrinking solids (Figure 1.2) and can be used to produce globally blended solids, as we shall see later.

1.1 BLENDING

The major types of blends encountered in practice can be categorized as follows:

- Surfaces governed by *strong functional constraints.* For example, the surface joining the wing of an aircraft to the fuselage must meet stringent aerodynamic requirements.
- *Aesthetic blends.* For example, the smooth transitional surface between the body and the stem of a wine glass is constrained by appearance more than by function.
- *Fairings* [Veenman 82] are transitional surfaces that are relatively large when compared to the surface features being blended. Their shape is not strongly constrained by either function or aesthetics. Typically, fairings connect functional features such as bearing housings. Many examples are found in automobile transmission or suspension parts, ducts and manifolds.

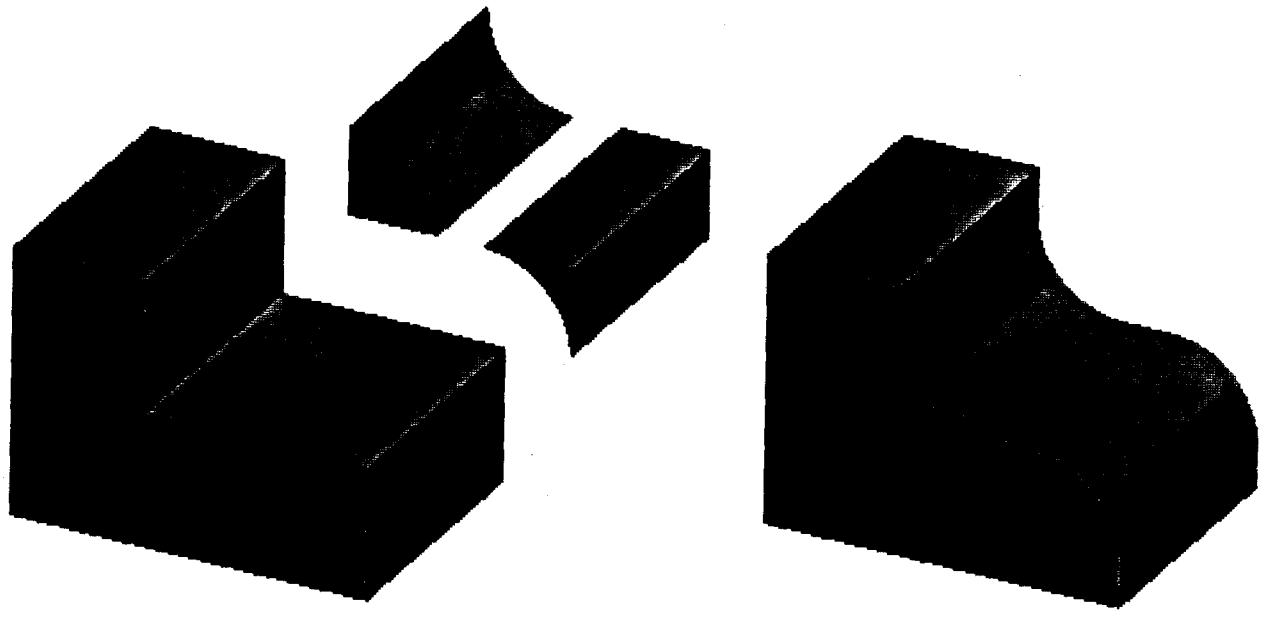

FIGURE 1.1: To fillet the concave edge of the solid S (left) we union to S a solid fillet (center). A convex edge can be rounded by subtracting a similar fillet from S (right).

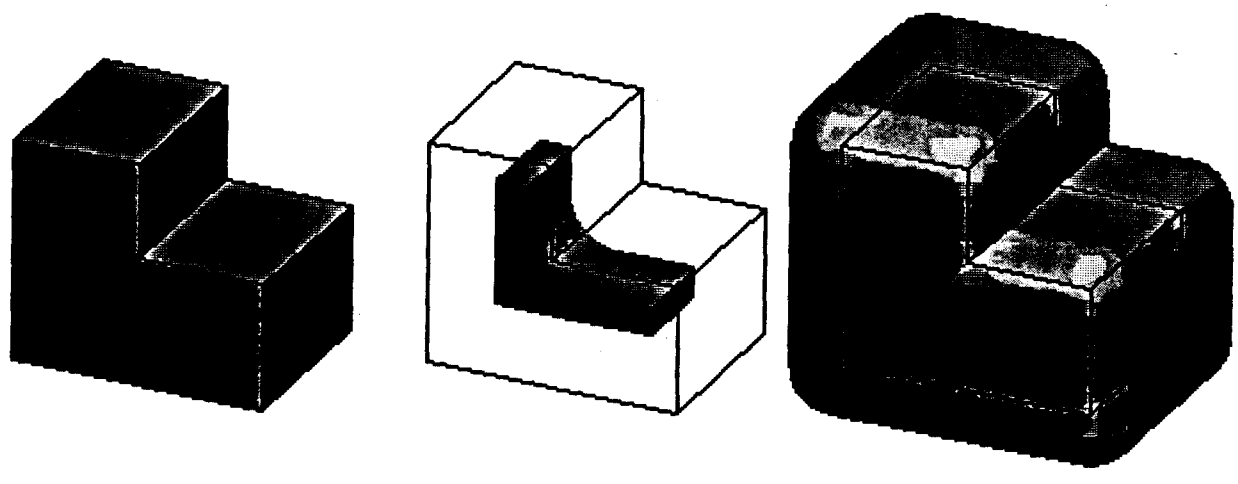

FIGURE 1.2: The solid (left) is shrunk (center) or expanded (right).

- *Rounds and fillets* are relatively small transitional faces found in most machined, cast or molded parts. They serve to relieve stress concentration, to simplify fabrication, or simply to improve appearance. Their shape is weakly constrained by function.

The shapes of fairings and of fillets are not crucial. Designers often specify only a few cross sections of a fairing or a maximum radius for a fillet; the exact shapes are left to the discretion of manufacturing engineers, who are expected to produce smooth blends from incomplete specifications. In contrast, blends that are aesthetic or that have strong functional constraints require precise control of shape.

Traditionally, blends have been specified in blueprints, but manual drafting is rapidly giving way to computerized geometric modelling systems, which contain complete, unambiguous representations for solids [Requicha 82, 83]. These representations permit (at least in principle) any well–defined geometric property of any represented solid to be calculated automatically.

Sculptured surfaces [Barnhill 74, Faux 79] can be used to model blends, but this technology is computationally expensive and numerically less robust than quadric–surface technology. Modelling sculptured objects is undoubtedly important for certain applications [Sarraga 83] and solid modellers that can accommodate them are beginning to emerge [Tiller 83], but the graceful meshing of sculptured surface and solid modelling technologies is still largely at the research stage. On the other hand, in many applications the objects are primarily unsculptured. For example, a survey of about 100 parts of a Xerox copier showed that only one of the parts contained a sculptured surface that did not have a blending function [Samuel 76].

Our work explores alternative methods for dealing with rounds and fillets in otherwise unsculptured, or functional parts. More specifically, we focus on constant–radius rounds and fillets (hereafter called simply *blends*) because they are the most common. Faces of constant–radius blends are generated (conceptually) by a sphere that rolls tangentially to the surfaces

being blended. The face of the blend is included in the surface that bounds the region swept by the rolling sphere (Figure 1.3).

Most modellers accommodate only solids bounded by standard surfaces (simple quadrics and, sometimes, toroidal surfaces) [Requicha 82, 83]. Surveys of parts indicate that such modellers would handle almost all of the unsculptured mechanical parts if they could support blends. Some fillets can be represented by a combination of cylinders, spheres and tori (Figure 1.4), but most require non–standard surfaces and imply true extensions of the modellers' domains. For example, the filleted cylinder/cylinder intersection of Figure 1.5 cannot be represented in a modern modeller such as PADL–2 [Brown 82]. The inability of current modellers to represent all the important details of mechanical parts is one of the major impediments to the widespread use of solid modelling in industry.

The design of blending facilities for solid modellers is strongly influenced by modellers' architectures. Most of the solid modellers use one of the following two basic architectures. The single–representation systems diagrammed in Figure 1.6 allow users to define solids by various techniques but store and maintain only a boundary representation (BRep) for each object. (A BRep is a graph whose nodes represent faces, edges and vertices of a solid [Requicha 80].) Users of the dual–representation systems of Figure 1.7 define objects primarily through constructive solid geometry (CSG) but these systems also use a BRep, which is derived algorithmically from the corresponding CSG. (A CSG representation is a tree whose internal nodes represent rigid motions or Boolean operators — regularized union, intersection and difference, denoted $\cup^*$, $\cap^*$, $-^*$ [Requicha 80] — and whose leaves represent primitive solids such as cuboids and cylinders.) All application programs (for graphics, mass property calculations, etc.) are supported from BReps in the single representation modellers, while dual–representation systems may support application algorithms that operate on either BReps or CSG, or both.

We are primarily interested in designing and implementing blending facilities for dual–representation modellers such as PADL–2. We must maintain consistency between BRep and CSG, and since algorithms for converting BRep into CSG are not available, we must seek CSG–like specifications and representations for blended objects, and algorithms to compute their boundaries [Requicha 85]. The basic geometric utilities (e.g., set

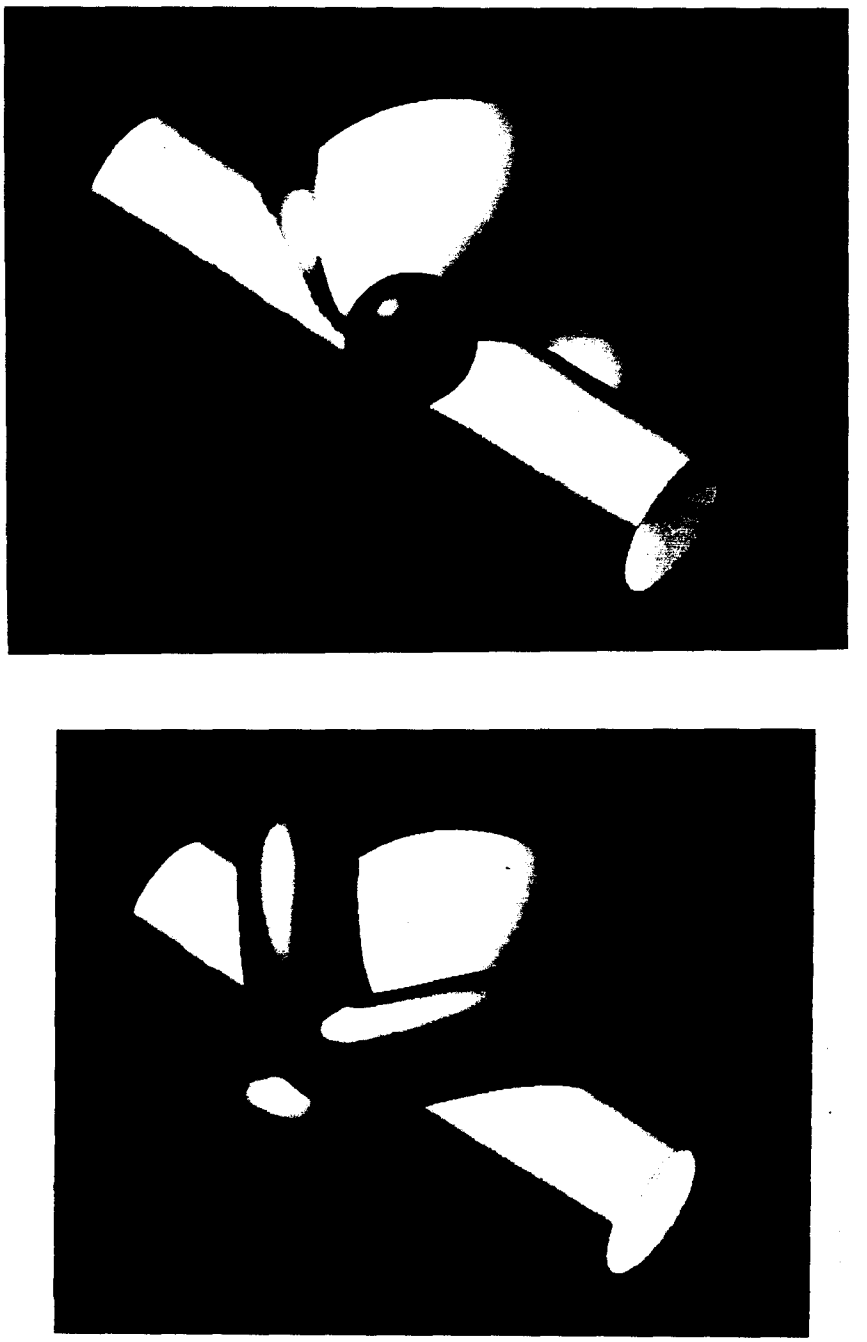

FIGURE 1.3: The sphere of specified radius rolls along the blended edge, remaining in contact with the blended faces (top). The face of the blend is contained in the envelope of the region swept by the sphere (bottom).

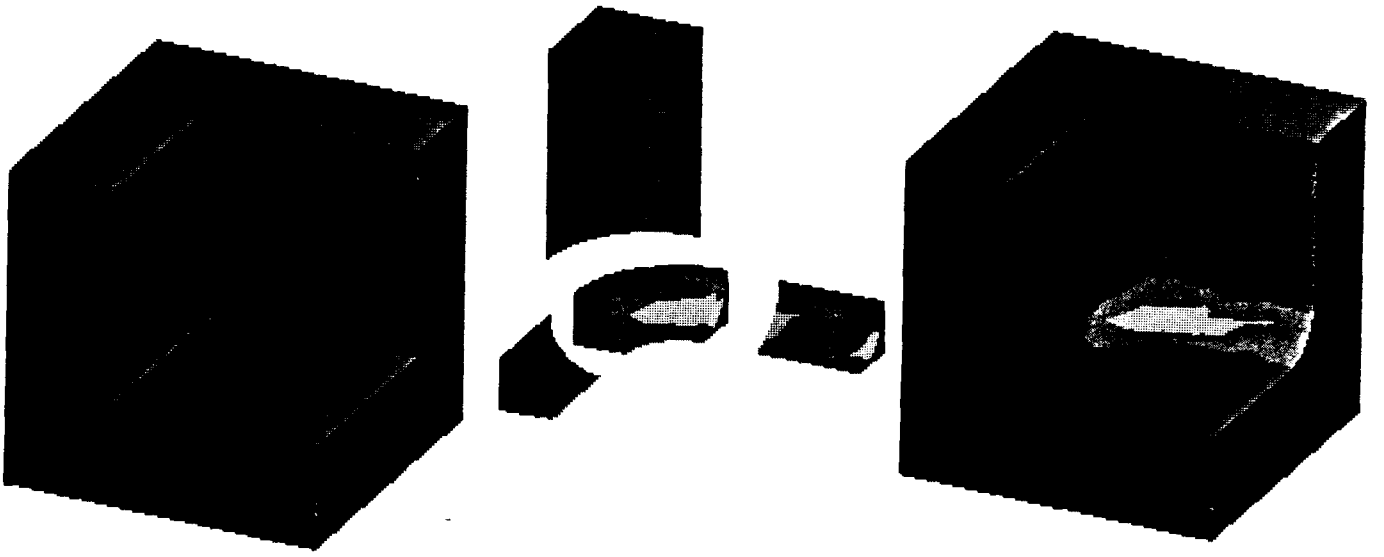

FIGURE 1.4 : The concave edges of the solid (left) can be blended (right) with a combination of fillets defined in CSG (center).

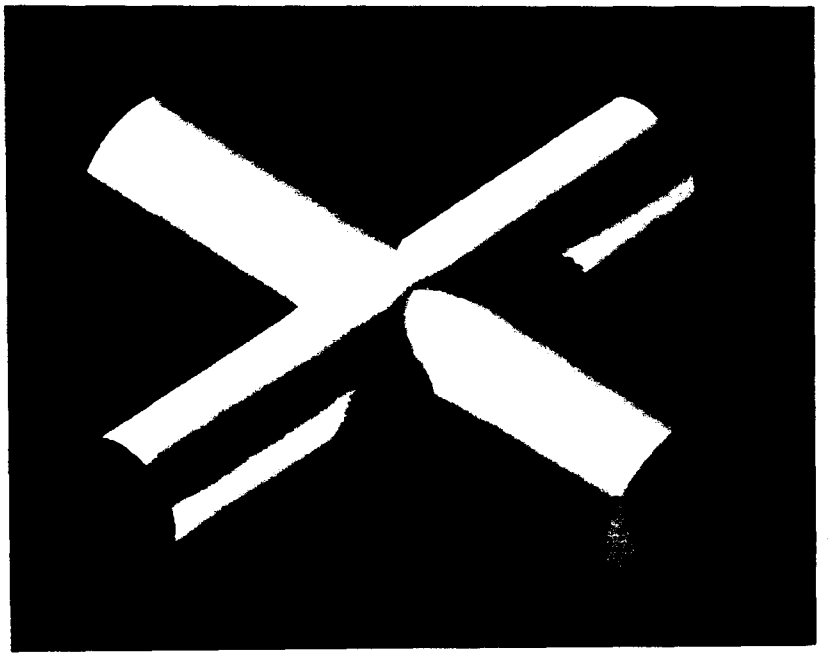

FIGURE 1.5 : The concave edge of the union of two cylinders is blended with a constant radius fillet. The result cannot be modelled with standard surfaces.

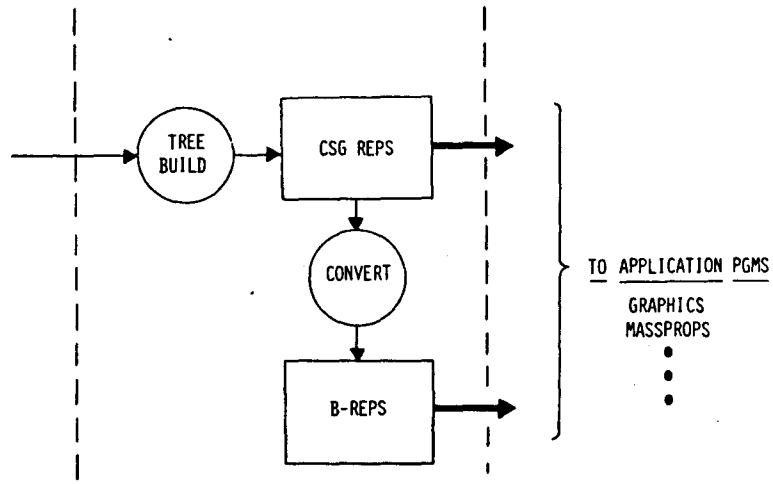

FIGURE 1.6: Single–scheme BRep modeller.

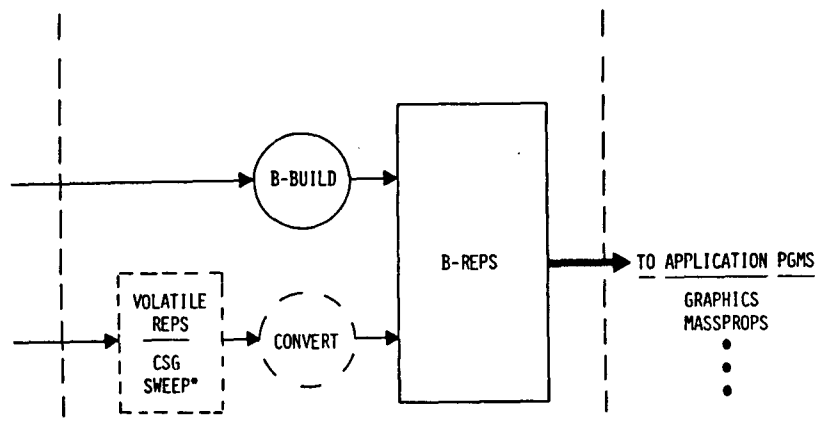

FIGURE 1.7: Dual–scheme CSG/BRep modeller. Both representations are consistent and available for application programs.

membership classification [Tilove 80]) needed by application algorithms must also be extended to accommodate blended objects. Provision of blending capabilities in modellers raises issues of user specification, internal representation, and computation. Specification must be simple; users should have to do little more than define the feature to be blended (edge, pocket...) and a radius. This is essentially how blends have been traditionally specified in blueprints. User specifications should correspond to mathematically defined shape modifications, which must be internally represented in a modeller in a form suitable for the operations and the applications it supports. It should be possible to perform for blended objects all the computations possible for unblended objects. Thus, one should be able to test blended objects for interference, to evaluate their volumes and moments of inertia, to display them in various styles, and so on. These requirements raise complex computational problems, as we shall see.

Our approach is to introduce rounding and filleting operators in CSG. The rounding operation transforms a solid by simultaneously rounding all its convex edges and vertices with the same radius. The dual operation fillets simultaneously all concave edges and vertices of a solid. The term blending will refer to both operations. Blended solids can be combined through Boolean operations and also used as arguments of other blending operations. Thus to blend a single edge E, we first blend a simple sub–solid that contains E, and then use Boolean operations to construct the desired fillet and combine it with the original solid.

1.2 OFFSETTING

An offset object is an expanded or contracted version of an original object. To expand a solid S by a positive distance r one adds to the solid all the points exterior to S that lie within a distance r of the boundary of S. To shrink (or contract) S by r one subtracts from S all the points within a distance r from its boundary. Solid offsetting appears to be very useful. Potential applications cover a wide range, from tolerance analysis and clearance testing, through modelling of such physical processes as coating or etching, to blending.

In this thesis we show that blending operations can be defined in terms of offsetting, and we propose to represent blended solids in an extended version of CSG containing offsetting operators. The design and

the implementation of offsetting facilities for dual–representation modellers raise issues similar to those discussed above in the context of blending. In essence, one must devise representations for boundaries of offset solids and set membership classification procedures, which are used for boundary evaluation and various applications.

1.3 THESIS ORGANIZATION

The thesis contains five major parts: blending, offsetting, representations and algorithms to support offsetting, edge approximations, and experimental implementation.

1) Chapter two covers blending semantics and discusses the advantages of our approach, which integrates blending in CSG through a global blending operator.

2) Chapter three provides mathematical definitions and studies topological and algebraic properties of offsetting operations. It also shows how offsetting operations can be combined to blend solids. Chapter four studies the boundaries of offset solids and explains how a superset of such boundaries can be computed.

3) Chapter five discusses requirements for representing offset solids in a modeller. Chapter six presents algorithms for supporting offsetting — and therefore blending — operations in solid modellers.

4) High level algorithms that compute approximations for intersection edges of surfaces that bound offset solids are proposed in chapter seven. The resulting approximations are PCC (Piecewise Circular Curves), whose nature, representations, and applications are presented in chapter eight. Chapter nine develops the detailed mathematics used in algorithms that operate on such PCC's.

5) Chapter ten describes our experimental solid modeller, outlines its architecture, presents some key algorithms, and discusses limitations and possible extensions.

CHAPTER II

BLENDING

In this chapter we review prior work on blending, discuss semantic issues that have an impact on the design of blending facilities for solid modellers, and propose a new approach to blending solids defined in CSG.

2.1 KNOWN APPROACHES TO BLENDING

Blending has long been recognized as an important problem in geometric modelling, but there is a relatively small body of research literature devoted to it. Approaches known to the author may be summarized as follows.

2.1.1 Annotation

The most obvious approach consists simply in treating a blend as an attribute or "note" associated with an edge of the object and specifying the radius of the blend. This approach was proposed for the PADL–1 system [Fisher 78] but never used. It was implemented at the University of Cambridge [Braid 80], to generate "symbolic", non–realistic graphic depictions of blended objects. Because the geometry of the blend is not fully specified, annotation *per se* is inadequate for the full range of applications supported by a solid modeller.

2.1.2 Function combination

More than a decade ago, Ricci [Ricci 73] devised a technique for describing solids through blended versions of the usual Boolean operations. Specifically, he considered solids defined by inequalities of the form $f(\mathbf{P}) \leq 1$, where $\mathbf{P}$ is a point of Euclidean space. Ricci defined the blended intersection of two solids A and B by the function

$$f_{A \cap B}(\mathbf{P}) = (f_A(\mathbf{P})^k + f_B(\mathbf{P})^k)^{1/k}$$

where the functions f_A and f_B define respectively A and B. Blended union and difference of two or more solids are defined similarly. When k approaches infinity, Ricci's operators approach the usual Boolean ones. For finite values of k the resulting solids are smoothly blended approximations to the usual Boolean combinations, and the degree of approximation can be controlled by varying k.

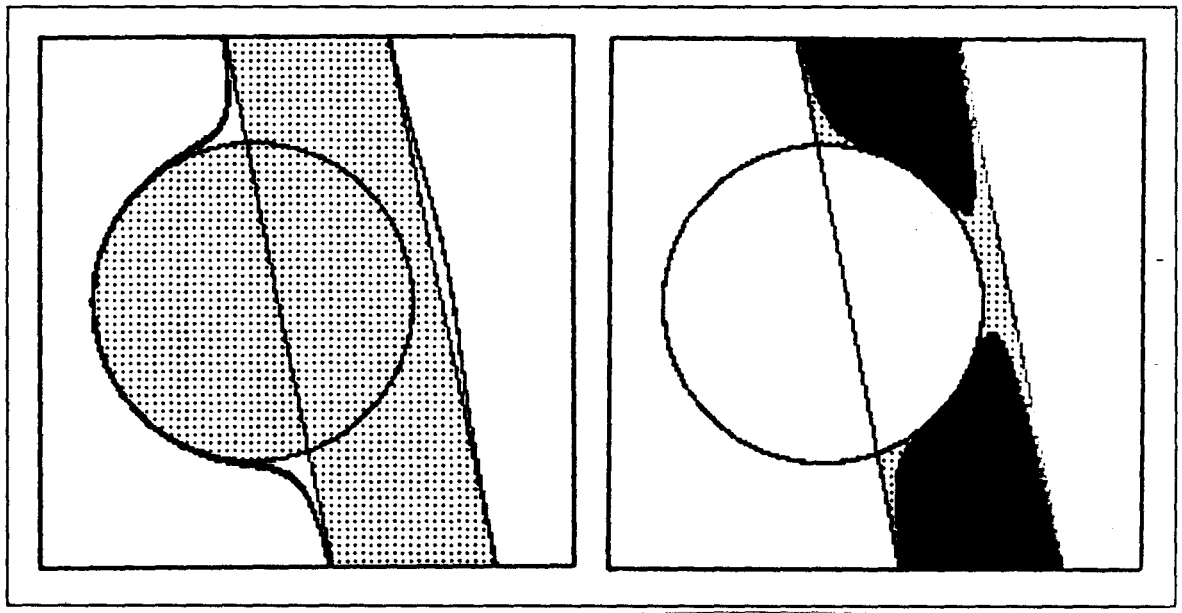

FIGURE 2.1: The union of a stripe with a circle is globally blended (left) by function combination: an unexpected "boss" appears in the originally straight edge that should not be modified by the operation. The difference of the two solids is rounded (right); the resulting blends are not circular.

A significant advantage of this approach is that the implicit equation of the blending surface ($f(\mathbf{P}) = 1$) is known and can be used to compute the intersection of curves with the boundary of the resulting solid. We

experimented with Ricci's method and concluded that it produces nicely rounded solids but may cause global object changes, because the effects of blending are difficult to localize (Figure 2.1). We also found that it does not provide easy control of the blending surface, which does not have constant radius.

Very recent work at Cornell University [Hoffmann 85] also provides techniques for deriving the implicit equation of a smooth blending surface F by suitably combining the implicit equations of two surfaces to be blended S_1 and S_2. The technique is promising, because the resulting surfaces are algebraic and of low degree. They are not, however, constant–radius surfaces, and their integration in solid modelling systems requires further study.

2.1.3 Sculptured surfaces

B–splines and other sculptured surfaces can be used to model most blends encountered in practice. (Non–rectangular patches, which have not been widely used until now, may be needed to model certain blended regions, especially near blended vertices.) Direct specification of blending surfaces by a user, e.g. by defining control points for a B–spline surface, is very difficult, and therefore facilities must be provided for automatically deriving the surface–defining data from simple user specifications. The following approaches have been proposed:

- *Recursive subdivision:* Polyhedra can be globally rounded by successively chopping (chamfering) their edges [Catmull 78], [Doo 78a], [Doo 78b]; some of the faces resulting from subdivision can be replaced by curved patches [Veenman 82]. This approach seems fruitful for designing solids with a generally rounded shape, but inadequate for controlling the shape or extent of the blends, and therefore has drawbacks similar to those of Ricci's operations.

- *Replacing edges by curved patches:* Researchers at the University of Tokyo report a technique for fitting patches between faces [Chiyokura 83]. The user marks all the edges to be rounded. Faces bounded by marked edges are split into four regions. Groups of two or more regions that share a marked edge are replaced by patches, in such a way that user–marked edges are smoothly blended.

- *Conformal surfaces:* Work under way at the University of Cambridge uses conformal mapping techniques to generate blending surfaces, which are then converted to B–splines [Rockwood 83].

Blending facilities based on sculptured surface techniques for dual–representation solid modellers have the following two major drawbacks:

- A modeller for sculptured objects must be fully implemented if blended objects are to be treated like other solids in the system. This implies, for example, that Boolean operations on sculptured solids must be supported.

- Patch manipulations are primarily surface operations, and techniques for converting the representation of blending faces into a CSG–compatible format are unknown.

2.1.4 Sweeping

Blending surfaces can be generated conceptually by sweeping (or rolling) a sphere in contact with the surfaces to be blended. Solid fillets can be generated by sweeping an appropriate (often *non–constant*) cross–section along the edge to be blended. Therefore sweeping techniques are potentially applicable to blending. Research on sweeping can be summarized as follows:

- A recent version of the CSG modeller TIPS includes a sweep primitive, defined by sweeping a constant planar cross–section along specific trajectories [Shirma 83]. However, it is not clear how to generate automatically the varying cross–section sweeps necessary to model complex fillets.

- A program verifier for numerically–controlled machine tools at General Dynamics [Fridshal 82] is based also on the TIPS system, and supports volumes swept by cutters in motion, but the mathematical and algorithmic methods used in the verifier have not been described publicly.

- Curve/surface intersection is one of the most important geometric utilities in a solid modeller. Morgan, at the General Motors Research Laboratories, defined the problem of intersecting a line with

the surface swept by a moving sphere in terms of a system of non–linear equations, which can be solved by homotopy continuation methods [Morgan 81]. These methods are robust, but fairly slow.

- Shaded displays of volumes swept by moving spheres are generated at Delph University [Van Wijk 84] by ray–casting techniques, using algebraic root–find procedures for intersecting lines with swept surfaces.

- Rolling–sphere surfaces are provided in several of the commercial turnkey CAD/CAM systems, which are based primarily on wire-frame representations [Requicha 80]. These systems have limited capabilities for "trimming" the surfaces so as to properly match abutting faces, and do not contain algorithms to support the operations required by solid modellers.

2.2 SEMANTIC ISSUES IN BLEND SPECIFICATION

Constant–radius blends are specified in drafting practice typically by showing a rounded profile in an appropriate view, and by indicating the nominal (or sometimes the maximal) value of the blending radius, which may be toleranced explicitly or by a general note in the drawing. The almost universally accepted interpretation for such a specification is that it defines a blending surface generated by conceptually rolling a sphere with the specified radius in contact with the faces being blended.

Specifying the blending radius and either the edge or the two faces to be blended is convenient for users of solid modellers, and is a direct extension of current practice. (Two–face specification is more general than edge specification, as shown in Figure 2.2). But what do such specifications mean exactly? In attempting to answer this question one is faced with several issues:

- *Complex vertices.* What is the nature of the blending surface in the neighborhood of a vertex where several convex and concave edges meet (Figure 2.3) or where three edges with different blending radii meet? Standard drafting practices seem to ignore these issues; blended vertices are not specified precisely, but the consensus seems to be that the resulting solids should be smooth.

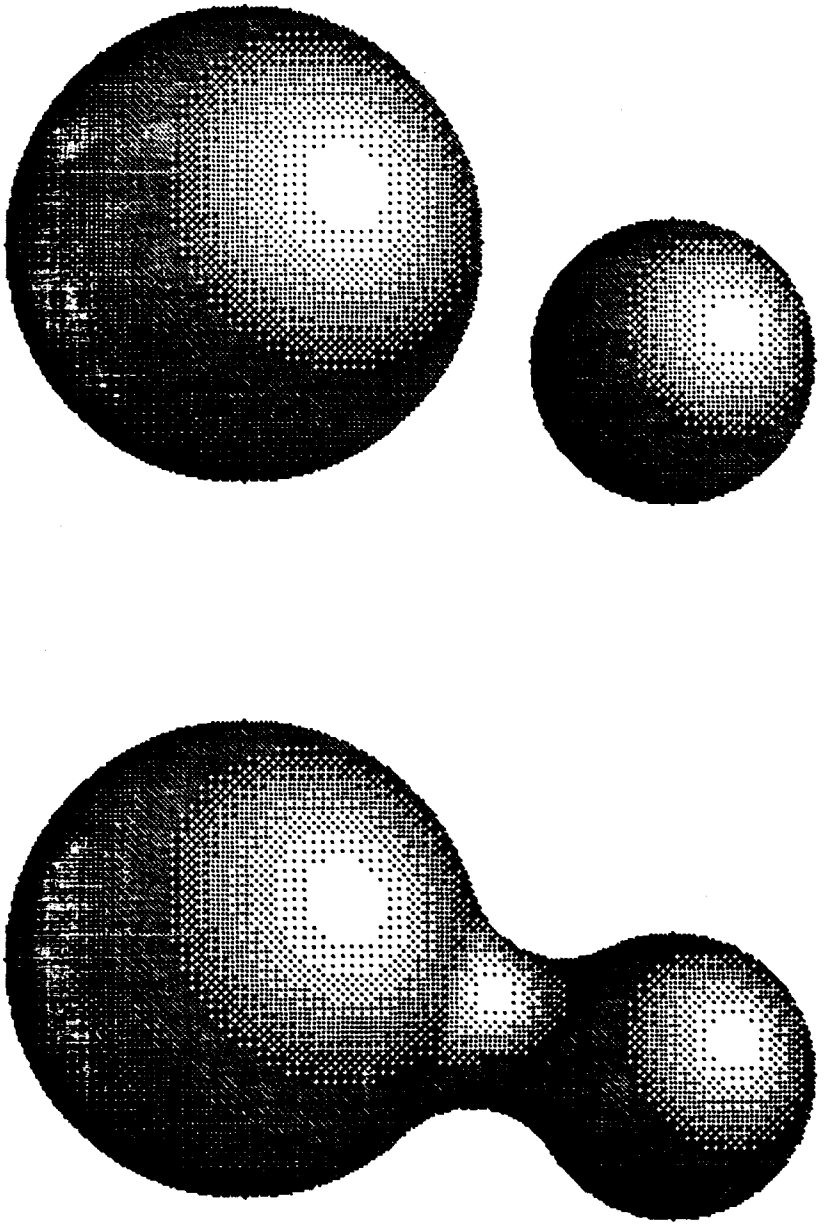

FIGURE 2.2: The union of two non intersecting spheres (top) can be filleted (bottom), although the solid has no edges.

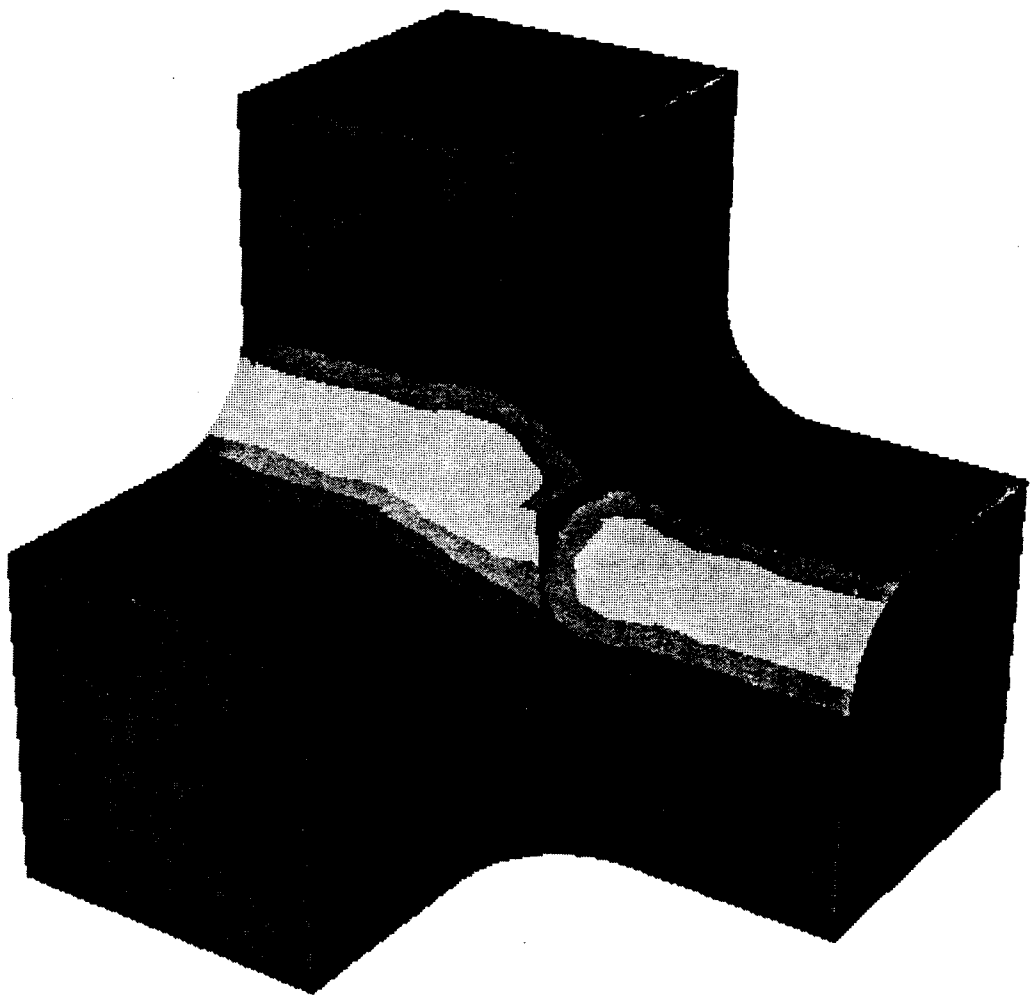

FIGURE 2.3: Six edges that intersect at a common vertex are all blended with the same radius. The blending surface near the common vertex should be smooth, but is not in this example, which was generated in PADL-2 by using toroidal surfaces.

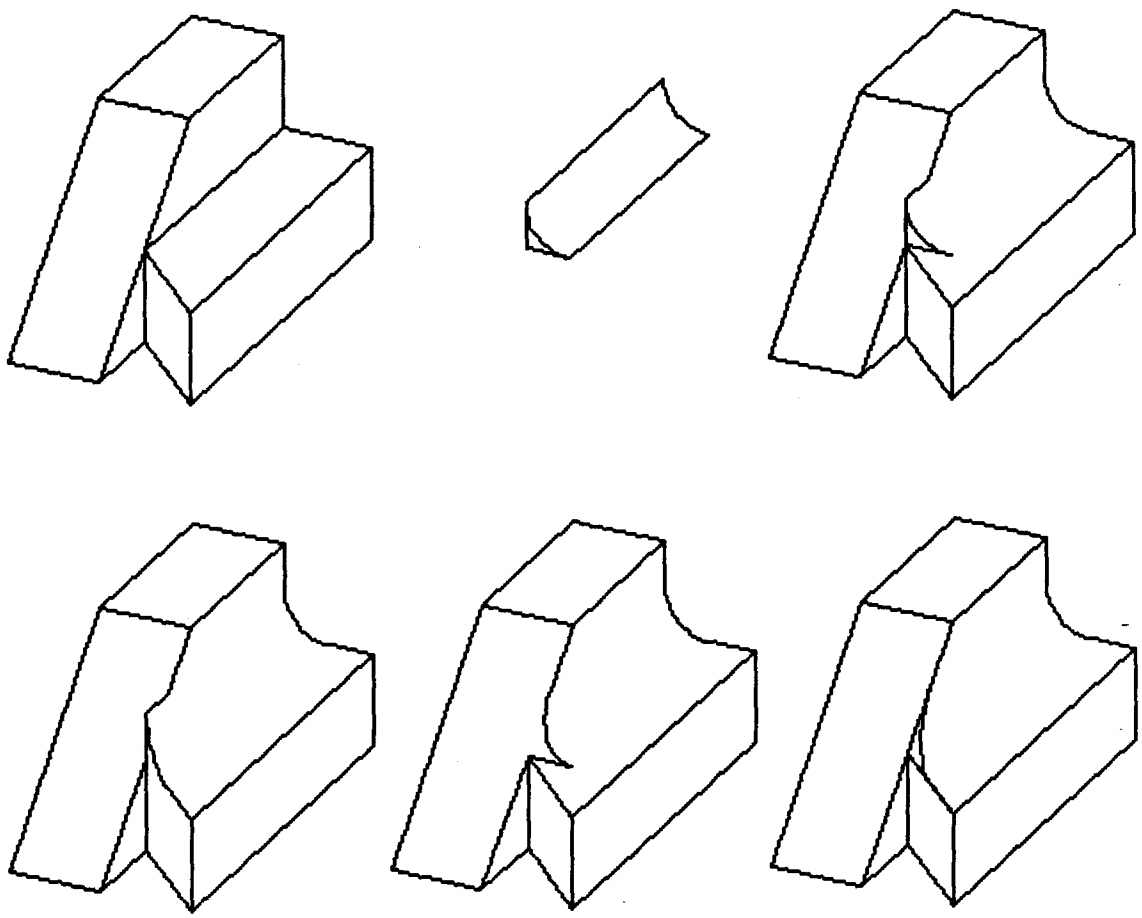

FIGURE 2.4: The concave edge E_1 (a) can be blended with the fillet (b). The fillet endings of (c), (d), and (e) are "obviously wrong". Bounding the fillet by the plane that contains edges E_2 and E_3 (f) seems reasonable. But what would we do if the edges E_2 and E_3 were curved?

- *End conditions.* How should blends be bounded when the edges being blended terminate in complex vertices? Drafting practice usually does not explicitly specify end conditions, and sometimes it is difficult to infer how blends should end (see Figure 2.4).

- *Blend interference.* Consider the specification of Figure 2.5. How should the two blends meet? Blend interference issues are resolved in blueprints by providing additional graphic information, but it is not clear how to specify analogous information in a solid modeller in a way that is convenient for the user.

- *Obliteration of details.* When a blending surface extends over an existing surface feature of an object, should the existing feature "disappear"? Figure 2.6 illustrates the issue.

- *Blend sequencing.* Is order significant when blending operations are specified sequentially in a modeller? Can blended objects be combined by Boolean operations? Can edges that result from blending operations be blended further? These issues do not arise in drafting practice because a blueprint depicts only a final object and does not record the intermediate steps in its definition.

- *Lack of smoothness of the blending surface.* Blending surfaces are not always smooth. Lack of smoothness may be due to the local behavior of the blend, e.g., when the radius of the rolling–sphere exceeds the radius of curvature of the trajectory along which the center of the sphere is moved (Figure 2.7). Global self–intersection of the envelope of the sweep might also create sharp edges.

- *Lack of smoothness at the junctions.* A blending surface may itself be smooth but join adjacent surfaces non–smoothly (Figure 2.8). These examples show that the specification of too large a blending radius is one of the major causes of non–smoothness.

Some of the issues discussed above also arise in current drafting practice but seem to be left to the common sense of manufacturing engineers, while other issues are indigenous to computer–based modelling systems. All must be considered by designers of blending facilities for solid modellers. Some awkward situations may be disallowed and flagged as errors,

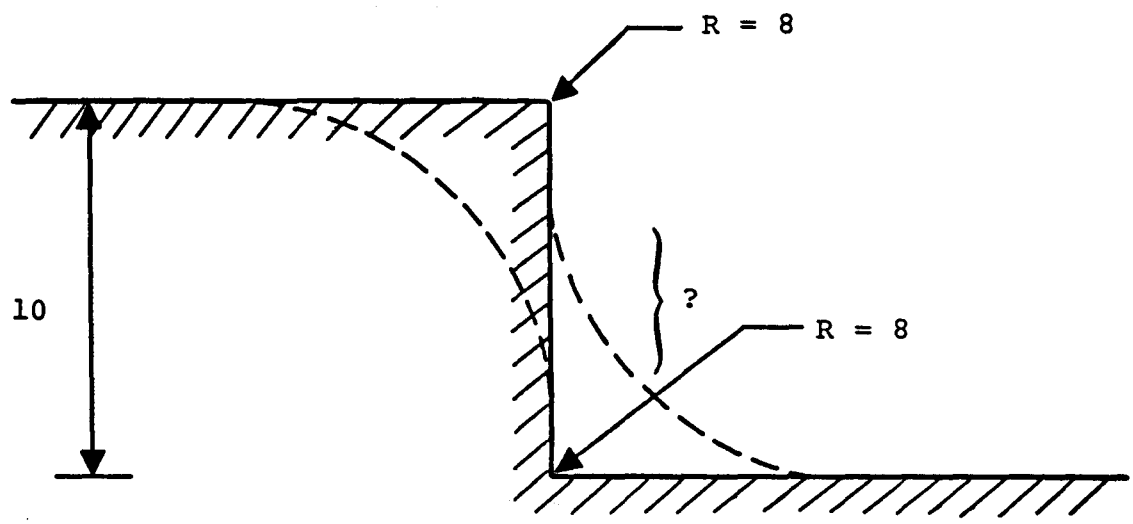

FIGURE 2.5: What is the shape resulting from these two blending operations?

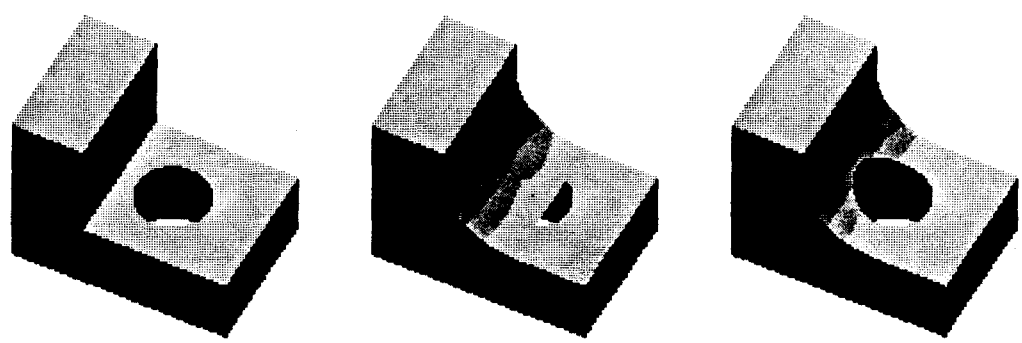

FIGURE 2.6: Blending the concave edge of the L–shaped object (left) might obliterate the circular hole, (center). A more reasonable interpretation (right) can be obtained by blending a sub–solid before the hole is made. It is not easy to design algorithms for "correctly" interpreting analogous but more complex cases.

FIGURE 2.7: Global and local self-intersections of the surface of the sweep.

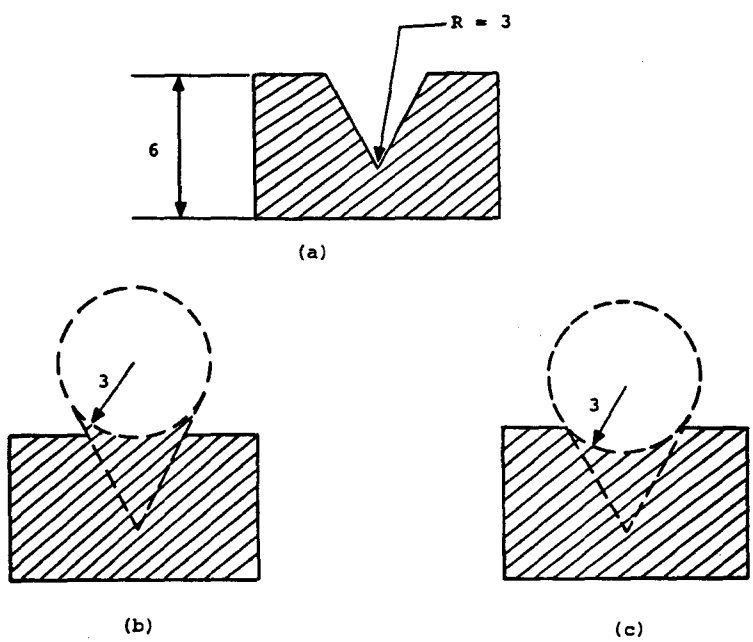

FIGURE 2.8: The concave edge is replaced by a circular blend. However, the blending face is not smoothly connected to the original solid.

but a reliable modeller must associate a valid solid object with any user specification that is not recognized as an error. A less crucial but very desirable feature is that a modeller ensure smoothness for all user–defined blends. Unfortunately, guaranteeing smoothness algorithmically seems neither easy nor cheap.

2.3 INTEGRATING BLENDING IN CSG

We propose a new approach to (constant–radius) rounding or filleting; its essence is to introduce in CSG a blending operator[1] that can be used per se to blend solids globally, or can be combined with Boolean operators to produce specific "blend primitives", which are subsequently added or subtracted to an original solid to produce the desired result. This approach has significant advantages:

- Blended solids can be represented in a form compatilbe with CSG, and can be combined by Boolean operations or blended further with different radii.

- Blend specification is independent of boundary representations, and therefore retains all the advantages of CSG. For example, one can easily parameterize blended solids and guarantee their validity.

- The high–level logic of all the CSG–based algorithms in a modeller is essentially unchanged; only a relatively small number of low–level geometric utilities is needed to support the new operator.

Our approach is illustrated in Figure 2.9. The sequence in which blends are performed is obviously important because the result of a series of Boolean operations generally depends on their order. Blended objects are treated essentially in the same way as unblended objects, and therefore can be blended successively and combined through Boolean operations. Some of the ambiguities in specification can be avoided by ordering blends and combining blended objects (see Figure 2.10 for an example).

[1] We shall see later that the blending operator is itself defined in terms of another operator, called offsetting.

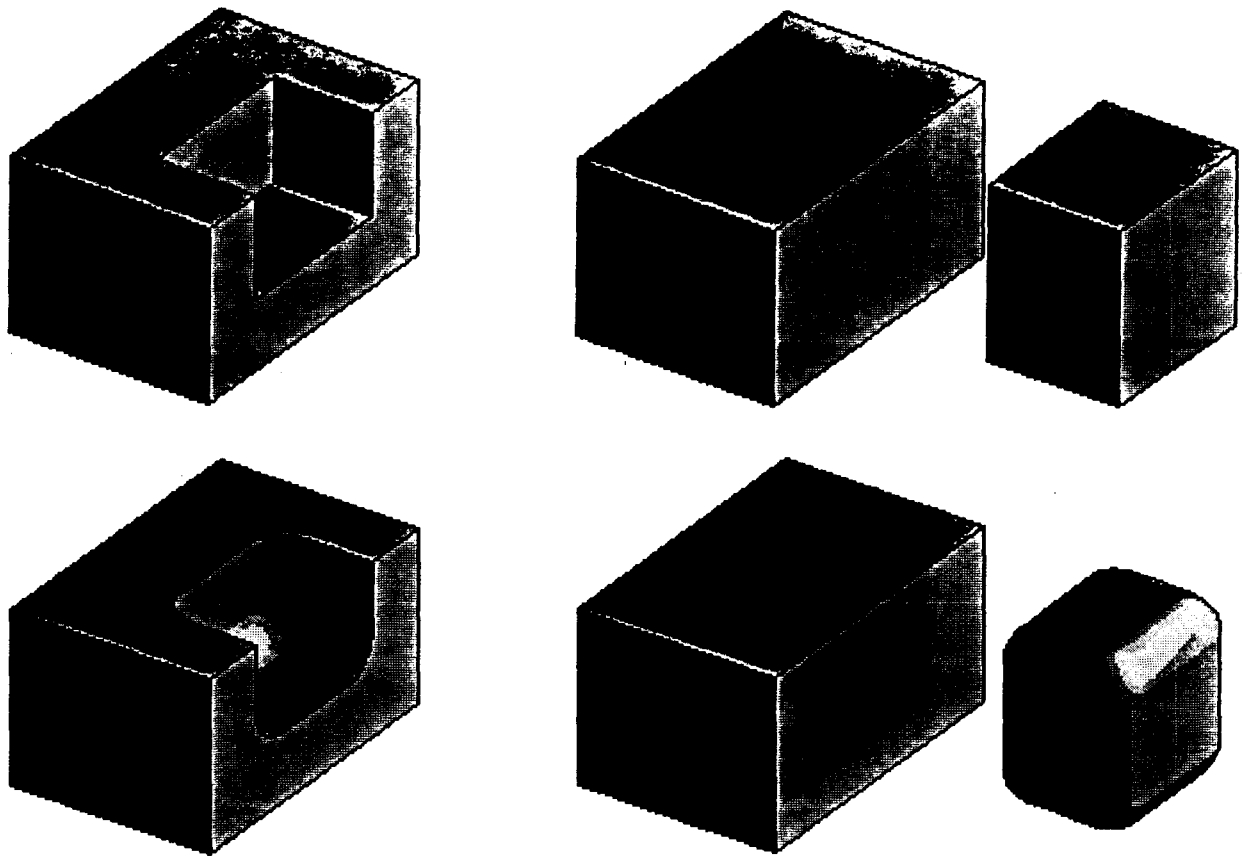

FIGURE 2.9: All edges of the pocket (top–left) can be filleted with specified radius (bottom–left). If the original solid is defined in CSG as the difference of two blocks (top–right), the desired result can be obtained by globally rounding the subtracted block (bottom–right) before subtraction.

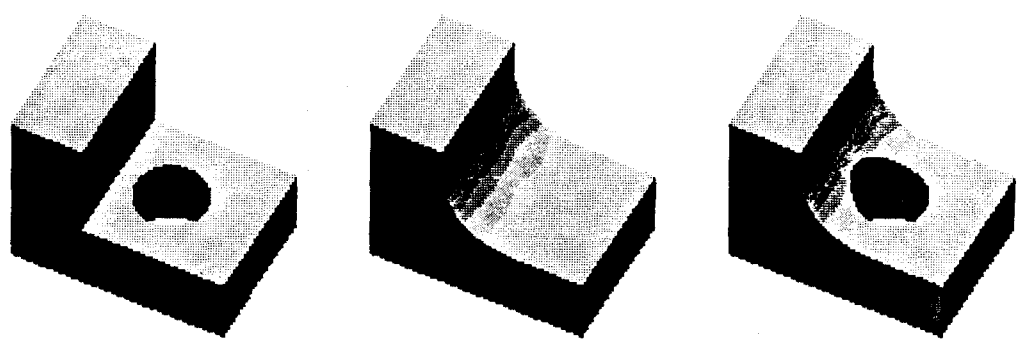

FIGURE 2.10: To fillet the concave edge of the solid (left) without covering the hole, one can first fillet the solid without the hole (center), then subtract an extended version of the hole (right).

This example typifies our approach: to provide users with facilities to define blends unambiguously, rather than to infer users' intentions from ambiguous specifications.

Global blending can still be used to produce localized fillets and rounds through addition or subtraction of appropriate blend primitives. These are specified by a three step procedure as shown in Figure 2.11. First the user defines a simple sub–solid Q that contains the faces to be blended and specifies a radius r for filleting Q. Typically, Q is the union of two half–spaces for concave–edge blends or the intersection of two half–spaces for convex–edge blends. Then, an oversized fillet B is produced by a set difference operation between Q and its blended version. The resulting solid may be theoretically unbounded, but in practice is bounded by some large, system–defined region guaranteed to enclose the user's universe. The third and final step eliminates the unwanted parts of B through Boolean operations. Trimming the blend to the correct "size" is usually accomplished by intersecting B with a box or another simple solid. The resulting blend primitive P is a solid in its own right, and may be manipulated in the same way as any other solid in the system. Presumably the user will combine P with the original S to obtain a locally blended solid, but P is available for other purposes as well. The combining operation is a *union* for filleting concave edges or a *difference* for rounding convex edges.

The specification procedure just described is somewhat inconvenient for users, but one can design interfaces that hide most of the annoying details (e.g., trimming) for the common, simple cases. We think that the ability to specify "manually" the blend primitive and to impose the desired end conditions by direct trimming through Boolean operations is a powerful and reasonable way to handle complexity. The second (or trimming) step in the blend specification procedure is often unnecessary.

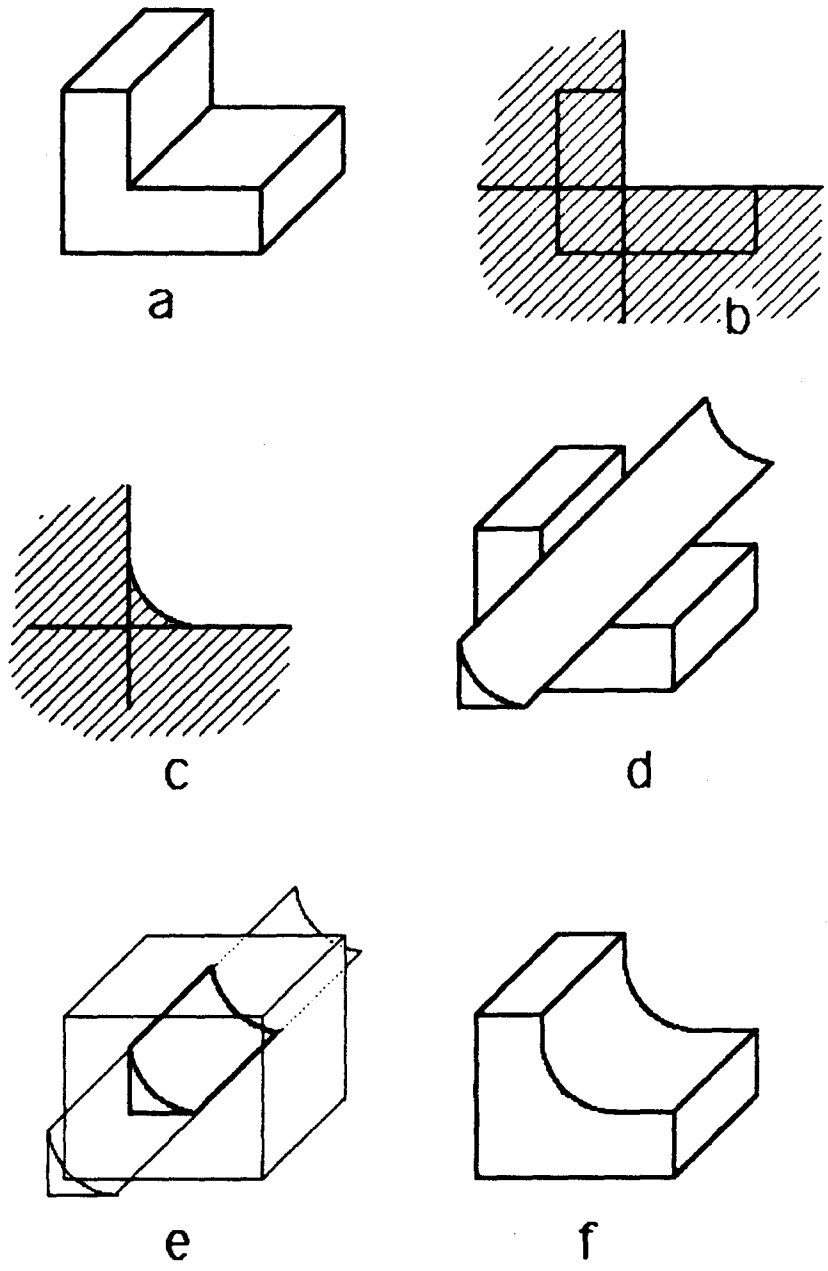

FIGURE 2.11: To define a blend for filleting the concave edge of the solid (a) the user first specifies a radius r and a sub–solid Q, which contains the concave edge. Q is the union of two half–spaces that contain the blended faces (b). Subtracting Q from its globally filleted version (c) produces the desired fillet B (d). The oversized fillet B is trimmed to P (e) by Boolean operations and combined with the solid (f).

2.4 SEMANTICS OF GLOBAL BLENDING

The semantics of blending is defined as follows. The globally rounded version of a solid S, is the region swept by a ball of radius r as it moves through S, remaining entirely inside of S at each "instant". (Precise mathematical definitions are given in chapter three.) Such an operation rounds with the same radius r, all convex edges and vertices of S, as illustrated in figure 2.12.

The dual operation is filleting, which is equivalent to rounding the complement of a solid — see figure 2.13 — and intuitively generates the region that cannot be reached by a ball that remains outside of S and whose radius is r.

Mathematical definitions of global blending operations that preserve regularity are proposed in chapter three. Such operations, when combined with regularized Boolean operations and applied to regular solid primitives are sufficiently powerful to produce the desired results for the common fillets and rounds, and ensure that these results are valid solids. Occasionally they produces surprising side-effects — e.g., fillets in unexpected regions when complex solids are blended with relatively large radii. Side effects are illustrated in Figure 2.14, which also shows how CSG specification can be used to obtain the desired result. The general approach is to define a sub-solid that contains all the features (edges and vertices) that should be blended with the same radius, blend it globally, and then combine it with the unblended parts of the original solid.

To blend simultaneously convex as well as concave edges and vertices of a solid, rounding and filleting operations can be combined (Figure 2.15). Such a combination of filleting and rounding in general is not commutative and does not always produce smooth solids (see Figure 2.16 for a counterexample).

As we shall demonstrate in chapter three, constant radius blending can be expressed in terms of offsetting operations. The remainder of this thesis will deal primarily with the mathematics of offset solids and with the design of representations and algorithms that support offsetting (and hence blending) in CSG-based modellers.

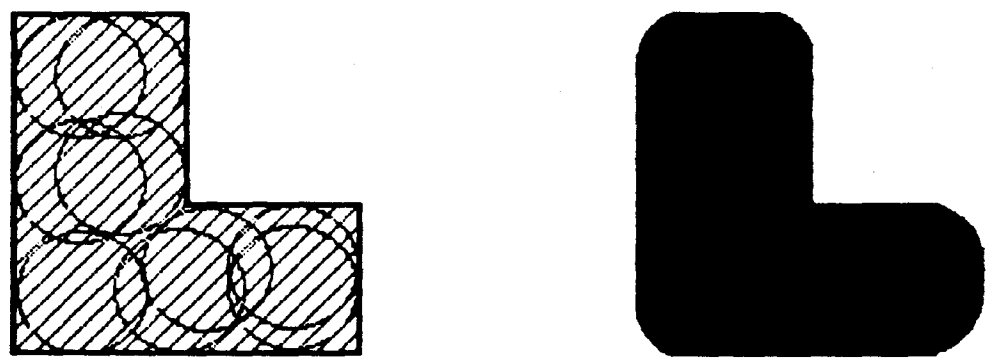

FIGURE 2.12: The ball of radius r remaining in S (left) cannot reach the convex corners. The result is a globally rounded solid (right).

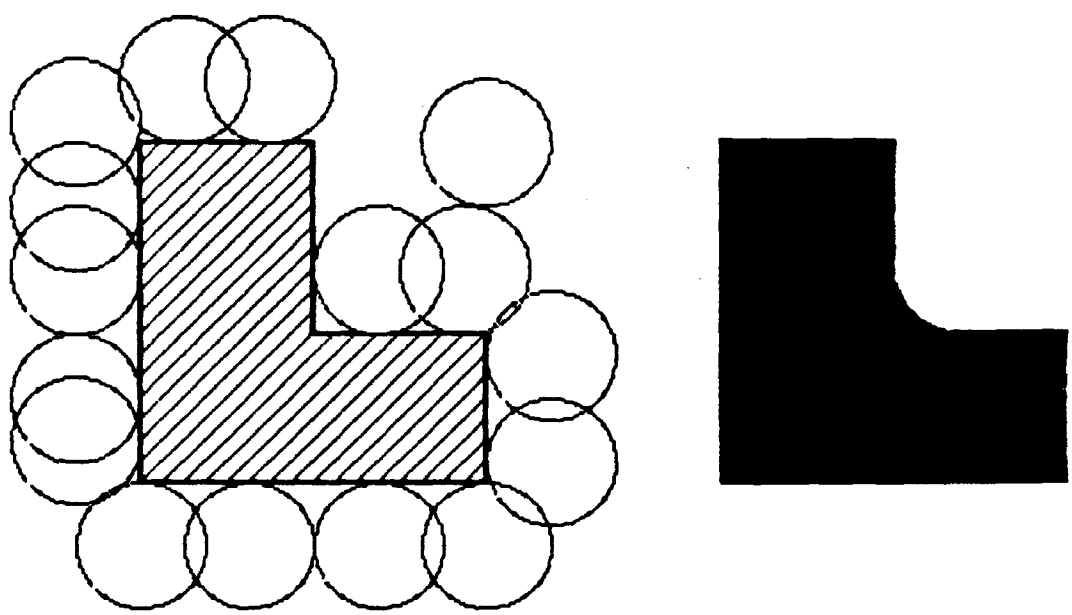

FIGURE 2.13: The ball of radius r remaining out of S (left) cannot reach the concave corners, which are blended in the globally filleted version of S (right).

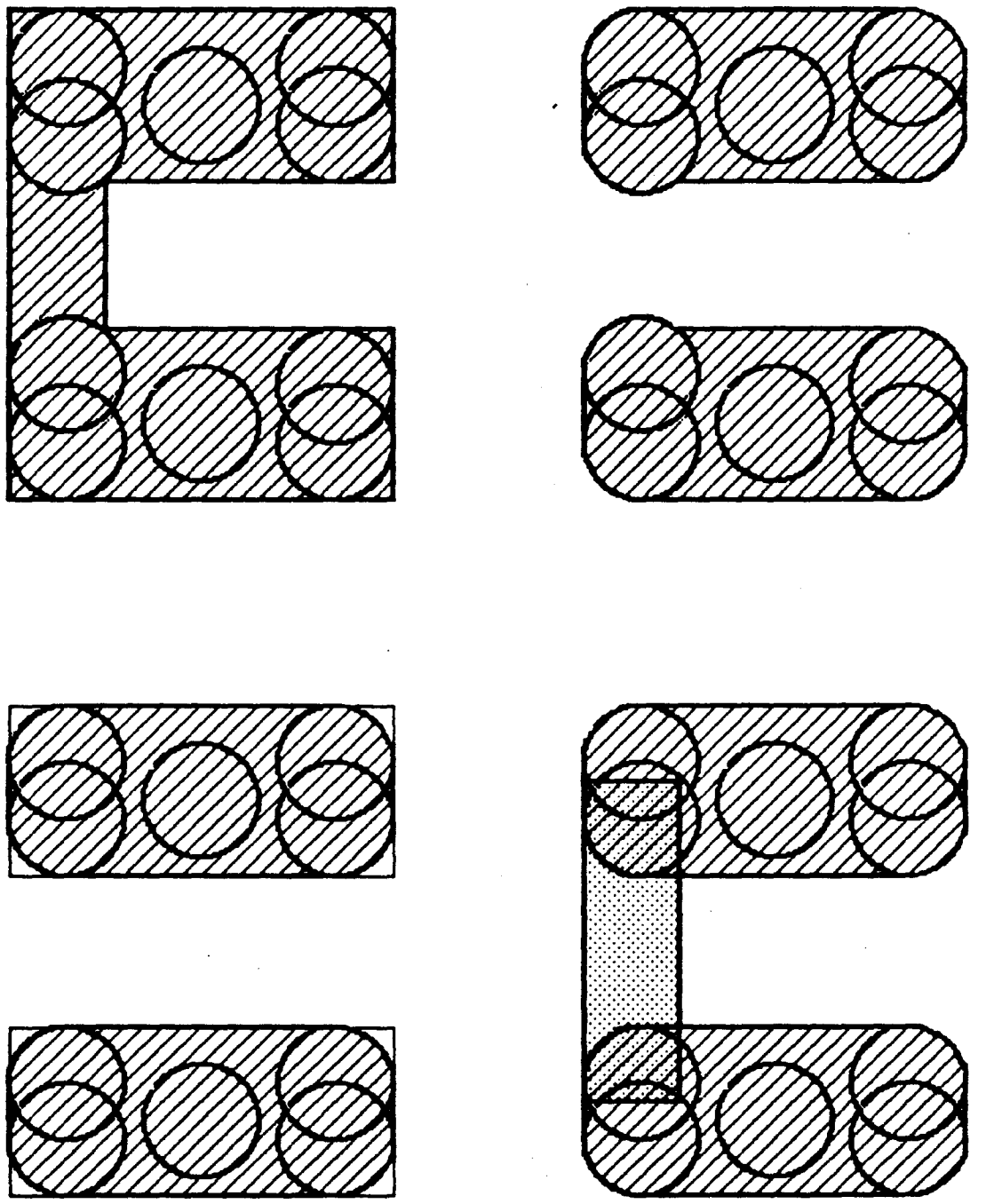

FIGURE 2.14: Globally rounding the solid S (top–left) will produce two disconnected parts (top–right) because the ball of radius r cannot penetrate the thin junction while remaining in S. The desired result (bottom–right) can be obtained by first rounding a sub–solid (bottom–left), then combining it with the thin unblended junction.

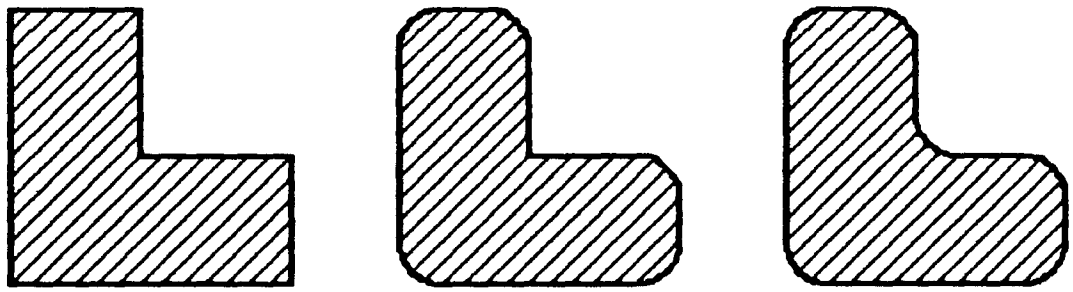

FIGURE 2.15: The solid (left) is first globally rounded (center) and then globally filleted (right). Both concave and convex edges are blended.

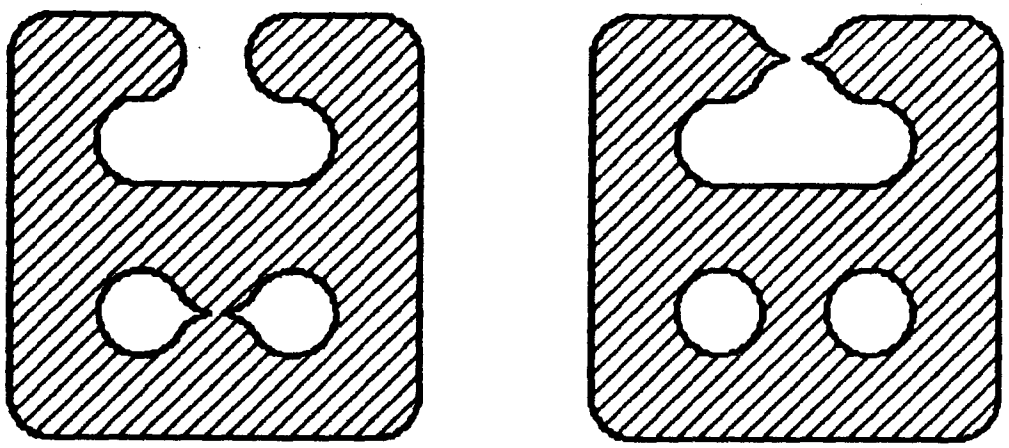

FIGURE 2.16: Filleting A (left) produces B (right). Rounding B yields A. Neither is smooth.

CHAPTER III

OFFSETTING

This chapter introduces *expanding* and *shrinking* transformations, also called respectively *positive* and *negative solid offsetting.* We review prior work, study the topological properties of offsets, and analyze how offsetting operations can be combined with other offsetting operations, with Boolean operations, and with rigid motions. Finally, we show how the global blending operations defined in chapter two can be expressed in terms of offsetting operations.

3.1 PRIOR WORK ON OFFSETTING

Variations of our offsetting operations have been used previously in the following contexts.

3.1.1 Interference and inclusion testing

Given two objects A and B, is $A \bigcap B = \emptyset$? Is $A \subset B$? To answer these questions it is often convenient to consider a point P of A and test it for inclusion in an expanded or contracted version B' of B, because point inclusion — or *point membership classification* [Tilove 80] — is relatively straightforward. This approach has been used for computing mass properties of solids [Lee 82] and for planning collision–free paths for robots [Lozano–Perez 79]. In both applications, objects are expanded or contracted by ad–hoc techniques that are related but not identical to our offsetting operations.

3.1.2 Design rule checking

In integrated circuit design, and especially in VLSI, it is important to ensure that certain design rules of a geometric nature are not violated. These rules typically involve minimal and maximal sizes, distances, and overlaps between 2–D rectangular entities. Variously–defined growing and shrinking operations on polygons have been used for rule checking in many CAD packages for VLSI. An exemplary package that supports the 2–D analogues of our offsetting operations is discussed in [Barton 80].

3.1.3 Cutter path generation

Numerically controlled (NC) machine tools are typically programmed in terms of cutter trajectories, which are specified by requiring that a cutter move in contact with given surfaces. However, machine tool control systems require trajectory specifications in terms of cutter centers or centerlines (CL data). To generate CL data for a ball–ended cutter with radius r from contact–surface information, one constructs a surface "parallel" to the contact surface by "moving along the normal" by distance r [Pressman 77, Faux 79]. (We refer to such an operation as *normal offsetting*, abbreviated *n–offsetting* to distinguish it from solid offsetting discussed in this chapter.)

CL data for cutting 2–D profiles with cylindrical cutters is generated by n–offsetting planar curves. The current understanding of n–offsetting of planar curves is summarized in [Tiller 84]. Briefly, there is no accepted mathematical definition of n–offsetting for curves and surfaces that are only piecewise smooth, and even when surfaces and curves are smooth, n–offsetting may lead to undesirable cusps and self–intersections. Heuristic approaches proposed in [Tiller 84] provide reasonable results in many but not all of the possible cases.

We shall study n–offsets in chapter four and use them to generate the boundaries of offset solids.

3.1.4 Mathematical Morphology

As we shall see below, offsetting operations are special cases of Minkowsky additions and subtractions. These are important in the field of geometric probability, and have been studied extensively by the French school of "mathematical morphology" [Matheron 75, Serra 82], which is primarily motivated by problems of texture analysis for geological applications. A few of the properties presented below have been derived independently in the mathematical morphology work.

3.2 DEFINITION OF OFFSETS

All sets discussed in this thesis are assumed to be subsets of the three–dimensional Euclidean space (E^3), and distances associated with offsetting operations are assumed to be positive. *Positive offsets* are obtained by *expanding* sets.

DEFINITION 3.1: **The expanded version of a set S by distance r is:**

$$S{\uparrow}r = \{P : \exists Q \in S \text{ such that } \|Q - P\| \leq r\}$$

It follows directly from this definition that both the empty set $\emptyset$ and the universal set E^3 are invariant under expansion. Positive offsets are called *generalized balls* in the mathematical literature [Nadler 78]. The following property (illustrated in Figure 3.1) shows that one can also define positive offsets in terms of closed balls $B(Q,r)$, where

$$B(Q,r) = \{P : \|P - Q\| \leq r\}$$

PROPERTY 3.1: **$S{\uparrow}r$ is equal to the union of all the closed balls of radius r whose center is in S.**

PROOF: **By definition of closed balls:**

$$\begin{aligned}\bigcup_{Q \in S} B(Q,r) &= \{P : P \in \bigcup_{Q \in S} B(Q,r)\} \\ &= \{P : \exists Q \in S \text{ such that } P \in B(Q,r)\} \\ &= \{P : \exists Q \in S \text{ such that } \|P - Q\| \leq r\} \\ &= S{\uparrow}r\end{aligned}$$

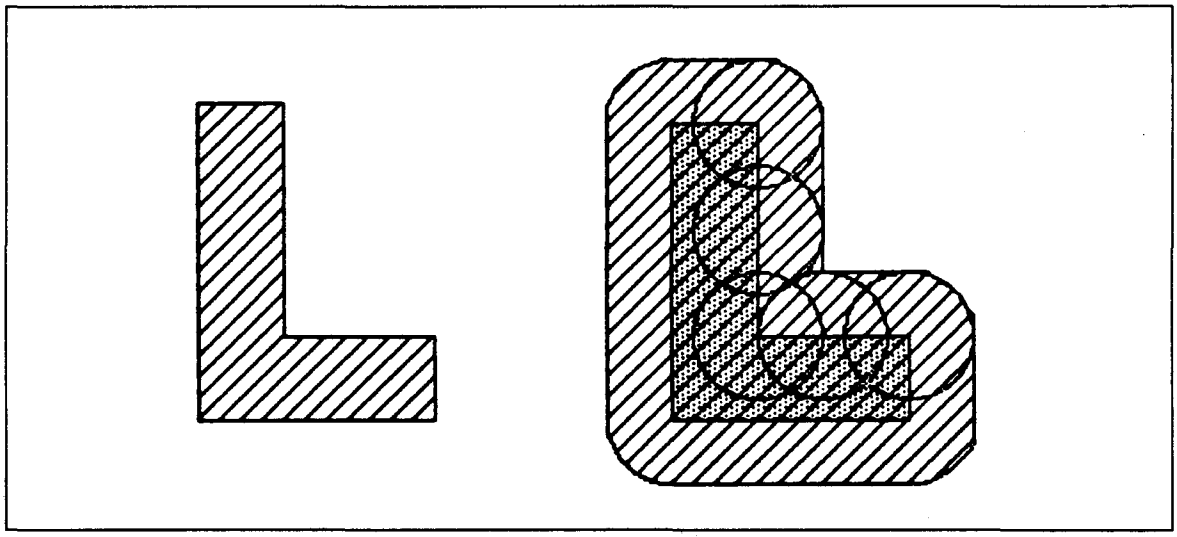

FIGURE 3.1: The expanded version of S is the region that can be reached by a sphere of radius r whose center remains in S.

This alternative definition shows that one can think of $S\uparrow r$ as being the region swept by a solid sphere $B(Q,r)$ of radius r, as its center Q moves through the entire set S. It follows that expanding lower dimensional sets (e.g. curves or surfaces) produces three dimensional sets.

Negative offsetting is defined in terms of positive offsetting and stands for both *regularized* and *non-regularized shrinking* (defined below). *Regularized offsetting* connotes expanding and regularized shrinking operations.

The topological *closure* of a set S is denoted kS and is defined as the union of S with the set of all limit points of S. The *interior* of a set S is denoted iS and is defined [Mendelson 75] as the set of points P such that S is a neighborhood of P, i.e., it contains an open ball around P. Regularization is an operation that takes a set S into kiS, which is shorthand for $k(iS)$. Regularized operations are defined as their usual counterparts followed by the regularization operation [Requicha 77a, Requicha 80].

DEFINITION 3.2: **Regularized shrinking takes a set S into $S\downarrow^* r$, defined as the regularized complement of the expanded regularized complement of S:**

$$S\downarrow^* r = \overline{\overline{S}^*\uparrow r}^*$$

DEFINITION 3.3: **Non–regularized shrinking takes a set S into $S{\downarrow}r$, defined as the complement of the expanded complement of S:**

$$S{\downarrow}r = \overline{\overline{S}{\uparrow}r}$$

It follows from these definitions that $\emptyset$ and E^3 are invariant under negative offsetting, and that regularized shrinking takes lower dimensional sets (e.g. surfaces or curves) into the empty set.

We shall prove below that regularized offsetting preserves regularity, but non–regularized shrinking can produce lower–dimensional sets (surfaces, curves or points). Since we are interested in modelling transformations that take solids onto solids, we shall use non–regularized shrinking only in conjunction with the dual expanding operation. As we shall see, such combinations produce regular sets that are globally blended.

Offsetting is related to Minkowsky addition, $S \oplus B$, and subtraction, $S \ominus B$ defined by [Matheron 75, Serra 82]:

$$S \oplus B = \{P:\ P = Q + R, Q \in S, R \in B\}$$
$$S \ominus B = \overline{\overline{S} \oplus B}$$

where S and B are sets and P, Q, R are points. Replacing B with the closed ball $B(O,r)$ of radius r centered at the origin yields:

$$\begin{aligned} S \oplus B(O,r) &= \{P:\ P = Q + R, Q \in S, \|R\| \leq r\} \\ &= \{P : \exists Q \in S, \|P - Q\| \leq r\} \\ &= S{\uparrow}r \end{aligned}$$

and

$$\begin{aligned} S \ominus B(O,r) &= \overline{\overline{S}{\uparrow}r} \\ &= S{\downarrow}r \end{aligned}$$

Matheron's results applied to Minkowsky addition and subtraction of closed balls can be used to derive some of the properties proven by other means in this chapter. We shall not elaborate on Matheron and Serra's work, but we wish to mention that one can deduce from their results some interesting properties, such as:

- offsetting and filleting preserve convexity;
- given two closed and convex sets A and B, $(A\uparrow r = B\uparrow r) \Rightarrow (A = B)$;
- denoting by S_V the translated version of the set S by a vector V:

$$S\uparrow r = \bigcup_{\|V\| \leq r} S_V \quad \text{and} \quad S\downarrow r = \bigcap_{\|V\| \leq r} S_V$$

3.3 TOPOLOGICAL PROPERTIES

3.3.1 Point/set distance

When a set S is *closed*, there is yet another equivalent definition of $S\uparrow r$ in terms of point/set distance. The distance $d(P,S)$ between a point P and a non–empty set S is defined [Nadler 78] by

$$d(P,S) = \inf_{Q \in S} \|P - Q\|$$

where *inf* denotes the *greatest lower bound*. It follows from this definition that if P is in S then $d(P,S) = 0$.

The following properties of the distance will be useful later; they are true for any non–empty set S.

PROPERTY 3.2: **For any pair of points P and R, $d(P,S) \leq d(R,S) + \|R - P\|$.**

PROOF: From the definition of $d(P,S)$ and from the *triangle inequality* it follows that, for any point R:

$$\begin{aligned} d(P,S) &= \inf_{Q \in S} \|P - Q\| \\ &\leq \inf_{Q \in S} (\|P - R\| + \|R - Q\|) \\ &\leq \|P - R\| + \inf_{Q \in S} \|R - Q\| \\ &\leq \|P - R\| + d(R,S) \end{aligned}$$

PROPERTY 3.3: **The distance $d(P,S)$ is a continuous function of P.**

PROOF: We shall prove that

$$\forall \epsilon > 0 \quad \exists \delta > 0 \quad \text{such that} \quad \|Q - P\| < \delta \Rightarrow |d(Q,S) - d(P,S)| < \epsilon$$

By property 3.2: $d(P,S) \leq d(Q,S) + \|P - Q\|$. Interchanging P and Q yields: $d(Q,S) \leq d(P,S) + \|P - Q\|$. Combining both results implies that:

$$|d(P,S) - d(Q,S)| \leq \|P - Q\|$$

For a given ϵ, we pick $\delta = \epsilon$ and we have

$$\|P - Q\| < \delta \Rightarrow |d(P,S) - d(Q,S)| < \epsilon$$

PROPERTY 3.4: $d(P,S) > 0 \iff P \in i\overline{S}$.

PROOF: Given $d(P,S) = r > 0$, all points Q of the open ball of radius $r/2$ around P are out of S, since, by property 3.2,

$$d(Q,S) \geq d(P,S) - \|P - Q\|$$

or $d(Q,S) \geq r - r/2 = r/2$. It follows that P has a neighborhood in $\overline{S}$ and therefore $P \in i\overline{S}$, or, equivalently, $P \notin kS$, because $\overline{kS} = i\overline{S}$ for any set S [Requicha 78].

Conversely, let $P \in i\overline{S}$. By definition of interior points: $\exists \epsilon > 0$, such that the open ball $O(P,\epsilon)$ of radius ϵ and center P is in $\overline{S}$, and therefore out of S. For each point $Q \in S$, the point

$$R = P + \frac{\epsilon}{2} \frac{Q - P}{\|Q - P\|}$$

$$d(P,S) = \inf_{Q \in S} \|P - Q\| \geq \frac{\epsilon}{2} > 0$$

For any set S: $P \notin i\overline{S} \Leftrightarrow P \in kS$. Therefore complementing property 3.4 yields:

PROPERTY 3.5: $P \in kS \iff d(P,S) = 0$.

3.3.2 Closest projection set

PROPERTY 3.6: $d(P,S) = d(P,kS)$.

PROOF: Consider a point P out of S. Because $S \subset kS$, the definition of distance implies that:

$$d(P,S) \geq d(P,kS)$$

To prove the reverse inequality, and therefore that $d(P,S) = d(P,kS)$, we consider an arbitrary point R of kS. By property 3.2, $d(P,S) \leq d(R,S) + \|P-R\|$, and since $d(R,S) = 0$ (property 3.5), we have $d(P,S) \leq \|P-R\|$. Therefore $d(P,S)$ is a lower bound on $\|P-R\|$ for $R \in kS$, and cannot be greater than the greatest lower bound $d(P,kS)$. Consequently

$$d(P,S) \leq d(P,kS)$$

We define the *closest projection set* $\{P\rightarrow|S\}$ of a point P on a set S, as the set of points of kS, for which the distance $d(P,kS)$ attains its minimum. Intuitively $\{P\rightarrow|S\}$ is the locus of all points of kS that are *closest* to P. We show that $\{P\rightarrow|S\}$ is a non–empty subset of the boundary of S.

DEFINITION 3.4: **Given a point P out of S, the *closest projection set* of P on S, denoted $\{P\rightarrow|S\}$, is the set of points Q of kS that are at distance $d(P,S)$ from P.**

PROPERTY 3.7: **Given a point P out of S, the closest projection set of P on S is not empty.**

PROOF: If $P \in kS$ then, by property 3.5, $d(P,S) = 0$ and $P \in \{P\rightarrow|S\}$. Therefore $\{P\rightarrow|S\} \neq \emptyset$. Now let us suppose that $P \notin kS$. By property 3.4 $d(P,S) > 0$. Let $S' = kS \bigcap B(P,2r)$, where $r = d(P,S)$. S' is the intersection of two closed sets and therefore is closed, and it is also bounded, because it is included in $B(P,2r)$. Therefore S' is compact since it is a closed and bounded subset of E^3 (Heine–Borel theorem [Mendelson 75]). Clearly all points of $kS - S'$ are at a distance from P larger or equal to $2r$, and, by property 3.6, $d(P,S) = d(P,kS)$. It follows that $d(P,S') = r$. Since, for a fixed P, $\|P-Q\|$ is a continuous function of Q, the greatest lower bound of $\|P-Q\|$ for Q in the compact set S' is attained for at least one point Q of S' ([Mendelson 75], page 161) and hence of kS.

PROPERTY 3.8: **Given a point P out of S, all interior points Q of S are at a distance from P greater that $d(P,S)$.**

PROOF: Let $Q \in iS$. Then, by definition of interior points, there is an open ball $O(Q)$ around Q that lies in S. This implies that there are points R in $O(Q)$, and hence in S, that are at a distance from P less than $\|P-Q\|$ (for example, the points of the intersection of $O(P)$ with the line segment (P,Q)). $d(P,S)$ is a lower bound of $\|P-R\|$ for all R in S, and there is a point R in S such that $\|P-R\| < \|P-Q\|$, therefore $d(P,S) \leq \|P-R\| < \|P-Q\|$. Thus all interior points Q of S must be at a distance from P greater than $d(P,S)$, therefore:

$$\forall P \in \overline{S},\ \forall Q \in iS \qquad \|P-Q\| > d(P,S)$$

The *boundary* of a set S, denoted ∂S, is defined [Mendelson 75] as the set of points for which all neighborhoods intersect both S and its complement $\overline{S}$. It is also expressed as $kS \bigcap k\overline{S}$ [Requicha 78], or equivalently as the difference $kS - iS$. From properties 3.7 and 3.8 we conclude that points in the closest projection set of P on S are in kS but not in iS, therefore:

PROPERTY 3.9: **Given a point P out of S, $\{P \rightarrow\!| S\} \subset \partial S$.**

The shortest distance from P to S (or, equivalently, to kS) is achieved for at least one point of ∂S. It follows immediately that:

PROPERTY 3.10: **Given a point P out of S, $d(P,S) = d(P,\partial S)$.**

3.3.3 Closure under offsetting

If S is closed it contains its boundary, and therefore the *inf* in the definition of $d(P,S)$ is always attained at points Q of S, and therefore can be replaced with *min* for closed sets.

PROPERTY 3.11: **If S is closed then**

$$d(P,S) = \min_{Q \in S} \|P-Q\|$$

This leads to an alternative definition of positive offsets for closed sets.

PROPERTY 3.12: **For S closed, $S{\uparrow}r = \{P : d(P,S) \leq r\}$.**

PROOF: First we show that

$$\{P : d(P,S) \leq r\} \subset S\uparrow r$$

Let $d(P,S) \leq r$. Since S is closed the minimum of $\|P-Q\|$ (in property 3.11) is attained for at least one point Q of S:

$$\exists Q \in S \text{ such that } \|P-Q\| = d(P,S)$$

and therefore $\|P-Q\| \leq r$. By definition 3.1, this implies $P \in S\uparrow r$. To prove the converse, let $P \in S\uparrow r$. By definition 1, $\exists Q \in S$ such that $\|P-Q\| \leq r$, and therefore $d(P,S) \leq r$, by definition of distance $d(P,S)$.

It follows that to expand a closed set S, one adds to it all points that are out of S, and whose distances to ∂S do not exceed r.

PROPERTY 3.13: **For closed S, $S\uparrow r = S \bigcup \left((\partial S)\uparrow r\right)$.**

PROPERTY 3.14: **For closed S, $S\uparrow r$ is closed.**

PROOF: It suffices to show that $S\uparrow r$ contains all its limit points. Let P be a limit point of $S\uparrow r$. Then any ball $B(P,\epsilon)$, with $\epsilon > 0$, contains at least one point R of $S\uparrow r$, such that $R \neq P$. By continuity of the point/set distance (property 3.3):

$$\epsilon \to 0 \quad \Rightarrow \quad R \to P \quad \Rightarrow \quad d(R,S) \to d(P,S)$$

$R \in S\uparrow r$ implies $d(R,S) \leq r$, and the limit $d(P,S)$ is also less or equal to r. Therefore, by property 3.12, $P \in S\uparrow r$.

PROPERTY 3.15: $d(P,S) < r \Rightarrow P \in i(S\uparrow r)$.

PROOF: Given a point P such that $d(P,S) < r$, we shall prove that there is a neighborhood of P in $S\uparrow r$. By definition of point/set distance, $(\forall Q \in S,\ \|P-Q\| \geq r) \Rightarrow d(P,S) \geq r$, and therefore

$$d(P,S) < r \Rightarrow \exists Q \in S \text{ such that } \|P-Q\| < r$$

It follows that $\exists \epsilon > 0$ and $\exists Q \in S$ such that $\|P-Q\| \leq r-\epsilon$. Therefore $\forall R \in B(P,\epsilon)$, $\|R-Q\| \leq \|R-P\| + \|P-Q\| = \epsilon + r - \epsilon = r$, and $B(P,\epsilon) \subset S\uparrow r$, and therefore $P \in i(S\uparrow r)$.

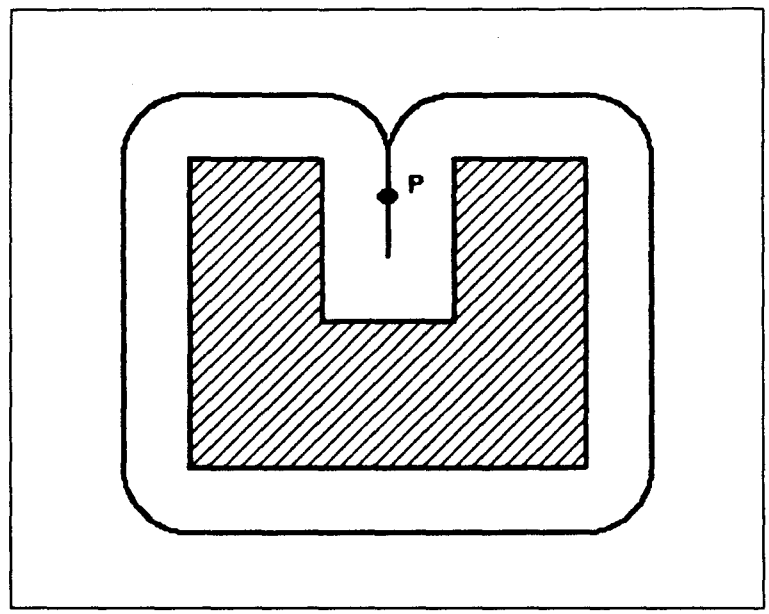

FIGURE 3.2: The point P is at distance r from S, but is in $i(S\uparrow r)$ and not in $\partial(S\uparrow r)$.

We show below that all points of the boundary of $S\uparrow r$ are at a distance r from S. However, as one can see on Figure 3.2, the converse is not true.

PROPERTY 3.16: $k(S\uparrow r) = (kS)\uparrow r$.

PROOF: $S \subset kS$, and since, as we show later, offsetting preserves set inclusion, $S\uparrow r \subset (kS)\uparrow r$. This implies $k(S\uparrow r) \subset k\big((kS)\uparrow r\big)$ because closure preserves inclusion relations. By property 3.14, $(kS)\uparrow r$ is closed and therefore $k\big((kS)\uparrow r\big) = (kS)\uparrow r$. It follows that

$$k(S\uparrow r) \subset (kS)\uparrow r$$

Now we must prove the reverse inclusion. Given $P \in (kS)\uparrow r$, $d(P, kS) \leq r$, and, by property 3.6, $d(P, S) \leq r$. Therefore, as in the proof of property 3.8, $\forall \epsilon > 0$, $\exists Q \in B(P, \epsilon)$, such that $d(Q, S) < d(P, S)$, and therefore $d(Q, S) < r$, which, by property 3.15, implies that $Q \in i(S\uparrow r)$. It follows that P is a limit point of $i(S\uparrow r)$, therefore $P \in ki(S\uparrow r)$, and, since $ki(S\uparrow r) \subset k(S\uparrow r)$, $P \in k(S\uparrow r)$ and

$$(kS)\uparrow r \subset k(S\uparrow r)$$

PROPERTY 3.17: $P \in \partial(S\uparrow r) \quad \Rightarrow \quad d(P, S) = r$.

PROOF: Given $P \in \partial(S\uparrow r)$, since the closure of a set is the union of the set with its boundary, $P \in k(S\uparrow r)$, and by property 3.16, $P \in (kS)\uparrow r$. It follows that $d(P, kS) \leq r$ because kS is closed, and, by property 3.6, that

$$d(P, S) \leq r$$

On the other hand, since $P \notin i(S\uparrow r)$, from property 3.15 it follows that

$$d(P, S) \geq r$$

PROPERTY 3.18: $P \in \overline{S\uparrow r}^* \Rightarrow d(P, S) \geq r.$

PROOF:

$$\begin{aligned}
P \in \overline{S\uparrow r}^* &\Rightarrow P \in k\big(i(\overline{S\uparrow r})\big) \\
&\Rightarrow P \in k\big(\overline{k(S\uparrow r)}\big) \text{ since } i\overline{X} = \overline{kX} \\
&\Rightarrow P \in \overline{i\big(k(S\uparrow r)\big)} \text{ since } \overline{iX} = k\overline{X} \\
&\Rightarrow P \notin i\big(k(S\uparrow r)\big) \\
&\Rightarrow P \notin i\big((kS)\uparrow r\big) \text{ by property 3.16} \\
&\Rightarrow d(P, kS) \geq r \text{ from property 3.15} \\
&\Rightarrow d(P, S) \geq r \text{ from property 3.6}
\end{aligned}$$

PROPERTY 3.19: **$S\uparrow r$ is regular for closed S.**

PROOF: By definition, $S\uparrow r$ is regular if and only if it is equal to the closure of its interior: $S\uparrow r$ is regular $\Leftrightarrow S\uparrow r = ki(S\uparrow r)$. A set contains its interior, therefore $i(S\uparrow r) \subset S\uparrow r$, and, for any two sets A and B, we have: $A \subset B \Rightarrow kA \subset kB$. It follows that $ki(S\uparrow r) \subset k(S\uparrow r)$. By property 3.14, $S\uparrow r$ is closed, and therefore it is equal to its closure $k(S\uparrow r)$. Combining the last two results we have $ki(S\uparrow r) \subset S\uparrow r$.

To show that $S\uparrow r \subset ki(S\uparrow r)$, we consider any point P of $S\uparrow r$. Since S is closed, by property 3.12, $d(P, S) \leq r$. If $d(P, S) < r$ then, by property 3.15, $P \in i(S\uparrow r)$ and therefore $P \in ki(S\uparrow r)$. By properties 3.7 and 3.9, if $d(P, S) = r$ then $\exists Q \in \partial S$ such that $\|P - Q\| = r$. $\forall \epsilon > 0$, the ball $B(P, \epsilon)$ contains a point $I \neq P$, for instance

$$I = P + \frac{\epsilon}{2} \frac{Q - P}{\|Q - P\|} \quad ,$$

such that $\|I - Q\| < r$, and therefore $d(I, S) < r$, which implies, by property 3.15, that $I \in i(S\uparrow r)$. Therefore P is a limit point of $i(S\uparrow r)$, i.e., $P \in ki(S\uparrow r)$.

Since regular sets are closed, and regularized shrinking is defined via regularized operations (definition 3.2), it follows from property 3.19 that:

PROPERTY 3.20: **Regularized offsetting operations take regular sets into regular sets.**

This property is important since it shows that any combination of regular sets (e.g., solid primitives) through regularized Boolean or regularized offset operations is a regular set. We can add regularized offset operators to the set of Boolean operators supported in CSG thereby defining an extended version of CSG called *CSG with offsetting* (abbreviated CSGO). Any set represented by a CSGO tree with regular primitives is guaranteed to be regular because of property 3.20.

3.4 ALGEBRAIC PROPERTIES OF OFFSETTING

In this section we analyze how offsetting operations can be combined with other offsetting operations, with Boolean operations, with rigid motions, and with inclusion relations between sets.

3.4.1 Invariance of inclusion under offsetting

PROPERTY 3.21: **The inclusion relation between any two sets is invariant under offsetting.**

PROOF: Consider any two sets A and B, such that $A \subset B$. By definition 3.1, $\forall P \in A\uparrow r$, $\exists Q \in A$ such that $\|P - Q\| \leq r$. Since $A \subset B$, $Q \in B$ and therefore $P \in B\uparrow r$ by definition 3.1. It follows that

$$A \subset B \quad \Rightarrow \quad A\uparrow r \subset B\uparrow r$$

Applying this result to complements yields $\overline{A} \subset \overline{B} \Rightarrow \overline{A}\uparrow r \subset \overline{B}\uparrow r$, and therefore:

$$B \subset A \Rightarrow \overline{A} \subset \overline{B} \Rightarrow \overline{A}\uparrow r \subset \overline{B}\uparrow r \Rightarrow \overline{\overline{B}\uparrow r} \subset \overline{\overline{A}\uparrow r} \Rightarrow B\downarrow r \subset A\downarrow r$$

To prove the same property for regularized shrinking operations, we apply

$$A \subset B \Rightarrow A\uparrow r \subset B\uparrow r$$

to regularized complements of any sets A and B:

$$\overline{A}^* \subset \overline{B}^* \quad \Rightarrow \quad \overline{A}^*\uparrow r \subset \overline{B}^*\uparrow r$$

Because i and k operators preserve inclusion and the complement operator reverses it:

$$B \subset A \quad \Rightarrow \quad \overline{A} \subset \overline{B} \quad \Rightarrow \quad \overline{A}^* \subset \overline{B}^*$$

It follows that: $B \subset A \quad \Rightarrow \quad \overline{A}^* \uparrow r \subset \overline{B}^* \uparrow r$. Complementing again both sides of the last inclusion relation yields

$$\overline{\overline{B}^* \uparrow r}^* \subset \overline{\overline{A}^* \uparrow r}^*$$

which, by definition, is equivalent to $B \downarrow^* r \subset A \downarrow^* r$. Therefore:

$$B \subset A \quad \Rightarrow \quad B \downarrow^* r \subset A \downarrow^* r$$

3.4.2 Distributivity over Boolean operations

PROPERTY 3.22: **For any set S: $\overline{S}\uparrow r = \overline{S \downarrow r}$ and $\overline{S} \downarrow r = \overline{S \uparrow r}$. For S regular: $\overline{S}^* \uparrow r = \overline{S \downarrow^* r}^*$ and $\overline{S}^* \downarrow^* r = \overline{S \uparrow r}^*$.**

PROOF: By definition 3.3: $\overline{S \downarrow r} = \overline{\overline{S} \uparrow r} = \overline{S} \uparrow r$ and $\overline{S} \downarrow r = \overline{\overline{S} \uparrow r} = \overline{S \uparrow r}$. For any regular set A: $\overline{\overline{A}^*}^* = A$. This follows from property 3.2.8 of [Requicha 78]. (For *regular sets* regularized Boolean operators are a Boolean algebra and therefore have all the properties of the usual Boolean operators.) Any operator preserves set equality, therefore, for any regular sets A and B: $A = B \quad \Rightarrow \overline{A}^* = \overline{B}^*$. Applying the above to regularized complements of A and B yields:

$$\overline{A}^* = \overline{B}^* \quad \Rightarrow \quad \overline{\overline{A}^*}^* = \overline{\overline{B}^*}^* \quad \Leftrightarrow \quad A = B$$

Combining both implications yields: $A = B \quad \Leftrightarrow \quad \overline{A}^* = \overline{B}^*$. Since, by property 3.19, $\overline{S}^* \uparrow r$ is regular, we obtain $\overline{S \downarrow^* r}^* = \overline{S}^* \uparrow r$ by complementing both sides of $S \downarrow^* r = \overline{\overline{S}^* \uparrow r}^*$. We can replace S by $\overline{S}^*$ in $S \downarrow^* r = \overline{\overline{S}^* \uparrow r}^*$, and obtain $\overline{S}^* \downarrow^* r = \overline{S \uparrow r}^*$ for regular S.

PROPERTY 3.23: **Expanding transformations distribute over the union of any two sets and also over the regularized union of any two regular sets.**

PROOF: For any two sets A and B, $(A \bigcup B)\uparrow r$ can be expressed as:

$$\begin{aligned}(A \bigcup B)\uparrow r &= \{P : \exists Q \text{ such that } Q \in (A \bigcup B) \text{ and } \|P-Q\| \leq r\} \\ &= \{P : \exists Q \text{ such that } \big((Q \in A) \text{ or } (Q \in B)\big) \\ &\qquad \text{and } \|P-Q\| \leq r\} \\ &= \{P : (\exists Q \in A \text{ such that } \|P-Q\| \leq r) \\ &\quad \text{or } \ (\exists Q \in B \text{ such that } \|P-Q\| \leq r)\} \\ &= \{P : (\exists Q \in A \text{ such that } \|P-Q\| \leq r)\} \\ &\quad \bigcup \ \{P : (\exists Q \in B \text{ such that } \|P-Q\| \leq r)\} \\ &= (A\uparrow r) \bigcup (B\uparrow r)\end{aligned}$$

We used here the fact that the union of two sets defined by properties a and b is: $\{P : a(P)\} \bigcup \{P : b(P)\} = \{P : a(P) \text{ or } b(P)\}$. By property 3.20, for A and B regular, $A\uparrow r$ and $B\uparrow r$ are also regular. Since for regular sets $\bigcup$ is the same as $\bigcup^*$ [Requicha 78], for any two regular solids A and B, $(A \bigcup^* B)\uparrow r = (A\uparrow r) \bigcup^* (B\uparrow r)$.

This property is illustrate Figure 3.3.

PROPERTY 3.24: **Regularized shrinking transformations distribute over the regularized intersection of any two regular sets and non–regularized shrinking distributes over the intersection of any two sets.**

PROOF: Given two regular sets A and B, by applying De Morgan's laws to $\overline{(A \bigcap^* B)}\downarrow^* r$ and by using the above property, we obtain:

$$\begin{aligned}(A \bigcap^* B)\downarrow^* r &= \overline{\left(\overline{A \bigcap^* B}^{\,*}\right)\uparrow r}^{\,*} \\ &= \overline{(\overline{A}^* \bigcup^* \overline{B}^*)\uparrow r}^{\,*} \\ &= \overline{(\overline{A}^*\uparrow r) \bigcup^* (\overline{B}^*\uparrow r)}^{\,*} \\ &= \overline{(\overline{A}^*\uparrow r)}^{\,*} \bigcap^* \overline{(\overline{B}^*\uparrow r)}^{\,*} \\ &= (A\downarrow^* r) \bigcap^* (B\downarrow^* r)\end{aligned}$$

The same demonstration holds for non–regularized shrinking of any two sets if one replaces regularized complement, union and shrinking operators by their non–regularized counterparts.

Since the Boolean difference can be expressed in terms of intersection, negative offsetting can be distributed over the difference (Figure 3.4).

PROPERTY 3.25: **For any sets A and B: $(A - B)\downarrow r = (A\downarrow r) - (B\uparrow r)$ and for A and B regular: $(A -^* B)\downarrow^* r = (A\downarrow^* r) -^* (B\uparrow r)$.**

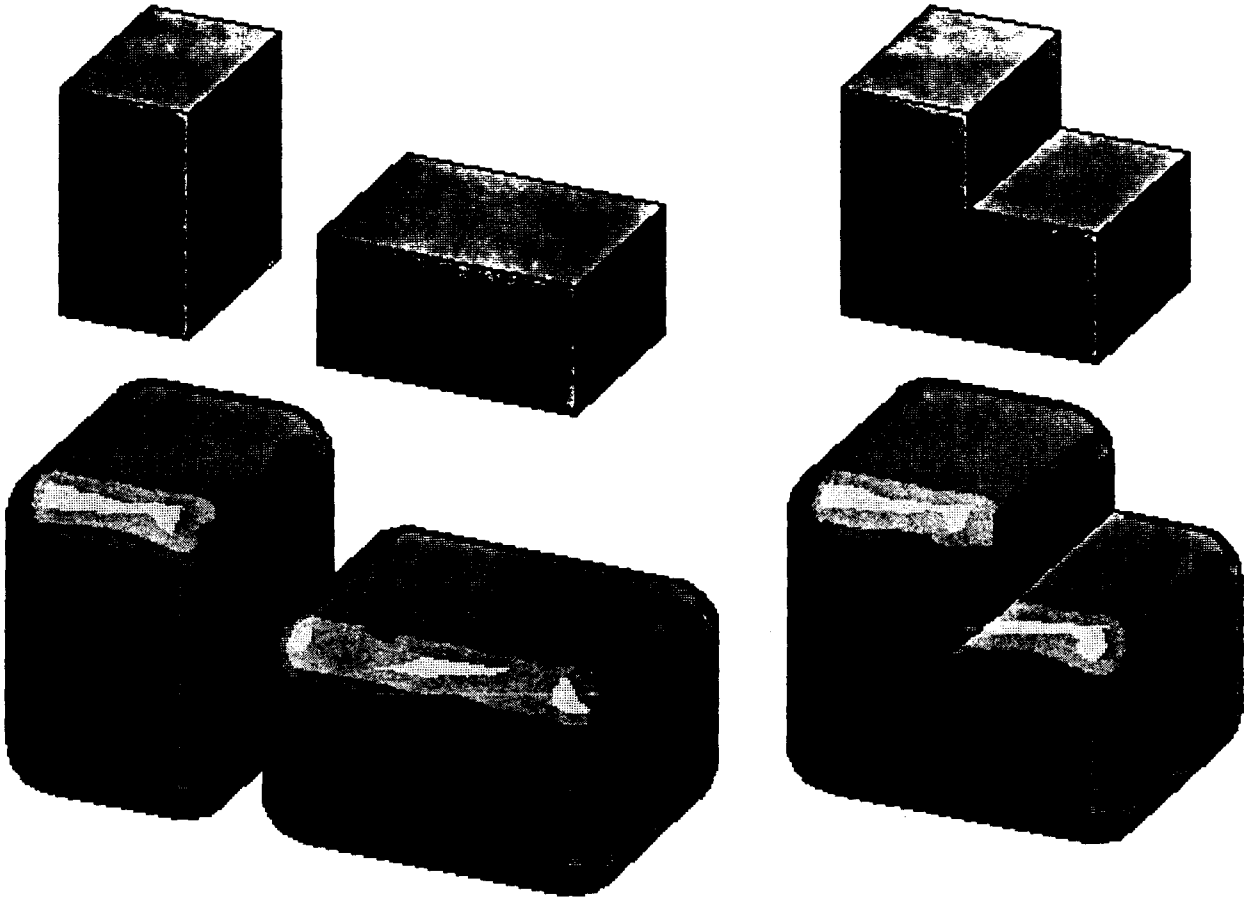

FIGURE 3.3: The union (top–right) of two blocks (top–left) is expanded (bottom–right). The same result is obtained by expanding the two blocks (bottom–left) before the union.

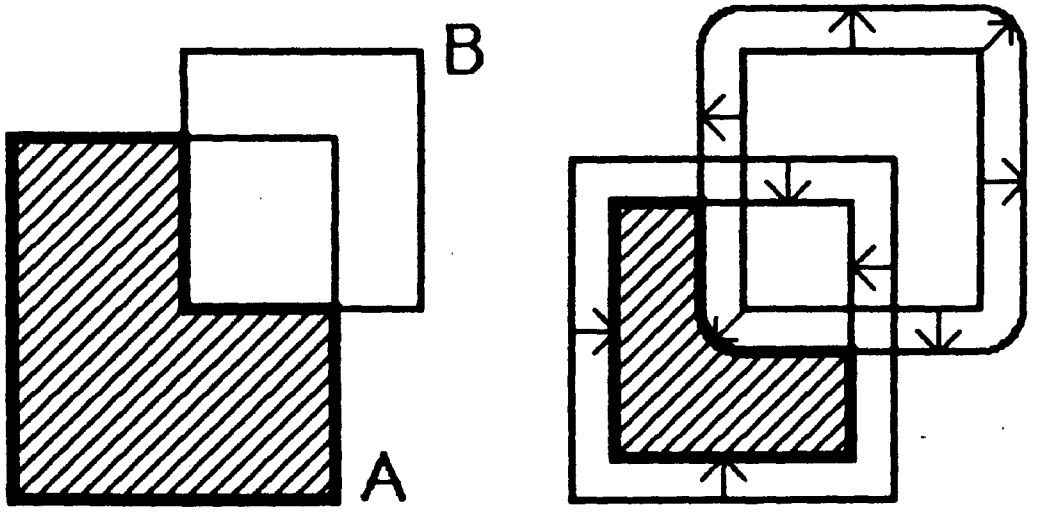

FIGURE 3.4: The solid defined as $A -^* B$ (left) is offset into $A\downarrow^* r -^* B\uparrow r$ (right).

PROOF: For regular A and B:

$$\begin{aligned}(A -^* B)\downarrow^* r &= (A \bigcap^* \overline{B}^*)\downarrow^* r \\ &= A\downarrow^* r \bigcap^* \overline{B}^*\downarrow^* r \quad \text{property 3.24} \\ &= A\downarrow^* r \bigcap^* \overline{B\uparrow r}^* \quad \text{property 3.22} \\ &= (A\downarrow^* r) -^* (B\uparrow r)\end{aligned}$$

The same demonstration holds for any sets A and B, if one replaces regularized difference, complement, and shrinking by their non-regularized counterparts.

On the other hand, expanding is not *always* distributive over the regularized intersection (Figure 3.5) and shrinking is not *always* distributive over the union (Figure 3.6). More specifically,

PROPERTY 3.26: $(A \bigcap^* B)\uparrow r \subset (A\uparrow r) \bigcap^* (B\uparrow r)$.

PROOF: From definition 3.1, it follows that

$$\begin{aligned}(A \bigcap B)\uparrow r &= \{P : \exists Q \in (A \bigcap B), \|Q - P\| \leq r\} \\ &\subset \{P : \exists Q \in A, \|Q - P\| \leq r \text{ and } \exists R \in B, \|R - P\| \leq r\} \\ &\subset (A\uparrow r) \bigcap (B\uparrow r)\end{aligned}$$

To prove that the above inclusion holds for regularized intersection let $P \in (iS)\uparrow r$. By definition 3.1 it follows that $\exists Q \in iS$ such that $\|P - Q\| \leq r$, and therefore $\exists \epsilon > 0$ such that the open ball of radius ϵ and center Q is in S. Let I be a point of such a ball, such that I also lies in the segment PQ (for instance let $I = Q + \frac{\epsilon}{2}\frac{P-Q}{\|P-Q\|}$). $\|P - I\| \leq r - \frac{\epsilon}{2} < r$ and $d(P, S) < r$, because $I \in S$. From property 3.15 it follows that $P \in i(S\uparrow r)$. We proved that $(iS)\uparrow r \subset i(S\uparrow r)$, and therefore that $k\big((iS)\uparrow r\big) \subset ki(S\uparrow r)$, which by property 3.16 implies

$$(kiS)\uparrow r \subset ki(S\uparrow r)$$

It follows that

$$\begin{aligned}(A \bigcap^* B)\uparrow r &\subset ki\big((A \bigcap B)\uparrow r\big) \\ &\subset ki\big((A\uparrow r) \bigcap (B\uparrow r)\big) \text{ because } k \text{ and } i \text{ preserve inclusion} \\ &= (A\uparrow r) \bigcap^* (B\uparrow r)\end{aligned}$$

Since $A \subset B \Rightarrow \overline{B}^* \subset \overline{A}^*$, applying De Morgan's laws and property 3.22 to the complement of the above inclusion, one obtains:

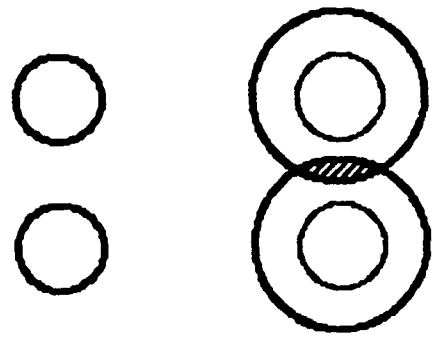

FIGURE 3.5: Given two disjoint spheres A and B (left) $A \bigcap^* B = \emptyset$ and therefore $(A \bigcap^* B)\uparrow r = \emptyset$. However $(A\uparrow r) \bigcap^* (B\uparrow r) \neq \emptyset$ (right).

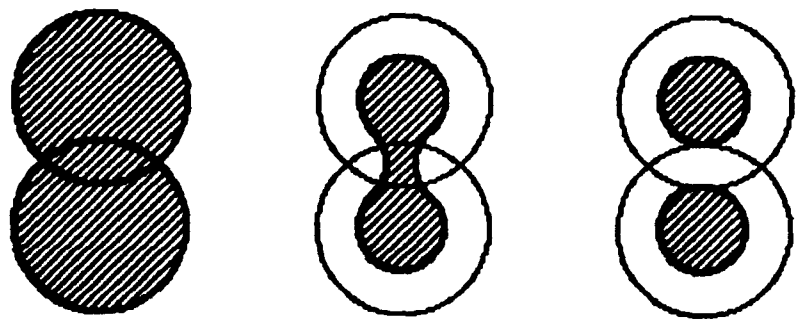

FIGURE 3.6: The shrunk version (center) of the union of two balls (left) differs from the union of the shrunk versions of the balls (right).

PROPERTY 3.27: $(A \cup^* B)\downarrow r \supset (A\downarrow r) \cup^* (B\downarrow r)$.

Therefore to offset a solid S defined in CSG as a Boolean combination of primitives P_i, one cannot in general offset each primitive and combine the results. However, this can be done in special cases. To characterize such cases, let us define the *positive formulation* of a CSG representation as follows. First, replace all the regularized difference operators by regularized intersection operators and complement their right operands ($A -^* B \longrightarrow A \cap^* \overline{B}^*$); then propagate the complements down to the primitives using De Morgan's laws ($\overline{A \cup^* B}^* \longrightarrow \overline{A}^* \cap^* \overline{B}^*$ and $\overline{A \cap^* B}^* \longrightarrow \overline{A}^* \cup^* \overline{B}^*$). The result is called the *positive formulation* of a solid's CSG representation, and is a Boolean combination of *regularized unions* and of *regularized intersections* of the original primitives P_i or of their regularized complements $\overline{P_i}^*$.

Properties 3.23 and 3.24 imply that, if the positive formulation of the CSG expression of a solid S is the regularized intersection of primitives P_i, $S\downarrow^* r$ can be expressed as the regularized intersection of $P_i\downarrow^* r$; similarly positive offsets by r of solids represented in CSG as the union of primitives P_i, can be expressed as the union of $P_i\uparrow r$.

As we shall see in the following chapter, offsets of *standard primitives*[2] can be defined in CSG in terms of standard primitives (see Figure 3.7 for an example).

It follows that a restricted class of offsetting operations can be performed in CSG without extending the geometric domain of current-generation modellers, but in general offsetting requires new surface types as shown in Figure 3.8. For offset solids that cannot be expressed in standard CSG one can obtain a useful approximation as follows. Let S be any solid defined in CSG, and let P_i be the primitives of a positive formulation of S. From properties 3.23 and 3.26 it follows that $S\uparrow r$ is included in the solid obtained by replacing in the positive formulation of S each primitive P_i by its expanded version $P_i\uparrow r$. Therefore the appropriate combination of offset primitives provides a CSG formulation of a solid that is guaranteed to contain $S\uparrow r$. Similarly, from properties 3.24 and 3.27 it follows that replacing each primitive by $P_i\downarrow^* r$ produces a solid that is contained in $S\downarrow^* r$.

[2] The following solids are considered as standard primitives: sphere, cylinder, cone, cuboid, and torus.

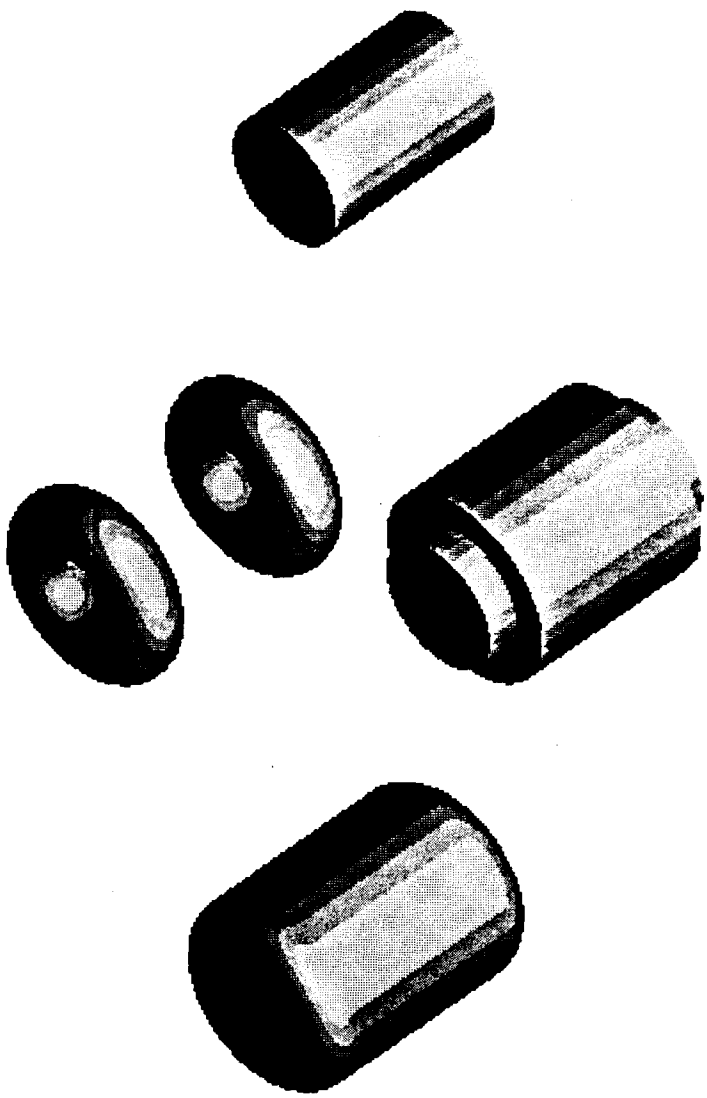

FIGURE 3.7: To expand a cylinder (top) we union two cylinders and two tori (center). The boundary of the result is smooth (bottom).

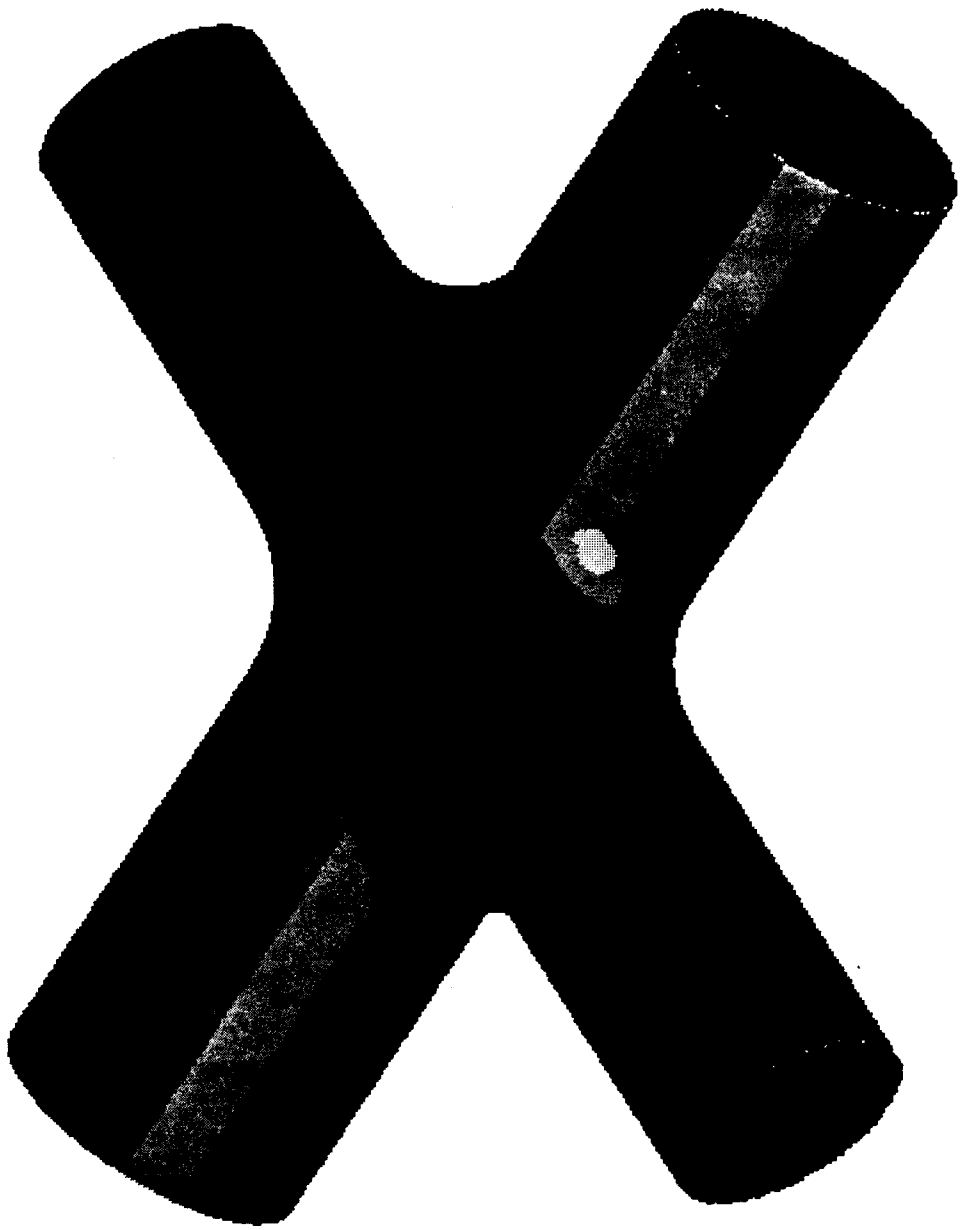

FIGURE 3.8: The shrunk version of the union of two cylinders cannot be obtained in CSG with standard primitives.

3.4.3 Combining offsetting operations

PROPERTY 3.28: **For any set S: $(S\uparrow a)\uparrow b = S\uparrow(a+b)$.**

PROOF: We remind the reader that a and b are assumed positive. We shall use the equivalence:

$$\left(\exists Q : \|P-Q\| \leq b \text{ and } \|Q-R\| \leq a\right) \Leftrightarrow \left(\|P-R\| \leq (a+b)\right)$$

The "$\Rightarrow$" implication follows from the triangle inequality. The "$\Leftarrow$" implication can be proven by choosing Q on the line segment (P,R), in such a way that: $\|P-Q\| \leq b$ and $\|Q-R\| \leq a$, which is always achieved for:

$$Q = \frac{aP+bR}{(a+b)}$$

Using the above equivalence, we have:

$$\begin{aligned}(S\uparrow a)\uparrow b &= \{P : \exists Q \in S\uparrow a \text{ such that } \|P-Q\| \leq b\} \\ &= \{P : \exists Q \text{ and } \exists R \in S \text{ such that} \\ &\qquad \|P-Q\| \leq b \text{ and } \|Q-R\| \leq a\} \\ &= \{P : \exists R \in S \text{ such that } \|P-R\| \leq (a+b)\}\end{aligned}$$

PROPERTY 3.29: **For any set S: $(S\downarrow a)\downarrow b = S\downarrow(a+b)$, and for regular S: $(S\downarrow^* a)\downarrow^* b = S\downarrow^*(a+b)$.**

PROOF:

$$\begin{aligned}(S\downarrow^* a)\downarrow^* b &= \overline{\overline{(S\downarrow^* a)}^*\uparrow b}^* \text{ by definition 3.2} \\ &= \overline{(\overline{S}^*\uparrow a)\uparrow b}^* \\ &= \overline{\overline{S}^*\uparrow(a+b)}^* \text{ by property 3.28} \\ &= S\downarrow^*(a+b) \text{ by definition 3.2}\end{aligned}$$

From definition 3.3 it follows that the same demonstration holds for non-regularized shrinking if one replaces regularized shrinking and complement by their non-regularized counterparts.

From properties 3.28 and 3.29 it follows immediately that:

PROPERTY 3.30: **Two expanding operations commute, two regularized shrinking operations commute, and two non-regularized shrinking operations commute.**

On the other hand, expanding operations do not *always* commute with shrinking operations. A counterexample is shown in Figure 3.9.

3.4.4 Commutativity with rigid motions

Offsetting transformations are defined in terms of distances which are invariant under rigid motions, therefore:

PROPERTY 3.31: **Rigid motion transformations commute with offsetting operations.**

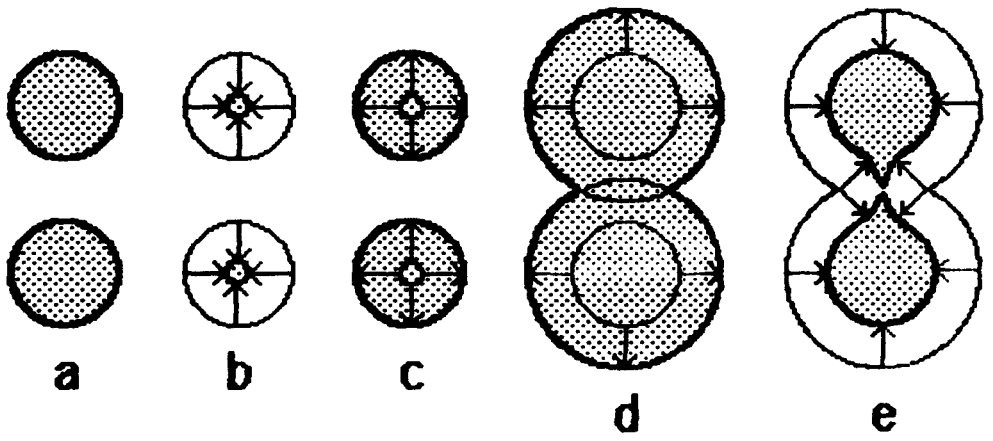

FIGURE 3.9: The union of two disjoint spheres (a) is not altered by the combination of shrinking (b) followed by expanding (c). However, expanding (d) followed by shrinking (e) produces a different result.

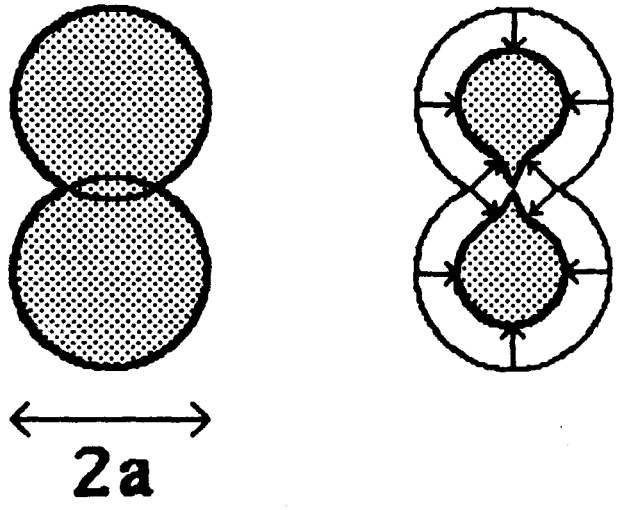

FIGURE 3.10: S is the regularized union of two intersecting spheres of radius a (left), and $\widehat{R}(S) = a$. Shrinking by $r < a$ produces convex vertices, and therefore $\widehat{R}(S\downarrow r) = 0 < (a-r)$.

3.5 BLENDING PROPERTIES OF OFFSETTING

3.5.1 Rounding and filleting

In this section we show that global blending can be modelled by a combination of two offsetting operations. In chapter two we introduced the notion of globally blended solids with constant radius r. A precise definition follows.

DEFINITION 3.5: **The rounded version with radius r of a solid S is**

$$R_r(S) = \bigcup B(Q,r) \qquad \text{with} \qquad B(Q,r) \subset S$$

Rounding replaces all convex edges by blends of specified radius. All concave edges can be filleted by the dual operation.

DEFINITION 3.6: **The filleted version with radius r of a solid S is**

$$F_r(S) = \overline{R_r(\overline{S})}$$

It follows that $R_r(\overline{S}) = \overline{F_r(S)}$ and $F_r(\overline{S}) = \overline{R_r(S)}$.

To prove that the blending operations defined above correspond to combinations of offsetting operations we shall use the following properties.

PROPERTY 3.32: $S = A{\uparrow}r \;\Rightarrow\; A \subset S{\downarrow}r$

PROOF: Let $S = A{\uparrow}r$ and let P be any point in A. By definition 3.1, $\forall R \in \overline{S}$, $\|P-R\| > r$. Therefore $P \notin \overline{S}{\uparrow}r$, and it follows that $P \in \overline{\overline{S}{\uparrow}r} = S{\downarrow}r$ by definition 3.3.

PROPERTY 3.33: $S = A{\downarrow}r \;\Rightarrow\; S{\uparrow}r \subset A$

PROOF: Complementing $S = A{\downarrow}r$ yields $\overline{S} = \overline{A{\downarrow}r} = \overline{A}{\uparrow}r$ by property 3.22. It follows, by property 3.32, that $\overline{A} \subset \overline{S}{\downarrow}r$. By complementing both sides we obtain $\overline{\overline{S}{\downarrow}r} \subset A$, and, by definition 3.3 $S{\uparrow}r \subset A$.

PROPERTY 3.34: $S{\downarrow}r{\uparrow}r \subset S \subset S{\uparrow}r{\downarrow}r$

PROOF: Let $X = S{\uparrow}r$. By property 3.32 $S \subset X{\downarrow}r$ and therefore

$$S \subset S{\uparrow}r{\downarrow}r$$

Replacing S with $\overline{S}$ we obtain $\overline{S} \subset \overline{S}{\uparrow}r{\downarrow}r$. Complementing this relation yields $\overline{\overline{S}{\uparrow}r{\downarrow}r} \subset S$. Since $\overline{\overline{S}{\uparrow}r{\downarrow}r} = \overline{\overline{S}{\uparrow}r}{\uparrow}r = S{\downarrow}r{\uparrow}r$, by property 3.22 and definition 3.3 it follows that

$$S{\downarrow}r{\uparrow}r \subset S$$

PROPERTY 3.35: $R_r(S) = S{\downarrow}r{\uparrow}r$ and $F_r(S) = S{\uparrow}r{\downarrow}r$

PROOF: Let $B(Q,r) \subset S$. $\forall P \in \overline{B(Q,r)}$, $\|P-Q\| > r$ and, by definition 3.1, $Q \notin \overline{B(Q,r)}{\uparrow}r$. Since $\overline{S} \subset \overline{B(Q,r)}$ and since expanding preserves set inclusion, $Q \notin \overline{S}{\uparrow}r$, and therefore $Q \in \overline{\overline{S}{\uparrow}r} = S{\downarrow}r$. By property 3.1 it follows that $B(Q,r) \subset S{\downarrow}r{\uparrow}r$ and therefore:

$$B(Q,r) \subset S \quad \Rightarrow \quad B(Q,r) \subset S{\downarrow}r{\uparrow}r$$

By definition 3.5 it follows that $R_r(S) \subset S{\downarrow}r{\uparrow}r$. To prove the reverse inclusion we consider a point $P \in S{\downarrow}r{\uparrow}r$. By definition 3.1, $\exists Q \in S{\downarrow}r$ such that $\|P-Q\| \leq r$. For such Q we have $P \in B(Q,r)$, and also, by property 3.1 $B(Q,r) \subset S{\downarrow}r{\uparrow}r$. From property 3.34, it follows that $B(Q,r) \subset S$, and therefore:

$$P \in S{\downarrow}r{\uparrow}r \quad \Rightarrow \exists Q \; : \; P \in B(Q,r) \subset S$$

which, by definition 3.5, is equivalent to $S{\downarrow}r{\uparrow}r \subset R_r(S)$. Combining both inclusions, one obtains $R_r(S) = S{\downarrow}r{\uparrow}r$. $F_r(S) = S{\uparrow}r{\downarrow}r$ can be derived from $R_r(S) = S{\downarrow}r{\uparrow}r$ by applying definition 3.6 and property 3.22.

From the above properties it follows clearly that

$$R_r(S) \subset S \subset F_r(S)$$

3.5.2 Regularity of blended solids

PROPERTY 3.36: **For S open: $S{\uparrow}r$, $S{\downarrow}r$, $R_r(S)$, and $F_r(S)$ are open.**

PROOF: Let S be an open set. By definition 3.1, $\forall P \in S\uparrow r$, $\exists Q \in S$ such that $\|P - Q\| \leq r$. By definition of open sets, $\forall Q \in S$, $\exists \epsilon > 0$ such that there is an open ball of center Q and radius 2ϵ in S, and therefore $B(Q,\epsilon) \subset S$. Let us consider any point R of $B(P,\epsilon)$. First notice that since $\|P - Q\| \leq r$ and $\|P - R\| \leq \epsilon$, we have $\|R - Q\| \leq r + \epsilon$. Then consider the point

$$T = Q + \epsilon \frac{R - Q}{\|R - Q\|}$$

T is in $B(Q,\epsilon)$ and therefore in S. We have

$$R - T = (\|R - Q\| - \epsilon)\frac{R - Q}{\|R - Q\|}$$

and $\|R - T\| = \|R - Q\| - \epsilon \leq r$. Therefore $R \in S\uparrow r$. We proved that for any point $P \in S\uparrow r$, $\exists \epsilon > 0$ such that $B(P,\epsilon) \subset S\uparrow r$, and therefore $S\uparrow r$ contains an open neighborhood around any of its points. It follows that $S\uparrow r$ is open.

For open S, $\overline{S}$ is closed, therefore, by property 3.14, $\overline{S}\uparrow r$ is closed and $\overline{\overline{S}\uparrow r}$ is open. Since, by definition 3.3, $\overline{\overline{S}\uparrow r} = S\downarrow r$, it follows that $S\downarrow r$ is open for open S. We proved that expanding and non–regularized shrinking preserve the property of being open; consequently, for open S, $R_r(S)$ and $F_r(S)$ are also open.

PROPERTY 3.37: **For S closed: $S\uparrow r$, $S\downarrow r$, $R_r(S)$, and $F_r(S)$ are closed.**

PROOF: By definition 3.3 $S\downarrow r = \overline{\overline{S}\uparrow r}$. For S closed, $\overline{S}$ is open and by property 3.36 $\overline{S}\uparrow r$ is open and therefore $\overline{\overline{S}\uparrow r}$ is closed. By property 3.14 $S\uparrow r$ is also closed. Combining these two results implies that, for closed S, $R_r(S)$ and $F_r(S)$ are also closed.

PROPERTY 3.38: **$R_r(S)$ is regular for closed S.**

PROOF: Let S be a closed set. Then by property 3.37 $S\downarrow r$ is closed and by property 3.19 $R_r(S)$ is regular.

PROPERTY 3.39: $S = R_r(S) \Leftrightarrow \exists A : S = A\uparrow r$ **and** $S = F_r(S) \Leftrightarrow \exists A : S = A\downarrow r$.

PROOF: Let $S = A{\uparrow}r$. By property 3.32, $A \subset S{\downarrow}r$, and since expanding preserves inclusion (property 3.21), $A{\uparrow}r \subset S{\downarrow}r{\uparrow}r$, which implies that $S \subset R_r(S)$. But, by property 3.34, $R_r(S) \subset S$, and therefore

$$\exists A : S = A{\uparrow}r \quad \Rightarrow \quad S = R_r(S)$$

On the other hand, suppose that $S = R_r(S)$; then $S = S{\downarrow}r{\uparrow}r$ by property 3.35. Choosing $A = S{\downarrow}r$ yields $S = A{\uparrow}r$. It follows that:

$$S = R_r(S) \quad \Leftrightarrow \quad \exists A : S = A{\uparrow}r$$

Applying this equivalence to complements of S and A, we obtain:

$$\overline{S} = R_r(\overline{S}) \quad \Leftrightarrow \quad \exists \overline{A} : \overline{S} = \overline{A}{\uparrow}r$$

which, by definition 3.6 and property 3.22 is equivalent to

$$S = F_r(S) \quad \Leftrightarrow \quad \exists A : S = A{\downarrow}r$$

PROPERTY 3.40: For $a \leq r$, $S = R_r(S)$ implies $S = R_a(S)$ and $S = F_r(S)$ implies $S = F_a(S)$.

PROOF: The property is true for $a = r$. Let $r = a + b$ with $b > 0$ and let $R_r(S) = S$. By property 3.39 $\exists A$ such that $S = A{\uparrow}r$, and therefore $S = A{\uparrow}(b + a)$. By property 3.28 $S = (A{\uparrow}b){\uparrow}a$, which by property 3.39 implies that $S = R_a(S)$. Now let $S = F_r(S)$. From definition 3.6, it follows that $\overline{S} = R_r(\overline{S})$, and we just showed that, for $a \leq r$, this implies that $\overline{S} = R_a(\overline{S})$, which is equivalent to $S = F_a(S)$.

Since, by property 3.35, $R_r(S) = S{\downarrow}r{\uparrow}r$ and $F_r(S) = S{\uparrow}r{\downarrow}r$, for any $r > 0$, $\exists A$ such that $R_r(S) = A{\uparrow}r$ and $\exists B$ such that $F_r(S) = B{\downarrow}r$, it follows by property 3.39 that rounding and filleting are idempotent operations:

$$R_r\Big(R_r(S)\Big) = R_r(S) \text{ and } F_r\Big(F_r(S)\Big) = F_r(S)$$

which implies by property 3.40 that rounding $R_r(S)$ or filleting $F_r(S)$ with a radius $a \leq r$ has no effect.

3.5.3 Maximum convexity and concavity

To analyze the effect of rounding and filleting we need a measure of the *maximum concavity* and *convexity* of a solid. Intuitively the *radius of maximum convexity* $\widehat{R}(S)$ of a solid S corresponds to the radius of the largest ball that can reach all points of S, while remaining in S. Similarly the *radius of maximum concavity* $\breve{R}(S)$ of a solid S corresponds to the radius of maximum convexity of $\overline{S}$.

Because $R_r(S)$ is the region swept by a ball that remains in S, we can define $\widehat{R}$ formally as follows.

DEFINITION 3.7: **The *radius of maximum convexity* $\widehat{R}(S)$ and the *radius of maximum concavity* $\breve{R}(S)$ of a solid S are defined by:**

$$\widehat{R}(S) = \sup_{S=R_r(S)} r \quad \text{and} \quad \breve{R}(S) = \sup_{S=F_r(S)} r$$

Since $S = R_r(S) \Leftrightarrow \overline{S} = F_r(\overline{S})$, we have

PROPERTY 3.41: $\breve{R}(S) = \widehat{R}(\overline{S})$

PROPERTY 3.42: $r < \widehat{R}(S) \Rightarrow R_r(S) = S$ **and** $r < \breve{R}(S) \Rightarrow F_r(S) = S$.

PROOF: By definition 3.7 $\forall \epsilon > 0$, $\exists a > \widehat{R}(S) - \epsilon$ such that $S = R_a(S)$. By property 3.39 this implies that $\exists A$ such that $S = A{\uparrow}a$. Let $\epsilon = \widehat{R}(S) - r$. It follows that $a > r$. Replacing a with $b+r$ yields $S = A{\uparrow}(b+r)$, which by property 3.28 is $S = (A{\uparrow}b){\uparrow}r$. Replacing $A{\uparrow}b$ with B yields $\exists B$ such that $S = B{\uparrow}r$, which by property 3.39 is equivalent to $S = R_r(S)$. Applying this result to $\overline{S}$ yields $r < \widehat{R}(\overline{S}) \Rightarrow R_r(\overline{S}) = \overline{S}$, which is equivalent to $r < \breve{R}(S) \Rightarrow F_r(S) = S$.

PROPERTY 3.43: $\left(\widehat{R}(S{\uparrow}r) \geq \widehat{R}(S) + r\right)$ **and** $\left(\breve{R}(S{\downarrow}r) \geq \breve{R}(S) + r\right)$.

PROOF: Let us suppose that $\widehat{R}(S{\uparrow}r) < \widehat{R}(S) + r$. It follows that $\exists a > 0$ such that $a < \widehat{R}(S)$ and $\widehat{R}(S{\uparrow}r) < a + r$. For instance we can pick $a = \left(\widehat{R}(S{\uparrow}r) + \widehat{R}(S) - r\right)/2$.

$$\begin{aligned}
a < \widehat{R}(S) &\Rightarrow R_a(S) = S \text{ by property 3.42} \\
&\Rightarrow \exists A : S = A{\uparrow}a \text{ by property 3.39} \\
&\Rightarrow \exists A : S{\uparrow}r = A{\uparrow}(a+r) \text{ by property 3.28} \\
&\Rightarrow S{\uparrow}r = R_{a+r}(S{\uparrow}r) \text{ by property 3.39} \\
&\Rightarrow \widehat{R}(S{\uparrow}r) \geq a + r \text{ by definition 3.7}
\end{aligned}$$

which contradicts $\widehat{R}(S\uparrow r) < a + r$. Exploiting this result we obtain:

$$\begin{aligned}\breve{R}(S\downarrow r) &= \widehat{R}(\overline{S}\uparrow r) \text{ by properties 3.41 and 3.22}\\ &\geq \widehat{R}(\overline{S}) + r \text{ proven above}\\ &\geq \breve{R}(S) + r \text{ by property 3.41}\end{aligned}$$

Expanding by r increases $\widehat{R}$ by at least r. The effect on $\widehat{R}$ of shrinking by r cannot be so easily predicted. Figure 3.10 shows that shrinking by r can reduce $\widehat{R}$ by more than r.

3.5.4 Alternative blending operators

The filleting operator of definition 3.6 is the dual — with respect to the non-regularized complement — of the rounding operator of definition 3.5, and therefore properties of $F_r(S)$ can be simply derived from properties of $R_r(S)$. Unfortunately, the operator F_r has two drawbacks:

- when applied to regular sets, F_r does not always produce regular sets (see Figure 3.11);
- in limit cases F_r produces counter-intuitive results (see Figure 3.12). Such cases occur when the filleting ball of radius r outside of S cannot reach some cavity that any ball of radius $a < r$ could reach. Hence rounding a regular solid is not equivalent to filleting its regularized complement.

To guarantee regularity and achieve intuitively expected results in the limit cases of Figure 3.12, we define a modified filleting operator F'_r:

DEFINITION 3.8: $F'_r(S) = \overline{R_r(\overline{S}^*)}^*$.

For any closed set S, $F'_r(S)$ is the closure of the set obtained by filleting the interior of S:

PROPERTY 3.44: **For S closed, $F'_r(S) = k(F_r(iS))$.**

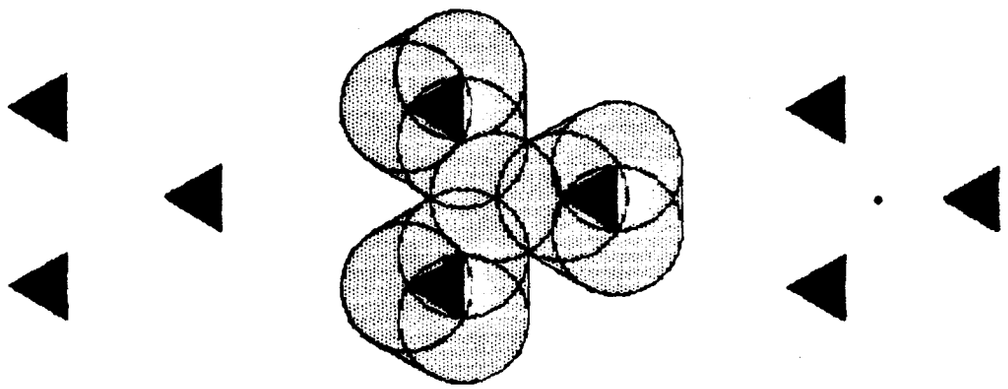

FIGURE 3.11: The regular solid S (left) is first expanded to $S\uparrow r$ (center) then shrunk to $S\uparrow r\downarrow r$ (right). The result contains an isolated point and therefore is not regular.

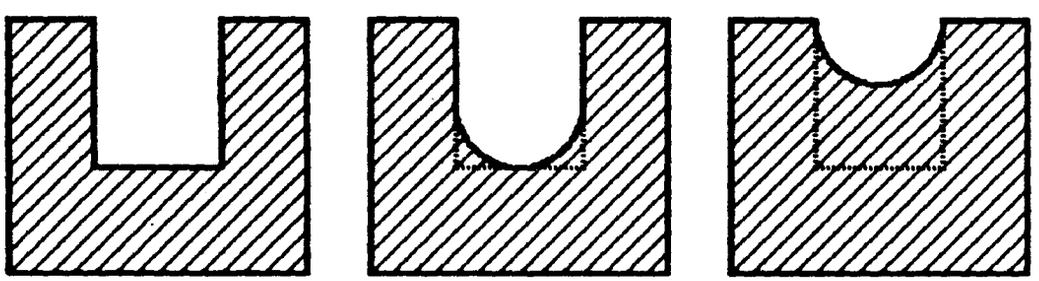

FIGURE 3.12: Filleting the solid S (left) produces $F_r(S)$ (right). A presumably more useful result can be obtained by $F'_r(S)$ (center).

PROOF:

$$
\begin{aligned}
F'_r(S) &= \overline{R_r(\overline{S}^*)}^* \text{ by definition 3.8} \\
&= \overline{\overline{S}^*\downarrow r\uparrow r}^* \text{ by property 3.35} \\
&= \overline{\overline{\overline{S}^*\uparrow r\downarrow r}}^* \text{ by definition 3.3 and property 3.22} \\
&= ki\left(\overline{\overline{\overline{(ki\overline{S})}\uparrow r\downarrow r}}\right) \\
&= ki\left(\overline{(ki\overline{S})}\uparrow r\downarrow r\right) \\
&= ki\left(\overline{\overline{ik\overline{S}}\uparrow r\downarrow r}\right) \text{ since } ki\overline{S} = k\overline{kS} = \overline{ikS} \\
&= ki\big((ikS)\uparrow r\downarrow r\big) \\
&= ki\big((iS)\uparrow r\downarrow r\big) \text{ since } S \text{ is closed} \\
&= k\big((iS)\uparrow r\downarrow r\big) \text{ by property 3.36 for open } iS
\end{aligned}
$$

Because it is defined in terms of regularized complements, modified filleting always produces regular sets.

Both rounding and modified filleting operations use non–regularized shrinking which produces open sets as intermediate results and forces one to deal with open sets in the modeller, and specifically to support expanding operations on open sets. This is not difficult to do because most computations on offsets rely on point/solid classification (as shown later), which is actually easier for open sets.

For instance, given a point P and a set S, we wish to find whether P is in $i(S\uparrow r)$, on $\partial(S\uparrow r)$, or in $i(\overline{S\uparrow r})$. For $d(P,S) \neq r$, the classification of P with respect to $S\uparrow r$ is simple and is independent of S being closed or open. Let us suppose now that $d(P,S) = r$.

PROPERTY 3.45: **For S open, $d(P,S) = r \Leftrightarrow P \in \partial(S\uparrow r)$.**

PROOF: Let $d(P,S) = r$. First let us prove that $P \in k(S\uparrow r)$:

$$
\begin{aligned}
d(P,S) = r &\Rightarrow d(P,kS) = r \text{ by property 3.6} \\
&\Rightarrow P \in (kS)\uparrow r \text{ by property 3.12} \\
&\Rightarrow P \in k(S\uparrow r) \text{ by property 3.16}
\end{aligned}
$$

Now let us prove that $P \notin i(S\uparrow r)$: By property 3.8, for open S, $\forall Q \in S$, $Q \in iS$ and $\|P - Q\| > r$, therefore $P \notin S\uparrow r$, and since $i(S\uparrow r) \subset S\uparrow r$, $P \notin i(S\uparrow r)$. It follows that

$$P \in k(S\uparrow r) - i(S\uparrow r) = \partial(S\uparrow r)$$

The inverse implication is true for all S by property 3.17.

It follows that, for open S, the classification of P with respect to $S\uparrow r$ can be deduced from $d(P,S)$.

On the other hand, let us suppose that S is closed and that $d(P,S) = r$. We know that $P \in S\uparrow r$, but we do not have enough information to decide whether $P \in i(S\uparrow r)$ or $P \in \partial(S\uparrow r)$. These two possibilities are demonstrated in Figure 3.13. In boundary evaluation algorithms described in chapter five, we must distinguish between points in $i(S\uparrow r)$ and points on $\partial(S\uparrow r)$. Such a distinction requires the classification of the neighborhood of the point with respect to $S\uparrow r$. Neighborhoods and their classification will be discussed later.

We conclude that point membership classification with respect to the expanded version of a set S is simpler when S is open than when S is closed.

Nevertheless dealing with non–regular sets requires care in the implementation of a modeller, and therefore, instead of the blending operators discussed above, we propose to support regularized blending which ensures regularity and is defined as follows:

DEFINITION 3.9: **The regularized rounding operator is $R_r^*(S) = S\downarrow^* r\uparrow r$, and the regularized filleting operator is $F_r^*(S) = S\uparrow r\downarrow^* r$.**

From property 3.22 it follows that $F_r^*(S) = \overline{R_r^*(\overline{S}^*)}^*$.

In limit cases regularized blending will produce intuitively surprising results (Figure 3.14), but for common objects, regularized and non–regularized blending produce the same result. Therefore, for simplicity of exposition, we focus in the sequel of this thesis on these regularized blending operators. In essence this reduces blending to regularized offsetting. Therefore, all that we have to discuss is how to support expanding and regularized shrinking operations on regular sets.

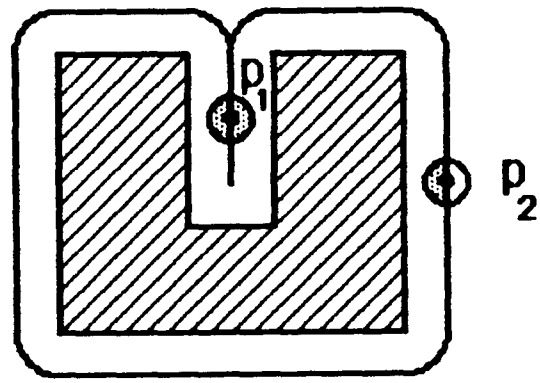

FIGURE 3.13: Both points P_1 and P_2 are at distance r from S. The point P_1 is in $i(S\uparrow r)$, but P_2 is on $\partial(S\uparrow r)$. To distinguish between "in" and "on" points, neighborhoods of such points must be classified with respect to $S\uparrow r$.

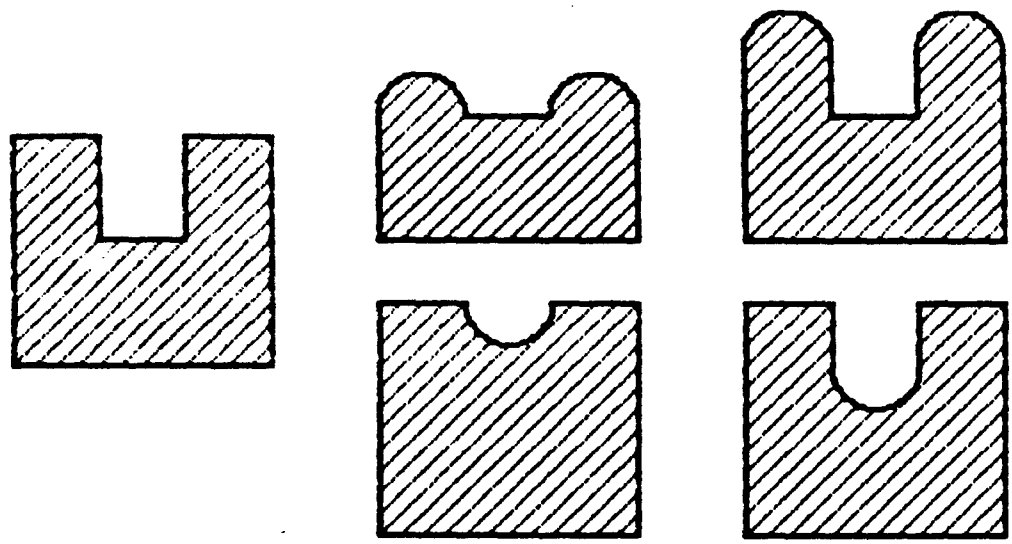

FIGURE 3.14: Given the solid S (left), regularized blending produces counter–intuitive results, $R^*_r(S)$ (center–top) and $F^*_r(S)$ (center–bottom). More intuitive results are obtained by non–regularized rounding $R_r(S)$ (right–top) and modified filleting $F'_r(S)$ (right–bottom).

CHAPTER IV

BOUNDARIES OF OFFSET SOLIDS

Explicit representations for the topological boundaries of solids play an important role in solid modelling. Multiple–representation solid modellers such as PADL–2 contain boundary representations, which are used to support calligraphic displays and other applications.

This chapter discusses mathematical properties of the boundaries of offset solids with a view to computational applications. In particular, we construct *supersets* of the boundary of an offset solid because representation conversion algorithms that evaluate boundaries [Requicha 85] operate on such supersets. (Boundary evaluation for offset solids is discussed in the following chapters.)

4.1 CONSTANT–DISTANCE SETS

DEFINITION 4.1: **The set of all points at distance r from a set S is called a "constant–distance set" and denoted $B_r(S)$.**

First we show that to obtain a superset of the boundary of an offset solid, one need only consider expanding transformations. In chapter three (property 3.17) we showed that $\partial(S\uparrow r)$ is contained in the set of points that are at distance r from S, therefore:

PROPERTY 4.1: $\partial(S\uparrow r) \subset B_r(S)$.

By definition, the boundary of any set S is equal to the boundary of its complement $\overline{S}$; and for regular S, to the boundary of $\overline{S}^*$ [Requicha 78, property 3.3.3]. These facts are used in the following proofs.

PROPERTY 4.2: $\partial(S\downarrow r) \subset B_r(\overline{S})$.

PROOF:

$$\begin{aligned}\partial(S\downarrow r) &= \partial\left(\overline{\overline{S}\uparrow r}\right) \text{ by definition 3.3}\\ &= \partial\left(\overline{S}\uparrow r\right)\\ &\subset B_r(\overline{S}) \text{ by property 4.1}\end{aligned}$$

PROPERTY 4.3: **For regular S:** $\partial(S\downarrow^* r) \subset B_r(\overline{S}^*)$.

PROOF:

$$\begin{aligned}\partial(S\downarrow^* r) &= \partial\left(\overline{\overline{S}^*\uparrow r}^*\right) \text{ by definition 3.2}\\ &= \partial\left(\overline{S}^*\uparrow r\right) \text{ since } \overline{S}^*\uparrow r \text{ is also regular}\\ &\subset B_r\left(\overline{S}^*\right) \text{ by property 4.1}\end{aligned}$$

Observe that, $d(P,S) = r > 0$ implies $d(P,\partial S) = r$ (property 3.10), and therefore:

PROPERTY 4.4: **For any set S, $B_r(S) \subset B_r(\partial S)$**

However, $B_r(\partial S)$ may include points that are not in $B_r(S)$; for example, the points of $B_r(\partial S)$ that lie in S are not in $B_r(S)$.

Since for any solid S, $\partial S = \partial\overline{S}$, and for regular S, $\partial S = \partial\overline{S}^*$, combining properties 4.1, 4.2, 4.3, and 4.4 one obtains:

Although properties 4.1 to 4.3 give us sharper results, i.e., smaller supersets, in the remainder of this chapter we shall investigate supersets which are larger than $B_r(\partial S)$, but are easier to deal with computationally.

4.2 SMOOTH FACES AND SINGULARITIES

The boundaries of objects of interest in solid modelling may be decomposed into three disjoint sets: the *smooth faces*, the *singular curves*, and the *singular points*, defined as follows.

- A *smooth face* is a connected set of points Q of ∂S that have two–dimensional neighborhoods $N_2(Q)$ in ∂S, such that ∂S is G^1 continuous within $N_2(Q)$ [Barsky 84], i.e., it has continuous unit normal.

- A *singular curve* C is a connected set of points Q of ∂S that do not belong to the smooth faces of ∂S but have one–dimensional neighborhoods $N_1(Q)$ in C such that C is G^1 continuous within $N_1(Q)$ [Barsky 84], i.e., it has continuous unit tangent.

- The *singular points* of ∂S are the set of points Q of ∂S that do not belong to the smooth faces or to the singular curves of ∂S.

For a flat–faced polyhedral solid, the smooth faces are the (2–D) interiors of what are usually considered the solid's faces. The singular curves are the edges without their end–points, and the singular points are the vertices. When a solid is curved, however, the singular curves and points typically are a subset of the edges and vertices that appear in the BRep of the solid.

To find a superset for $B_r(\partial S)$ — and thus for $\partial(S\uparrow r)$, $\partial(S\downarrow r)$, and $\partial(S\updownarrow^* r)$ — we argue as follows. Let P be a point at distance r from ∂S. We know from properties 3.7 and 3.9 that the minimum distance is achieved for at least one point Q of ∂S. This point must lie either in a smooth face, or in a singular curve, or be a singular point of S. In the following subsections we analyze each of these cases separately and construct sets of points that are guaranteed to include all points of $B_r(\partial S)$.

4.3 N–OFFSETS

4.3.1 N–offsetting smooth faces

Let P be a point of $B_r(\partial S)$ and suppose that the minimum distance from P to ∂S is achieved at a point Q that belongs to a smooth face F of S.

Given a bi–parametric representation $F(u,v)$ of F, $\|P - F(u,v)\|$ is minimal for $u = u_0$ and $v = v_0$ with $F(u_0,v_0) = Q$. Therefore at $F(u_0,v_0)$, $\partial\|P - F(u,v)\|^2/\partial u = 0$, and $\partial\|P - F(u,v)\|^2/\partial v = 0$. These two conditions imply that P lies on the normal to F at Q. Such a point Q belongs to the *normal projection* of P on F, defined as follows.

DEFINITION 4.2: **The *normal projection* of a point P on a face F is the set of points Q of F for which $(P - Q)$ is orthogonal to F at Q.**

Since $\|P - Q\| = r$, P is generated by displacing a point Q of F by a distance r along the normal to F at Q. The set of all points P generated in this manner is usually a pair of faces "parallel" to F, called the *n–offset* of F.

The n–offset of a smooth surface F is usually defined[3] as a surface *parallel* to F [Willmore 58, p. 116] obtained by offsetting each point Q of F by r along the oriented unit vector $N(F,Q)$ normal to F at Q. (A similar definition is used for n–offsets of *smooth* curves in two dimensions.) Our definition of n–offset for a smooth face is a direct extension of the usual one. We distinguish between *positive* and *negative* n–offsets. Assuming that at each point Q of a smooth face F, $N(F,Q)$ has unit length and predefined orientation, the positive n–offset of F by distance r is:

$$F\|_r^+ = \{P : \exists Q \in F \text{ such that } P = Q + rN(F,Q)\}$$

and the negative n–offset is:

$$F\|_r^- = \{P : \exists Q \in F \text{ such that } P = Q - rN(F,Q)\}$$

The global term "n–offset" denotes the union of the positive n–offset with the negative one:

$$F\|_r = F\|_r^+ \cup F\|_r^-$$

[3] N–offsets near singularities are not defined in the literature.

Given a point Q in F, let Q' and Q'' be the corresponding n–offset points defined by $Q' = Q + rN(F,Q)$ and $Q'' = Q - rN(F,Q)$. The three normals $N(F\|_r^+, Q')$, $N(F\|_r^-, Q'')$, and $N(F,Q)$ are parallel [Willmore 58].

Let k_a and k_b be the principal curvatures of F at Q. The principal curvatures k'_a and k'_b of $F\|_r$ at Q' and Q'' are [Faux 79]:

$$k'_a = \frac{k_a}{1 \pm rk_a} \quad \text{and} \quad k'_b = \frac{k_b}{1 \pm rk_b}$$

Thus, n–offsetting by r transforms points of F where one of the principal radii of curvature equals r, into singular points of $F\|_r$, at which the curvature becomes infinite. Such points lie on the concave side of F. It follows that, even though F is smooth, $F\|_r$ can have singularities or can cross itself. For instance, the negative n–offset by r of a spherical face of radius r, whose normal is oriented towards the exterior, is not a surface but a single point, the center of the sphere.

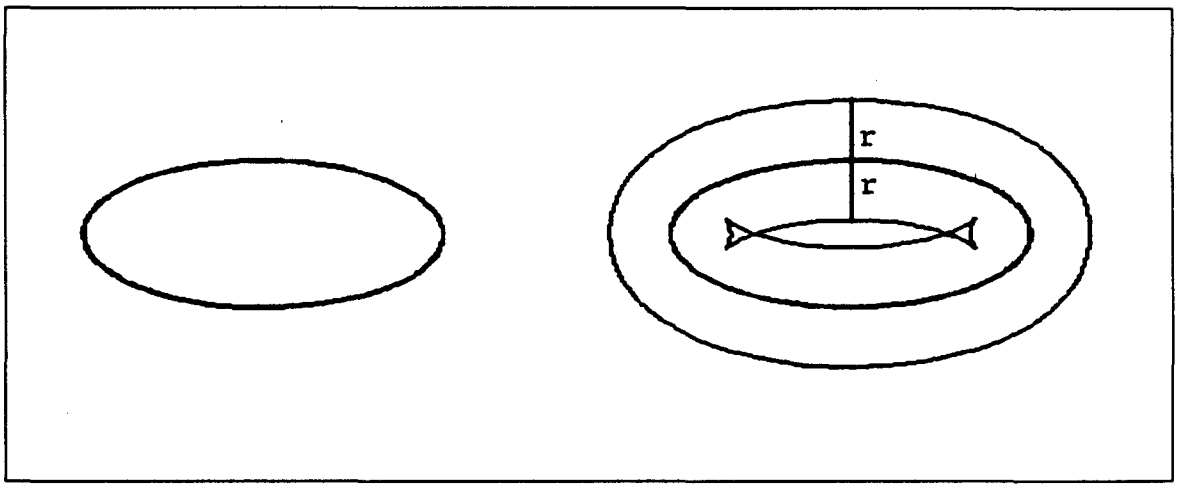

FIGURE 4.1: The ellipse C (left) is n–offset by a distance r. Each point of C is moved by r along the normal. The n–offset curves $C\|_r$ (right) are not conics. The interior curve is self–intersecting.

Willmore also shows that the curves of $F\|_r$ that correspond to *lines* mensions. The set of circles and lines is closed under n–offsetting, but the set of conics is not (see Figure 4.1 for a counterexample).

4.3.2 N–offsetting singular curves

Let P be a point of $B_r(\partial S)$ and suppose that the minimum distance from P to ∂S is achieved at a point Q that belongs to a singular curve C of S.

Given a parameterization $C(t)$ of C, $\|P - C(t)\|$ is minimal for some $t = t_0$ such that $C(t_0) = Q$. Therefore $d\|P - C(t)\|^2/dt = 0$ for $t = t_0$. This equation implies that $(P-Q)\cdot T(t_0) = 0$, where $T(t_0)$ is the tangent to C at Q. $T(t_0)$ is well defined since C is a singular curve of S. Thus, Q belongs to the *normal projection* of P on C, defined as follows.

DEFINITION 4.3: **The *normal projection* of a point P on a curve C is the set of points Q of C, for which $(P - Q)$ is orthogonal to the tangent to C at Q.**

Together with $\|P - Q\| = r$, the requirement that Q belong to the normal projection of P on C implies that P belongs to the circle of radius r centered at Q and lying in the plane normal to C at Q. It follows that the set of points P, for which the minimal distance to ∂S is equal to r and is attained in at least one point Q of a singular curve C of S, is contained in the sweep of a circular cross–section of radius r along C. For simplicity we shall refer to such a set of points by the term canal face of radius r and spine C. Such canal faces are part of *canal surfaces of spheres of constant radius* [Monge 1849] mathematically defined and analyzed in Appendix A. To use a consistent terminology we shall say that canal faces are *n–offsets* of their spines.

4.3.3 N–offsetting singular points

Finally, let P be a point of $B_r(\partial S)$ and suppose that the minimum distance from P to ∂S is achieved at a singular vertex Q of S. This implies that P belongs to the sphere[4] of radius r centered at Q.

[4] "Sphere" stands here for spherical surface.

4.3.4 N–offsetting a solid's boundary

The previous discussion shows that any point of $\partial(S\uparrow r)$, $\partial(S\downarrow^{*} r)$, or $\partial(S\downarrow r)$, belongs to the superset of $B_r(\partial S)$ defined as the union of:

— n–offsets by r of all smooth faces of ∂S;

— canal faces of radius r around all singular curves of ∂S;

— spheres of radius r centered at all singular points of ∂S.

We refer to this union simply as the "n–offset" of ∂S.

4.3.5 Closure under n–offsetting

Using a set of faces that is closed under n–offsetting is extremely important for the implementation of offset operations in a solid modeller. The faces of an offset solid are subsets of the n–offset of the original solid's boundary, as we have shown in this chapter. Closure under n–offsetting implies that no faces of a new type have to be introduced to support offsetting. We show in the following section that "standard" faces (which lie in planes, cylinders, cones, spheres and tori) supported by current modellers, together with canal faces are closed under n–offsetting. (It is worth noting that the set of general quadrics is not closed under n–offsetting; for example the n–offset of an ellipsoid is not a quadric surface).

4.4 N–OFFSETTING STANDARD FACES

Since n–offsetting is defined in terms of distances and of intrinsic properties of solids, it commutes with rigid motions, and therefore n–offsetting can be studied in any coordinate system. We shall use a parameterization $F(u,v)$ of *face* F of ∂S. We denote by A the surface in which F lies. The set of possible values for (u,v) defines the extent of F in A.

4.4.1 N–offsetting a planar face

Any planar face can be defined in some local coordinates as a bi–parametric patch $F(u,v) = (u,v,0)$ for some set of values (u,v). The normal to F is constant: $N(u,v) = (0,0,1)$. It follows that the n–offset of F is $F(u,v)\|_r = (u,v,\pm r)$, which is composed of two planar faces obtained by translating F by r along the Z–axis.

4.4.2 N–offsetting a spherical face

In its origin–centered coordinate system, a spherical face of radius a can be defined by $F(u,v) = aN(u,v)$. The n–offset is composed of two spherical faces of radius $|a \pm r|$ centered at O: $F(u,v)\|_r = (a \pm r)N(u,v)$.

4.4.3 N–offsetting a conical face

A conical face can be defined by $F(u,v) = vZ + v\frac{a}{\sqrt{1+a^2}}N(u)$, where Z is the unit vector along the Z–axis and a is the tangent of the tip half–angle. The parameter u corresponds to the angle around the Z–axis, and v corresponds to the Z–coordinate of the intersection of the normal to F at $F(u,v)$ with the Z–axis. The normal does not depend on v and is $N(u) = \frac{1}{\sqrt{1+a^2}}(\cos u, \sin u, -a)$. The n–offset is

$$F(u,v)\|_r = vZ + (v\frac{a}{\sqrt{1+a^2}} \pm r)N(u)$$

Replacing v with $w \mp \frac{r\sqrt{1+a^2}}{a}$ yields $F(u,w)\|_r = wZ + \frac{aw}{\sqrt{1+a^2}}N(u) \mp \frac{r\sqrt{1+a^2}}{a}Z$, which is composed of two faces. Each face lies on a version of the original cone translated by $\mp\frac{r\sqrt{1+a^2}}{a}$ along the Z–axis. The correspondence between points on the original face and their n–offset counterparts is defined by the parameter substitution $v = w \mp \frac{r\sqrt{1+a^2}}{a}$.

4.4.4 N–offsetting a canal face

We use a parameterization of canal faces in terms of the spine $C(u)$, which is the smooth space curve along which the center of the circle is swept, and of the normal $N(u,v)$, which is a unit vector orthogonal to the spine at $C(u)$. The canal face is $F(u,v) = C(u) + aN(u,v)$. Its n–offset is

$$F(u,v)\|_r = C(u) + (a \pm r)N(u,v)$$

It is composed of two canal faces, which have the same spine as F, but radii equal to $|a \pm r|$. The resulting faces need not be smooth; they may contain edges of regression analyzed in Appendix A.

Cylinders and tori are special cases of canal faces, and therefore their offsets are respectively cylinders and tori of the same spine, but different radii.

CHAPTER V

REPRESENTATIONAL REQUIREMENTS

Our primary goal is to support offsetting operations in solid modellers that contain both CSG and BReps, and to ensure that the solids that result from offsetting operations are treated like other solids in the modeller, i.e., that they can be displayed, combined by Boolean operations, further offset, and so forth. This section discusses representational requirements for achieving this goal.

5.1 SPECIFICATION

We demonstrated in chapter two how models of solids that contain constant radius rounds and fillets can be specified in terms of simple primitives combined through Boolean operations and blending operations, which take a solid S and a distance r and return a globally filleted solid, $F_r^*(S)$, or a globally rounded solid, $R_r^*(S)$. In chapter three we proved that such *global blending* operations can be modelled by a combination of expanding and regularized shrinking operations: $F_r^*(S) = S{\uparrow}r{\downarrow}^*r$ and $R_r^*(S) = S{\downarrow}^*r{\uparrow}r$.

To provide blending and offsetting capabilities, we develop in the following chapters tools for supporting offset operations in dual–representation solid modellers, in which solids are specified by a sequence of operations applied to sub-solids (typically these operations are Boolean set operations and rigid motions). We propose to add to the specification syntax two *regularized* offsetting operators, which take a solid S and a distance r and return $S{\uparrow}r$ or $S{\downarrow}^*r$.

5.2 REPRESENTATIONS

Solid definitions in current dual–representation modellers are stored in CSG trees, which are automatically converted by the systems into consistent BReps. How should these representations be extended to support offsetting?

5.2.1 Conversion to CSG

The (conceptually) simplest approach is to treat offsetting operations as "volatile commands" that are executed to transform an object and discarded. Execution of the commands should produce both a CSG representation and a BRep for the offset solid that results from the operation. Such an approach has the advantage that standard CSG–oriented algorithms may be used for all applications. We investigated this approach and found that it is difficult to generate automatically CSG representations for offsets of complicated solids, and that the resulting representations are difficult to manipulate algorithmically, although CSG representations for offsets of simple solids, e.g. standard primitives, are easy to generate.

To represent offset solids in CSG we must support primitives of a new type. Property 3.13 shows that the expanded version of a solid S can be expressed as the union of S with the expanded version of the boundary of S. Since the boundary of S is the union of faces, and expanding distributes over the union, we can express the expanded version of S as the union of S with all the expanded versions of the faces of S. Clearly offset operations could be supported in CSG if expanded versions of faces were available.

Computation on CSG often uses the fact that standard primitives can be expressed as intersections of half–spaces, which are bounded by simple surfaces, and for which point–membership classification is simple. The computational complexity of classification against expanded faces is similar to the classification against offset solids. (As we shall see later, it is based on the computation of point/set minimum distance.)

Classification with respect to expanded faces could be simplified if, for a given face F bounded by edges E_i, one could produce a CSG expression for $F{\uparrow}r$ in terms of simple half–spaces. Suppose that the face F is part of

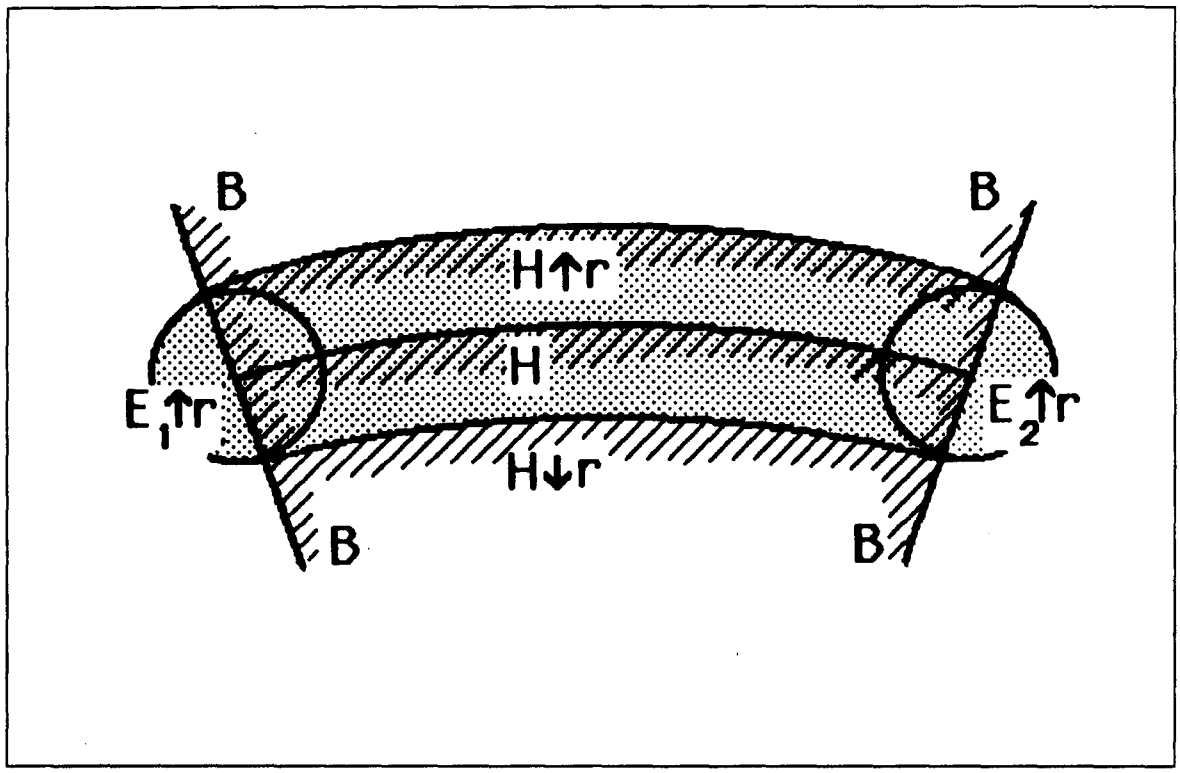

FIGURE 5.1: The expression in CSG of the expanded version of a face uses the region B, bounded by ruled surfaces.

the boundary of a half–space H. Then, as indicated in figure 5.1, $F\uparrow r$ can be expressed in CSG as:

$$F\uparrow r = \left((H\uparrow r \; -^* \; H\downarrow r) \cap^* B \right) \cup^* \left(\bigcup_{all\ i} E_i\uparrow r \right) \quad ,$$

where $E_i\uparrow r$ are regions bounded by canal surfaces of radius r and spine E_i, and where B is a region bounded by ruled surfaces that contain all lines normal to H at points Q of E_i. Such ruled surfaces depend on the edges E_i, as well as on the direction of the vectors normal to H along these edges. This direction generally is not constant along an edge. Therefore, the classification of points with respect to half–spaces bounded by such surfaces is not an easy task, even though these surfaces have a reasonably simple parametric expression. It is not even clear what these half–spaces are, since the ruled surfaces can cross themselves. Besides, such auxiliary surfaces do not contribute to the boundary of offset solids and one should try to avoid introducing unnecessary complexities in the solid modeller.

We would like to point out that this approach is very successful in two dimensions, where the auxiliary surfaces become straight lines. Therefore,

the offset of a two dimensional solid defined in CSG can be easily expressed in CSG, provided that the faces of the solid are closed under n–offsetting (and this is the case for lines and circles).

Instead of trying to express expanded primitives by intersections of some half–spaces, one could introduce new primitives such as $F\uparrow r$, which are defined by a superset of their boundary and by a rule for point membership classification. Such an approach is a special case of our approach to offsetting solids. It might be investigated as an economical tool for specifying fillets and rounds: the user would define fillets by expanding and shrinking union of faces instead of whole sub–solids. In three dimensions the above approach does not really simplify the problem, and "face offsetting" can be made available as a side product, if solid offsetting is supported.

5.2.2 CSGO

We opted for an alternative approach, in which offset solids are represented by an extended form of CSG called *CSG with offsetting*, or simply *CSGO*. Representations in CSGO are trees containing offsetting operators as non–terminal nodes, in addition to the usual rigid motion and Boolean operators.

This decision has non–trivial implications, and amounts essentially to redefining the problem we set out to solve initially. Thus, instead of a CSG/BRep dual modeller, we shall seek to build a CSGO/BRep modeller.

5.3 BOUNDARY REPRESENTATIONS

CSGO trees are a trivial extension of the usual CSG trees, but BReps for offset solids are considerably more complex than those used for modellers that support only the standard quadric and toroidal surfaces. Offsetting introduces new types of surfaces in a modeller — canal surfaces — and, since offset solids may be combined by Boolean operations, edges of intersection between canal surfaces or between canal and standard surfaces may also appear. Therefore we need representations for canal surfaces and for subsets of their intersection curves.

Canal surfaces can be represented indirectly, by their spine curves and radii. (In essence, these are sweep representations [Requicha 80].) However, these representations are computationally awkward because they lead to a nightmare of indirections. Think, for example, of intersecting a curve with a canal surface represented by a radius and a spine, when the spine results from the intersection of two other canal surfaces, which in turn are represented indirectly, and so forth. Our approach is to represent canal surfaces indirectly (and exactly) but also carry an approximate representation (in terms of their radii and of an approximation of their spines), which is used in all the necessary numerical calculations.

The entire curves of intersection between two canal surfaces, or between a canal and a standard surface may be represented indirectly, by using pointers to the surfaces. However, these curves need not be one–manifolds (i.e., they may self–intersect) and therefore a *subset* of a curve cannot be represented simply by the host curve and a pair of endpoints [Requicha 80]. The standard representations for curve segments in solid modelling are based on *parameterizations* of the curves, but we do not know of any general methods for parameterizing the intersections of canal surfaces. These difficulties led us to represent edges through parameterized approximations. We also carry references to the intersecting surfaces, which may be used to refine the approximations when needed.

Edge approximation is a crucial issue in our approach. We devised a new approximation technique, called *piecewise constant curvature* approximation, which is especially well–suited for solid modelling. It is presented in chapter eight. Edges are approximated by *piecewise circular curves*, and the canal surfaces that result from n–offsetting such edges are approximated by smoothly joined pieces of tori and cylinders. (In the sequel we use the abbreviation *PCC* both as a noun to denote "piecewise circular curve", and as an adjective to mean "piecewise constant curvature".)

In summary, our approach is to represent in BReps the new surfaces and edges introduced by offsets both indirectly (exactly) and by PCC approximations.

CHAPTER VI

COMPUTATION IN CSGO

A large class of blended and offset solids can be represented in CSGO (Constructive Solid Geometry with Offsetting operations) by a set of standard primitives transformed by rigid motions and combined through Boolean and offset operators. CSGO provides powerful and flexible specification capabilities, but the computation of boundaries and of other properties of solids defined in CSGO raises many algorithmic issues, which are analyzed in this chapter.

Today's solid modellers have facilities for generating calligraphic and shaded displays of the represented objects, for computing mass properties, and for evaluating the boundaries of objects defined by Boolean operations. What algorithms are needed to support such facilities in a CSGO/BRep system? All applications are based on set membership classification algorithms that take a set e and a solid S and, return the subsets of e that lie in the interior of S, on the boundary of S, or in the complement of S.

Throughout this chapter we assume that the BRep of a solid S contains the following information:

- A sufficient set F of *patches* (subsets of standard surfaces), whose union contains the boundary of S. Patches are supersets of the smooth faces of the solid and are bounded by simple edges (segments of lines or circles), which also lie in curvature lines and correspond to constant parameter curves in a simple parameterization of the host surface that contains the patch. In CSG, patches correspond to faces of primitives. We use patches (instead of true faces) because they have a simple explicit representation. In CSGO

we use piecewise toroidal and cylindrical approximations for canal faces. It follows that no additional patch–type is necessary for integrating offsetting operations in CSG.

To each patch we associate a normal vector, such that if the patch contains a face of the solid, the normal points to the exterior of the solid.

- The set E of edges (singular curves) of the solid. We shall use PCC approximations for all edges of the represented solids. PCC's are discussed in Chapters seven and eight.

- The set V of vertices (singular points) of the solid. Vertices can bound the edges, but also can be isolated singularities, such as the apex of a cone.

As we shall see, our algorithms do not use connectivity information, such as face/edge incidence relations, normally contained in a BRep.

We propose below a high level outline of the major algorithms necessary to support standard applications and boundary evaluation in CSGO. Our simplified implementation of these algorithms is described in Chapter ten.

6.1 CURVE MEMBERSHIP CLASSIFICATION

6.1.1 Motivation

Calligraphic displays usually are generated in solid modellers by applying standard computer graphics techniques to edge lists, or to BReps. (To display curved surfaces, one also needs *profile edges*, whose generation will be discussed later.) Shaded displays, however, often are generated directly from CSG by *ray casting*. The basic ray casting algorithm [Roth 82] can be extended to dual CSGO–BRep modellers. We propose the following simplified formulation in pseudo–PASCAL.

```
procedure SHADE (S);
begin
for each pixel in screen
      do begin
         e:=ray through view point and pixel;
         e_in S:=TRIM_EDGE(e,S,in);
         if e_in S ≠ ∅
            then begin
                    P:=closest point of e_in S to view point;
                    I:=intensity of light reflected by S at P;
                    intensity[pixel]:=I;
                    end;
         end;
end;
```

In words: cast a ray e between the viewpoint and each pixel in the screen, find the first point P where the ray enters the solid S. If P exists, use an illumination model to compute the appropriate intensity I, and write it onto the screen.

The essential procedure in this algorithm is *TRIM_EDGE*, which takes an "edge" (i.e., a curve segment) e, the dual CSGO–BRep representation of a solid S, and a flag *IBO* (here $IBO = in$) indicating whether the result R should be $e_{in}S$, $e_{on}S$, or $e_{out}S$, which are respectively, the subsets of e that are inside, on the boundary of, and outside the solid S [Tilove 80, Requicha 85]. *TRIM_EDGE* will be also used below for boundary evaluation (with $IBO = on$).

There are many methods for mass–properties calculations, but the most commonly used in dual–representation modellers are based on ray casting or cell classification [Lee 82]. The ray casting algorithm involves the *TRIM_EDGE* procedure mentioned above. Cell classification uses a simplified version of the *CLASS_POINT* procedure described later.

6.1.2 Curve classification algorithm

In CSG, edge classification can be done through divide and conquer methods, by classifying the edge against primitives and by combining the results [Tilove 80, Requicha 85]. Unfortunately, we do not know how to infer the classification of an edge e with respect to $S\uparrow r$ or $S\updownarrow^* r$ from the classification of e with respect to S. Therefore, to extend the divide and conquer method to CSGO, we need an algorithm for classifying edges with respect to offset solids.

Such an algorithm, outlined below, can be integrated in divide and conquer methods and used only at offset nodes, but for the sake of simplicity we present it as an independent algorithm, which classifies an edge e with respect to a solid S represented by a CSGO tree. We assume that the set F of patches (that include the faces of S) is available, and also that the BReps of all sub–solids used in the expression of S are available. (Edges and vertices of S are not needed to classify an edge with respect to S, and therefore this algorithm is well suited for incremental boundary evaluation, described later on.)

The procedure *TRIM_EDGE* takes an edge e and first breaks it into segments as follows. It computes the set I of intersection–points of e with the patches of S. Then the set I is sorted along e and coincident points are merged. Segments bounded by two distinct intersection points are guaranteed not to cross the boundary of S. Classification of each edge segment is obtained by classifying the mid–point (or any other convenient point) of the interior of the segment. Segments whose classification differs from the desired one (specified by the value of *IBO*) are discarded. In our pseudo–PASCAL formulation the + operator denotes the "union" when applied to sets.

```
function TRIM_EDGE(e,S,IBO):R;
begin
I:=∅;
for each face f of F do I:=I+CURVE_INT_FACE(e,f);
Sort I along e and merge coincident points;
M:=segments of e bounded by consecutive intersections of I;
R:=∅;
for each segment m of M
```

```
    do begin
        P:=any point of the interior of m;
        (cls,nbh):=CLASS_POINT(P,S);
        if cls=IBO then R:=R+{m};
        end;
merge adjacent segments of R;
end;
```

Our algorithm uses two procedures: *CURVE_INT_FACE* (discussed below), which computes the intersection of a curve (or edge) with a patch, and *CLASS_POINT* (discussed in Section 6.2), which computes point membership classification.

The procedure *CURVE_INT_FACE* computes the intersections of a edge e with a patch f. This is done as follows. First compute the set J of intersection points of e with A, the host surface of f. (Curve/surface intersection is disussed below in Section 6.1.3.) Then classify points of J against f and discard points that are not on f. Point/face classification is expensive for true faces of a solid (such faces might be bounded by many complicated edges), but point/patch classification is straightforward. For instance, for a cylindrical patch f of radius r (around the Z–axis) limited by the two planes $z = 0$ and $z = l$, we check whether a point j that lies on the host surface of f is also in f by simply checking whether $0 \leq j_z \leq l$.

One could replace patches by their host surfaces in the BRep, thus avoiding point/face classification. But our algorithm would generate additional intersection–points, and therefore the edge would be split into more segments, whose mid–points would have to be classified against the whole solid. As we shall see, point/solid classification is in general much more expensive than point/patch classification.

6.1.3 Curve/surface intersection

The standard method for computing intersections of a curve C with a surface A uses an implicit equation $A(x,y,z) = 0$ for A (for instance, for a sphere $\|P-C\|^2 - r^2 = 0$) and a parametric expression for C (for instance $C(t) = (x(t), y(t), z(t))$. The intersection is the set of points $C(t)$ for which the implicit equation of A is satisfied. To compute such points, we replace the coefficients x, y, z in the implicit equation of A with the parametric

expressions of the coefficients of $C(t)$ and solve $A\Big(x(t), y(t), z(t)\Big) = 0$ for t. The resulting t values are used to generate the corresponding intersection points on C.

Boundaries of solids represented in CSGO may contain faces that are part of canal faces, and therefore we need procedures to intersect curves with canal faces. In general, no implicit equation is known for a canal face, and therefore we cannot use the standard method for intersecting a parametric curve with a canal face.

A canal face of radius r and spine S, can be described by a set of non–linear equations. Iterative methods can be used [Morgan 81] to solve the intersection problem. Morgan defines the spine S of the canal face as the intersection of two faces, described by implicit equations $S_1(P) = 0$ and $S_2(P) = 0$.

If a point $Q = C(t)$ on the curve is also on the canal face (i.e., is an intersection–point), then the distance from Q to the spine S of the canal face is r, and therefore there is a point P on S, such that the distance from P to Q is r and such that P is a normal projection of Q on S. These constraints lead to a system of four equations:

$$\begin{aligned} S_1(P) &= 0 \\ S_2(P) &= 0 \\ \|P - C(t)\|^2 - r^2 &= 0 \\ \Big(P - C(t)\Big) \cdot \Big(N_1(P) \times N_2(P)\Big) &= 0 \end{aligned}$$

where $N_1(P)$ and $N_2(P)$ are the respective normals at P to S_1 and S_2. The four equations are solved simultaneously for t and for the three coordinates of P through numerical methods. Experimental results, in the case where $C(t)$ is a simple line, indicate that the method is too slow to be used in interactive systems.

We investigated another iterative algorithm to solve the same problem. The iterations were guided by geometric criteria. Given a starting point $S(u)$ on the spine, represented in a parametric form, compute the intersection $C(t)$ of the curve C with the plane normal to the spine at $S(u)$. $S(u)$ belongs to the normal projection of $C(t)$ on the spine. If the

distance $\|S(u) - C(t)\|$ is equal to r, then $C(t)$ is also on the canal face, and is an intersection of the curve with the canal face. If the distance $\|S(u) - C(t)\|$ is not r, a new point $S(u')$ on the spine is computed from the relative orientation of the vector $(S(u) - C(t))$ and of the tangents to the curve and to the spine at the considered points.

This algorithm was tried on the intersection of a line L with a canal face around a spine defined as the intersection–curve between two cylinders. Starting points were inferred from the intersections of the spine with a cylinder of radius r and axis L. Such intersection–points define the possible range for the starting point $S(u)$. Although no care was taken to develop an efficient implementation of this algorithm, timing–tests were discouraging (it took about one second per intersection) and we decided to approximate canal faces by smoothly joined pieces of standard faces.

6.1.4 Summary of requirements

To classify an edge with respect to a solid defined in CSGO we need the following routines:

— a point membership classification algorithm (discussed below),

— an algorithm that computes the intersection of a curve with standard and canal faces. We do this by using PCC approximations for curves and piecewise toroidal or cylindrical approximations for canal faces (see Chapter nine).

— a routine that generates, for any solid S defined in CSGO, a set of patches of S.

6.2 POINT MEMBERSHIP CLASSIFICATION

6.2.1 Motivation

To classify points, the procedure *TRIM_EDGE* uses *CLASS_POINT* discussed below. A simplified version of *CLASS_POINT*, can also be used for computing mass–properties by cell classification [Lee 82].

Point membership classification (PMC) amounts to determining whether a point is inside, outside, or on the boundary of a given solid. It can be done in CSG by *divide and conquer* methods: the point is classified against primitives and the results are combined according to the Boolean operators in the CSG tree. The correct processing of cases where the classified point lies on the boundary of two sub–solids requires the computation and combination of *neighborhoods.*

6.2.2 Neighborhoods

The notion of "neighborhood" is used in CSG for the classification of sets that lie on the boundaries of several sub–solids. For instance, given a point P that lies on the boundaries of both A and B, one cannot compute the classification of P with respect to $A \bigcup^* B$ by combining the classifications of P with respect to A and to B (see Figure 6.1).

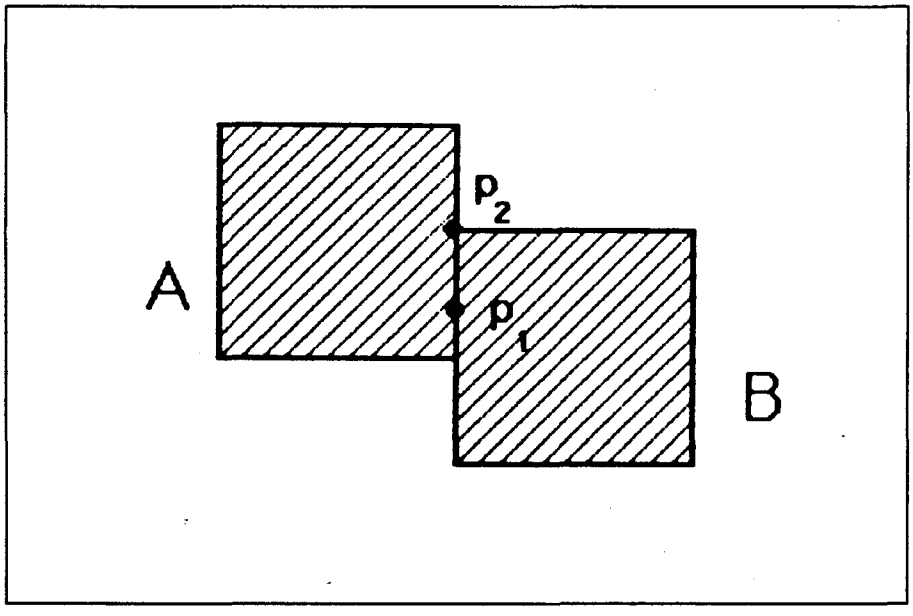

FIGURE 6.1: Both points P_1 and P_2 are on A and on B, but only P_2 is on $A \bigcup^* B$.

The "neighborhood" of a point P with respect to a solid X is the intersection of X with an open ball of center P and radius ϵ, which we shall consider infinitely small. Given a set X, if such a neighborhood is entirely in X, then P is in iX. On the other hand, if the neighborhood is in $\overline{S}$, the point is out of S. If the neighborhood is partly in and partly out of X, the point is on the boundary of X.

Edge classification algorithms used for boundary evaluation, displays, and so on, only require neighborhoods for points lying in the interior of edge segments [Requicha 85]. The neighborhood of an edge e with respect to a solid X can be viewed as the intersection with X of the tube $e\uparrow\epsilon$, and usually is not constant along the edge. However, in edge classification algorithms, we break the edge into segments of constant classification. It follows that one can compute segment classification from a cross–section of $e\uparrow\epsilon$ at any point lying in the interior of the segment. The cross–section is obtained by intersecting the neighborhood of e in X with a plane L that is perpendicular to e at its mid–point P. A two dimensional representation of the neighborhood is sufficient, and is obtained by generating the intersection–curves of L with all the patches of X that contain e. To combine neighborhoods at Boolean operator nodes one has to determine the order in which the curves are organized around P [Requicha 85]. Usually the tangents to those curves at P provide sufficient information, but if two curves have the same tangents higher order derivatives are required. For patches imbedded in quadric surfaces, the intersections with L are conics. In the worst case one needs only compute their derivatives at P up to the fourth order. The intersection of a torus with a plane in general is not a conic and its representation is therefore more complex. Also higher order derivatives are needed to combine neighborhoods induced by toroidal surfaces. Exact neighborhoods of canal faces would be even more complex, but we use piecewise toroidal and cylindrical approximations for these surfaces. We did not study in detail the problem of combining neighborhoods for tori.

As we shall see later, classification in CSGO requires PMC of points that do not lie in the interior of edge segments of constant classification. Neighborhoods of such points generally cannot be represented in 2–D. Representation for "vertex–neighborhoods" are unwieldy [Requicha 85], but we can avoid dealing with them because point classification can be

deduced from the classification of a sufficient set of edge segments as follows.

A point P will be on the boundary of the solid S if and only if there exists a face F of S such that P lies in the closure of F (i.e.; P can lie in the interior of F or on its boundary). If such a face F exists, it lies on at least one patch of S that contains P. Therefore, to classify P with respect to S we construct the set F_P of patches of S that contain P, and check whether any of these patches contain a region of ∂S in contact with P. (If F_P contains less than two patches the classification of P does not require neighborhoods and the technique described here need not be used.) In the neighborhood of P, a patch F_i of F_P is divided into region of constant classification by intersection–edges E_{ij} of F_i with other patches F_j of F_P. If for a given patch such edges exist, classification of each region can be deduced from the classification of its bounding edges. On the other hand, if in the neighborhood of P the intersections of a patch F_i with other patches of F_P is not one–dimensional (i.e.; the patches intersect only at P or are coincident), then F_i has (in the neighborhood of P) a constant classification with respect to S, and this classification can be obtained by classifying with respect to S any curve (*dummy edge*) of F_i that passes through P.

The actual classification of a point P with respect to the solid S can be organized as follows. First find the set F_P of all patches of S that contain P. If there is less than two, use simple classification without neighborhoods. If F_P contains more than one patch, compute the pairwise intersection–edges of the patches of F_P. On each patch of F_P that does not intersect other patches of F_P in an edge that passes through P, construct a dummy edge that passes through P and add it to the set of pairwise intersection–edges. (For instance when P is the apex of a cone, the dummy edge can be any linear generator of the cone.) At this point we have a set of edges E_{ij} that pass through P and lie on patches of F_P. For each edge E_{ij} compute the intersections of E_{ij} with each patch of S that does not contain E_{ij}. Sorting intersection points along E_{ij} defines segments of constant classification on E_{ij}. Consider only segments that are adjacent to P, and classify with respect to S an intermediate point on each segment. If all such segments are in the interior of S, P is in S. On the other hand, if all such segments are out of S then P is also out of S. Otherwise P is

on the boundary of S. In fact the classification can be interrupted as soon as a segment that lies on S, or two segments of different classification are found. It follows that we only need support PMC for points that lie on edge segments of constant classification, and the neighborhoods for such points can be represented in 2–D as noted earlier.

6.2.3 Algorithm

The high level design of a PMC algorithm for CSG [Requicha 77b] can be extended to CSGO (see below) by simply adding to the case statement a new branch that corresponds to offset nodes. *CLASS_POINT* takes a solid N and a point P that lies on a segment e of constant classification with respect to S, and returns R and *nbh*. R indicates whether P is in the interior, on the boundary, or in the complement of N. When $R = ON$, *nbh* is a 2–D representation of the intersection with N of the neighborhood of e at P.

```
function  CLASS_POINT(P,N):(R,nbh);
begin
case N.type of
primitive: (R,nbh):=CLASS_POINT_PRIMITIVE(P,N);
 motion:  begin
          P':=APPLY_MOTION([N.motion]^-1,P);
          (R,nbh):=CLASS_POINT(P',N.left);
          nbh:=APPLY_MOTION([N.motion],nbh);
          end;
 Boolean: begin
          (Rl,nbhl):=CLASS_POINT(P,N.left);
          (Rr,nbhr):=CLASS_POINT(P,N.right);
          (R,nbh):=COMBINE(Rl,Rr,nbhl,nbhr,N.type);
          end;
  Offset: begin
          (R,nbh):=CLASS_WRT_OFFSET(P,N.left,N.odist);
          end;
end;
```

Point membership classification against primitives can be performed (procedure *CLASS_POINT_PRIMITIVE*) by classifying the point against

the half–spaces that define the primitive; this is done by checking the sign of a low degree polynomial. For instance, if the primitive S is a sphere of radius r, a point will be **in** iS if it is closer than r to the center of the sphere. Neighborhoods of points with respect to primitives are also directly available.

Algorithms for combining classifications and edge neighborhoods at Boolean nodes (procedure *COMBINE*) are available [Tilove 80], for all standard surfaces but tori. We still need to provide algorithms for classifying points with respect to offset solids.

6.2.4 Offset nodes

To classify a point against the offset ($S\uparrow r$ or $S\downarrow^* r$) of a solid S defined in CSGO, we need the BRep[5] of S. To avoid evaluation of the boundary of S one might be tempted to classify the point with respect to the offsets of *primitives* of S and to combine the results accordingly to the CSG expression of S, but, as shown in chapter three, such an approach is incorrect because offsetting does not in general distribute over Boolean operations.

Point classifiers for offset solids are relatively straightforward. Let us study classification with respect to expanded solids; classification against shrunk solids is similar. The classification of a point P with respect to $S\uparrow r$ can be organized as follows. First one classifies P against S. If P is **in** or **on** S, then P is also **in** $S\uparrow r$. If P is **out** of S, its classification depends on the distance $d(P,S)$, which by property 3.10, is equal to $d(P,\partial S)$ for P out of S.

The function *CLASS_WRT_OFFSET* takes a point P, a solid S, and an offset distance r. It returns the classification of P with respect to $S\uparrow r$ and the corresponding neighborhood.

---

[5] The BRep of a solid contains a sufficient set F of patches, a set E of singular curves, and a set V of singular points of the solid.

```
function CLASS_WRT_OFFSET(P,S,r):(R,nbh);
begin
if CLASS_POINT(P,S)≠ OUT
    then R:=IN
     else (R,nbh):=POINT_O_BDRY(P,S,r);
end;
```

By property 3.13, $S\uparrow r = S \bigcup (\partial S)\uparrow r$, and therefore if P is out of S, it suffices to classify it against the version of ∂S expanded by r, which is done by the procedure *POINT_O_BDRY* as follows. *POINT_O_BDRY* checks whether $d(P,\partial S) < r$. For this it computes the minimum distances from P to the vertices, edges, and faces, of ∂S.

— If $d(P,\partial S) < r$ then P is in the interior of $S\uparrow r$.

— If $d(P,\partial S) > r$ then P is out of $S\uparrow r$ (or more specifically in the interior of the complement of $S\uparrow r$).

— If $d(P,\partial S) = r$ then, as pointed out in chapter three, P is on $\partial(S\uparrow r)$ for open S. If S is closed, one must compute the *neighborhood* of P in $(\partial S)\uparrow r$.

POINT_O_BDRY stops and returns "R=IN" as soon as it finds a point on ∂S that is closer than r to P. The procedure returns "R=OUT", if $d(P,\partial S) > r$.

Suppose now that $d(P,S) = r$ and the minimum distance is achieved several points Q_i of ∂S. Each Q_i lies on a boundary element E_i of S (i.e., a face, an edge, or a vertex of ∂S), and P is at a distance greater than r from all other elements of ∂S that do not contain a closest projection point Q_i. It follows that, in the neighborhood of P, $(\partial S)\uparrow r$ equals the union of the solids $E_i\uparrow r$ whose boundaries are contained in the set of faces F_i obtained by n–offsetting E_i. (See Figure 6.3 for an example.) Faces F_i are imbedded in surfaces A_i. (For instance, if Q_i lies on linear segment of an edge of S, then P lies on the cylinder obtained by n–offsetting the line segment.) The orientation of the n–offset surfaces A_i is also available; solid material is on the same side as Q_i. In the neighborhood of P, $S\uparrow r$ can be expressed as the union of the half–spaces defined by the oriented

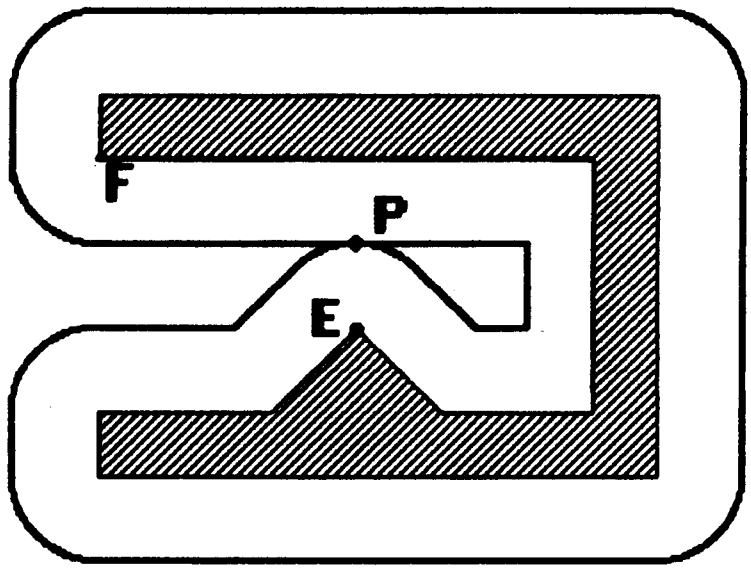

FIGURE 6.2: The point P is at distance r from the face F and the edge E of ∂S. The neighborhood of P in $S\uparrow r$, can be computed from $F\uparrow r \bigcup E\uparrow r$.

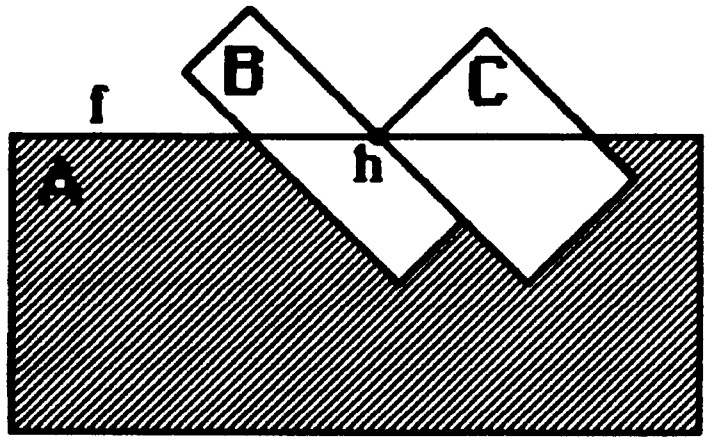

FIGURE 6.3: The point h is on the face f of A but also on B and on C. Neighborhoods are needed to conclude that h is out of $A -^* (B \bigcup^* C)$.

surfaces A_i. Neighborhood combination techniques discussed above can be used here to compute *nbh* (procedure *COMBINE_UNION_NBH*).

The distance $d(P, \partial S)$ is the minimum of the distances from P to the vertices and to the normal projections of P the edges and faces of S.

The *normal projection points* h of P on an edge e are defined in chapter four as the set of points h of e, such that the vector $(P-h)$ is orthogonal to the tangent to e at h. The computation of point/edge normal projections depends on the nature and representation of the edge, but in general is expensive if the edge is more complex than a line or a circle. If a parametric representation $e(t)$ is available for the edge e, we can compute the normal projections by finding the values of t for which $\|P - e(t)\|^2$ is extremum.

Each face of S is contained in a patch f of S and therefore is a subset of A, the host surface that contains f. The normal projections of P on a face of S are computed in two steps. First we find the normal projections h of P on A. The computation of normal projections of P on a standard surface are straightforward. The normal projections of P on a canal face can be simply computed from the normal projections Q of P on the spine of the canal face; it suffices to move from Q towards or away from P by the radius of the canal face.

In the second step we must discard the points h that do not lie on a face of S. Point/face classification can be done by standard boundary oriented methods, which require that one store, for each face, a list of pointers to the edges that bound the face, in addition to the patch or host surface. Alternatively, we can replace point/face classification by point/solid classification, because it is sufficient to know whether h lies on S or not. This can be tested by a recursive call to *CLASS_POINT*.

Normal projection points h, used to compute the distance from a point P to a solid S, do not necessarily lie in the interior of an edge segment of constant classification, and therefore their neighborhoods with respect to S cannot be represented and combined in two dimensions.

The point h may be "vertex–like", i.e., the intersection of several surfaces (see Figure 6.3), and its neighborhood cannot be represented in 2–D. As explained above, the classification of such points will be inferred from the classification of a sufficient set of edge segments.

Pseudo–code for *POINT_O_BDRY* follows. When $d(P,S) = r$, B contains the list of oriented offset surfaces used for computing the neighborhood of P in S.

```
function  POINT_O_BDRY(P,S,r):(R,nbh);
begin
B:=∅;
for each vertex v of V
      do if ||P − v|| < r
            then return R:=IN
             else if ||P − v|| = r then B:=B+SPHERE(v,r);
for each edge e of E
      do begin
         H:=PROJECT_ON_EDGE(P,e);
         for each point h of H
               do if ||P − h|| < r
                     then return R:=IN
                      else if ||P − h|| = r
                              then B:=B+CANAL_FACE(e,r);
         end;
for each face f of F
      do begin
         H:=PROJECT_ON_SURF(P,HOST_SURF(f));
         for each point h of H
               do begin
                  (R_h,nbh):=CLASS_POINT(h,S);
                  IF (||P − h|| < r) and(R_h = ON)
                     then begin
                             if ||P − h|| < r
                                then return R:=IN
                                 else B:=B+N_OFFSET_FACE(f,r);
                             end;
                  end;
         end;
if B= ∅
    then RETURN R:=OUT
     else RETURN (R,nbh):=COMBINE_UNION_NBH(P,B);
end;
```

6.2.5 Summary of requirements

To classify points in CSGO we must compute point/solid distances, and therefore we need:

— the BReps (i.e., patches, edges and vertices) of all the arguments of offset nodes in the CSGO tree (we shall see below how such BReps are computed),

— algorithms that compute normal projections on curves and surfaces. Such algorithms will be greatly simplified by our use of PCC approximations.

(The algorithms used by standard modellers, e.g., to combine neighborhoods, are also necessary, of course.)

6.3 BOUNDARY EVALUATION

Boundary evaluation is a considerably more complicated process than ray casting or mass-property calculation. Again, there are many approaches to boundary evaluation, but several CSG/BRep modellers, including PADL-2, use a more elaborate version of the following basic algorithms.

6.3.1 Non recursive algorithm

The function *BREP_EVAL* takes a solid (S) defined in CSG and returns a set F of patches and the set E of the edges of S.

function *BREP_EVAL(S):F,E;*
begin
F:=PATCHES(S);
E:=T_EDGES(F);
E:=TRIM_ALL_EDGES(E,S,ON);
end.

Thus, one starts with a set F of patches, or tentative faces that are guaranteed to include the faces of the desired object. In CSG the set F can be obtained as the union of the faces of the primitives that define S,

but in CSGO, the generation of a superset of the boundary of the represented solid requires n–offsetting operations. Specifically, if the CSGO representation contains an offset node that corresponds to $S\uparrow r$, we generate a superset for $\partial(S\uparrow r)$ by n–offsetting the edges, faces, and vertices of S. It follows that the BRep of S must be available before we compute the BRep of $S\uparrow r$. This leads naturally to an incremental boundary evaluation algorithm.

6.3.2 Incremental boundary evaluation

We propose below an adaptation to CSGO of a simplified recursive formulation of an incremental boundary evaluation algorithm in CSG [Requicha 85]. The function *BREP_EVAL_O* takes a node N of a CSGO tree and returns the corresponding BRep; i.e., the set of patches F, edges E and vertices V of the solid defined by N. The resulting BRep is attached to the current node. (A similar algorithm may be used to produce more elaborate BReps containing explicit connectivity information.)

```
function BREP_EVAL_O(N):F,E,V;
begin
case N.type of
primitive: (F,E,V):=PRIMITIVE_BREP(N);
  motion: begin
          (Fl,El,Vl):=BREP_EVAL_O(N.left);
          (F,E,V):=APPLY_MOTION([N.motion],(Fl,El,Vl));
          end;
 Boolean: begin
          (Fl,El,Vl):=BREP_EVAL_O(N.left);
          (Fr,Er,Vr):=BREP_EVAL_O(N.right);
          F:=Fl+Fr;
          E:=∅;
          for each pair of patches f1 and f2 of F
                do E:=E+SURF_INT_SURF(f1,f2);
          for each canal face f of F
                do E:=E+SELF_CROSS_EDGES(f);
          E:=TRIM_ALL_EDGES(E,N,ON);
          V:=VERTICES(E);
```

```
        end;
  Offset: begin
        (Fl,El,Vl):=BREP_EVAL_O(N.left);
        F:=N_OFFSET(Fl,El,Vl,N.odist);
        for each pair (f1,f2) of patches of F
              do E:=E+SURF_INT_SURF(f1,f2);
        for each canal face f of F
              do E:=E+SELF_CROSS_EDGES(f);
        E:=TRIM_ALL_EDGES(E,N,ON);
        V:=VERTICES(E);
        end;
attach (F,E,V) to N;
end;
```

PRIMITIVE_BREP returns the BRep of a primitive, and is straightforward since the BReps of standard primitives can be simply inferred from their types and dimensions. The patches of a primitive are its faces. The parameters, limits, and rigid motions of such patches can be simply derived from the parameters of the primitives (see Chapter ten for an example and a description of patch and primitive representations). The edges of standard primitives are line segments or circles. The vertices and singular points can also be simply obtained.

We still must discuss the generation of tentative intersection edges (procedures *SURF_INT_SURF* and *SELF_CROSS_EDGES*), edge classification *TRIM_ALL_EDGES*, and the generation of vertices (procedure *VERTICES*) and patches for offset nodes (procedure *N_OFFSET*).

For the sake of clarity we presented a simplified and non–optimized version of *BREP_EVAL_O*, which might compute the same tentative edges many times during the boundary evaluation of a complex solid. At Boolean nodes N one actually needs only to generate tentative edges that are intersections of patches of the left sub–tree with patches of the right sub–tree. Other tentative edges of N are already available in the BReps of the two sons of N and need only be classified against the other son.

6.3.3 Surface/surface intersection

At Boolean nodes, pairwise intersection of patches generates tentative edges, which together with self–intersection edges of canal faces are guaranteed to include those of the object.

The procedure *SURF_INT_SURF* takes two patches and returns their intersection, which might be composed of several tentative edges. In CSGO one might have to compute tentative intersection–edges of canal faces with other patches.

The problem of generating intersections of two surfaces will be discussed in the following chapters. Self–crossing edges of canal faces will be handled by approximating the canal faces with pieces of tori and cylinders, and by computing their pairwise intersections.

Extraneous portions of these tentative edges are discarded by classifying them with respect to N. The function *TRIM_ALL_EDGES* takes the set of edges E, the definition N of the solid, and the flag *IBO*. It returns the set of edge segments R that lie on N (if $IBO = ON$). *TRIM_ALL_EDGES* simply calls *TRIM_EDGE* (already discussed). Therefore we must assume that the set of patches for N and the BReps of all arguments of offset nodes in the definition of N are available.

function *TRIM_ALL_EDGES(E,N,IBO):R;*
begin
R:=∅;
for each *edge e of E* **do** *R:=R+TRIM_EDGE(e,N,IBO);*
end;

6.3.4 Patches of offset nodes

At each offset node in a CSGO tree we must generate a sufficient set of patches, whose union is guaranteed to include the boundary of the offset object. This amounts to computing a superset of the boundary of the offset solid $S\uparrow r$ or $S\updownarrow^* r$, which may be done by n-offsetting ∂S, as shown in chapter four. Therefore we must n-offset the patches, edges and vertices of ∂S. The necessary information is contained in the BRep of S, and n-offsetting patches, edges and points is straightforward.

The following facts may be used to generate smaller supersets, and therefore to speed-up patch generation, as well as subsequent computations. (We assume below that the positive direction of the normal to patches of S is toward the exterior of S.)

— Negative n-offsets (towards the interior of S) of patches need not be considered when computing patches for $S\uparrow r$. Similarly, positive n-offsets of patches are not needed for $S\downarrow^* r$.

— Concave edges need not be n-offset for $S\uparrow r$. Similarly, convex edges are not needed for $S\downarrow^* r$. BRep edges that separate tangent faces are not singular curves, because the normal to ∂S is continuous, and therefore never need to be n-offset.

The procedure *N_OFFSET* takes a CSGO representation S, the associated BRep (F,E,V), and an offset distance r, and returns the set R of patches of $S\uparrow r$. N–offsets for shrinking operations are obtained in a similar manner.

```
function N_OFFSET(F,E,V,r):R;
begin
R:=∅;
for each face f of F do R:=R+N_OFFSET_FACE(f,r);
for each convex edge e of E do R:=R+CANAL_FACE(e,r);
for each vertex v of V do R:=R+SPHERE(v,r);
end;
```

The function *N_OFFSET_FACE* takes a patch f with specified normal–direction N that points out of the solid, and a distance r, and returns $f\|_r^+$. Patches are standard faces or canal faces. N–offset of standard faces and canal faces can be simply obtained as discussed in Chapter four. The function *CANAL_FACE* takes an edge e and a radius r and returns a patch that is a canal face of spine e and radius r. We must also n–offset the vertices (all singular points) of S. The function *SPHERE* takes a point v and a radius r and returns the spherical surface of center v and radius r.

6.3.5 Vertices

Most vertices of a solid S are obtained from the end–points of the edges of S. Care must be taken here to count only once a vertex that appears in several adjacent edges.

We also need to n–offset singular points that do not lie on any edge; for example the apices of semi–cones, or the self–crossing points of a toroidal surface whose cross–section radius exceeds the radius of its spine. Such vertices could be treated separately by generating all singular points of the patches of S and discarding those that are not on S. However, when neighborhoods need be considered, it is simpler to provide robust classification algorithm for edges than for points. Points may correspond to vertices of several sub–solids, and their correct classification requires three–dimensional neighborhoods, while edge classification is based on the classification of mid–points, which do not lie on vertices of any sub–expression, and for which two–dimensional neighborhoods are sufficient [Requicha 85].

We shall use the following method to avoid direct classification of isolated singular points. For each patch that contains isolated points we construct a dummy edge that lies in the patch and contains the isolated points. (For half–cones this edge is a line, and for elbows, which are toroidal sections, this edge is a circle.) The dummy edge is not a singularity and in general does not correspond to a singular curve of the solid, therefore it should not be used in most of the algorithms. However, it will be classified together with all other edges in the boundary evaluation algorithm. Let e be such an edge starting at the singular point P. In the procedure *TRIM_EDGE*, e is first segmented into subsets whose mid–points are classified with respect to the solid. If the first segment is on the solid, then its end–points (which include P) are also on the solid. If dummy edges are marked, the procedure *TRIM_EDGE* need not classify all segments of e, but only those connected with isolated singular points.

The procedure *VERTICES* returns the end points of all the edges of the solid (including dummy edges), and therefore will return also isolated singular points that lie on the solid.

6.3.6 Summary of requirements

In addition to all the requirements for edge–classification and for PMC, we need procedures that compute surface/surface intersections.

6.4 COMPUTATIONAL REQUIREMENTS IN CSGO

We wish to point out that the above algorithms are presented in a very simplified form. For the sake of clarity we chose to present a non–optimized version of each algorithm. Some of the details will be discussed together with the description of our experimental modeller.

From the algorithms presented above we conclude that to perform on offset solids the usual computations available in solid modellers, we need the following tools.

- Procedures to compute the intersection–edges of any two standard or canal faces.
- Procedures to compute the intersection of a curve with a standard or canal face.
- Procedures to compute the normal projections of a point on an edge.

These low level tools will be described in the following chapters.

CHAPTER VII

APPROXIMATION OF EDGES

To integrate offsetting operations in standard dual modellers, we must support canal faces. The introduction of new surface-types complicates considerably the generation, representation and classification of intersection-edges. To avoid such complications we decided to approximate each canal face by a set of smoothly joined standard faces. The torus and the cylinder are simple cases of canal faces and therefore seem suitable for such approximations.

Piecewise cylindrical approximations of general canal faces are simple but their are not smooth.[6] Smoothness is important for offsetting operations; therefore we chose more complex approximations consisting of a sequence of smoothly joined pieces of tori or cylinders. Spines of cylinders are straight lines, and spines of tori are circles. Since canal faces can be represented by their radius and spine, we can generate a smooth piecewise cylindrical or toroidal approximation of any canal face by approximating its spine with a PCC (smooth piecewise linear or circular space curve).

Spines of the canal faces of an offset solid S correspond to intersection-edges of some sub-solids used in the definition of S. We propose in this chapter a method for deriving PCC approximations for all intersection-edges of solids defined in CSGO. Exact parametric representations can be computed analytically for intersection-edges of standard quadric surfaces. Therefore, the PCC approximations of such edges could be easily derived from the corresponding parametric representations. Unfortunately, to the best of the author's knowledge, algorithms that compute and process exact

[6] Throughout this thesis the term *smooth* denotes G^1 geometric continuity [Barsky 84].

representations of intersection-edges of tori with other standard surfaces are not available.

To overcome this problem, we developed a method for generating PCC approximations for intersection-edges between any two "standard faces" [7] including tori.

In addition to providing a simple approximation for canal faces and an economical way to fully support the torus, our method allowed us to simplify most computations performed on intersection-edges. For instance, the intersection of a PCC approximation with a torus can be computed analytically by finding the roots of fourth degree polynomials, while finding the intersection of an edge approximated by other methods with a torus generally requires solving much more complex equations. PCC approximations also are advantageous for computing distances, thus computing the minimum distance between a point and a cylinder/cylinder intersection-edge requires finding roots of an eighth degree polynomial, but replacing the edge by its PCC approximation reduces the problem to root–finding for second degree polynomials.

7.1 GENERATION OF INTERSECTION–EDGES

In this section we present our approach to the problem of computing PCC approximations for intersection–edges between any two standard faces (i.e., faces of standard primitives). Our method is a variation of the *parametric grid* method [Sabin 80] and can be adapted to non–standard patches or simply modified to generate smooth piecewise cubic curves instead of PCC's. It has the advantage of interpolating the exact intersection–edge at discrete intersection points, and therefore seems more suitable for solid modelling than *recursive subdivision*, which converts the faces into a grid of low degree patches (often planar facets) [Carlson 82, Koparkar 84]. For standard faces, our approach is also simpler and more efficient than *tracing methods*, which "crawl" along the intersection–edges from some starting points [Timmer 77], and compute the next point through expensive iterations. Our method has three steps:

[7] Standard faces are faces of standard primitives: blocks, cylinders, spheres, half-cones and tori.

(1) generate intersection points and tangents; (2) sort the points along the edge; (3) interpolate the points by PCC's. To meet accuracy requirements, one may have to iterate this sequence by recursively refining the approximation in selected areas.

The first step requires the computation of curve/surface intersections. For patches (faces of standard primitives), such computations can be reduced to the intersection of lines or circles with standard surfaces. An efficient implementation is proposed in chapter nine.

The second step is usually expensive; it requires sorting a list of points in space. We solve it efficiently by mapping intersection points in an array of cells in the two-dimensional parameter space of one patch. Our method locates consecutive points practically without searching.

In the last step, we interpolate pairs of consecutive intersection-points and tangents with twisted bi-arcs, thus generating a PCC that interpolates the actual edge at all intersection points. Computation of twisted bi-arcs and representation of the PCC are derived in chapter eight.

7.1.1 Constant parameter curves

Given a solid S, defined as a Boolean combination of primitives P_i, any face of S is contained in the union of the faces of all P_i. Faces of standard primitives can be parameterized in such a way that constant parameter curves correspond to lines of curvature, and are circles or line segments (see Figure 7.1). A patch can be represented in its bi-parametric form by a vector-valued function $F(u,v)$ that maps a rectangle in parameter space ($u \in [u_1,u_2]$ and $v \in [v_1,v_2]$) into a standard face. Constant parameter curves (called also *generators*) are *u-curves* of the form $C_u(t) = F(u,t)$, and *v-curves* of the form $C_v(t) = F(t,v)$.

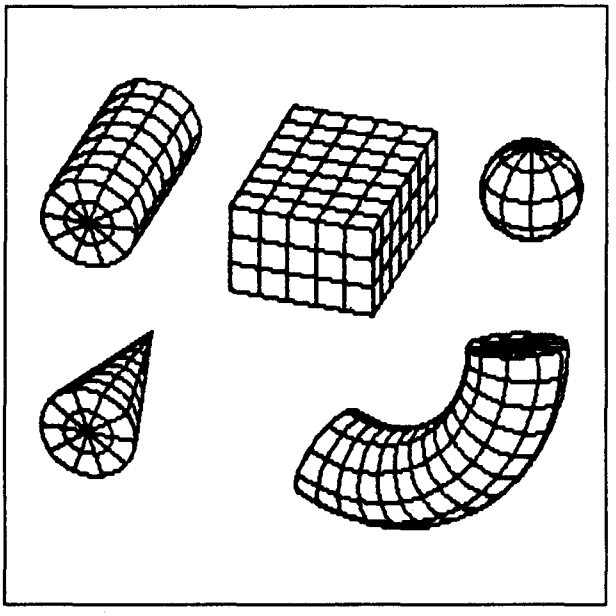

FIGURE 7.1: Constant parameter curves of standard faces are straight lines or circles.

7.1.2 Intersection points and tangents

Let F_1 and F_2 be two patches contained respectively in the standard surfaces S_1 and S_2. We assume that the intersection of S_1 with S_2 is a set of dimension one or is empty (special cases such as coincident or tangent surfaces are treated separately). We generate points that lie on F_1 and on S_2 by using a grid of generators (u-curves and v-curves) of F_1. We compute the intersection points of each generator with S_2. Unless we are in a special case where the whole generator lies on S_2, we obtain at most four intersection points. For example, given a u-curve C_u of F_1, we compute its intersection points $C_u(v_i)$ with S_2. These intersection-points are represented in the parametric space of F_1 by the parameter values (u, v_i), and lie on F_1 and on S_2, but can be out of F_2. Points out of F_2 could be easily rejected, since, given a point P on S_2, it is easy to obtain the parameterization (u, v) such that $P = F_2(u, v)$. However, we must keep such points, because they will be needed to generate the last bi-arcs that are partly on[8] and partly out of F_2 (Figure 7.2). It is necessary to generate similar intersection points for v-curves of F_1, otherwise we could

[8] Such a bi-arc is the approximation of a segment E of the intersection-edge between S_1 and S_2. By saying that the bi-arc is partly on F_2 we mean that E is partly on F_2, since, in general the bi-arc is not entirely "on" any of the intersecting surfaces.

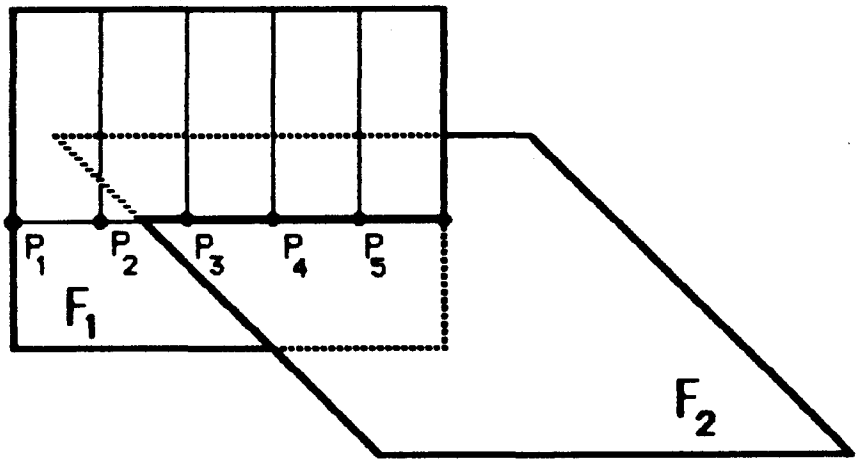

FIGURE 7.2: The point P_2 is out of F_2, but is needed to specify the bi-arc (P_2, P_3), which partly lies on F_2.

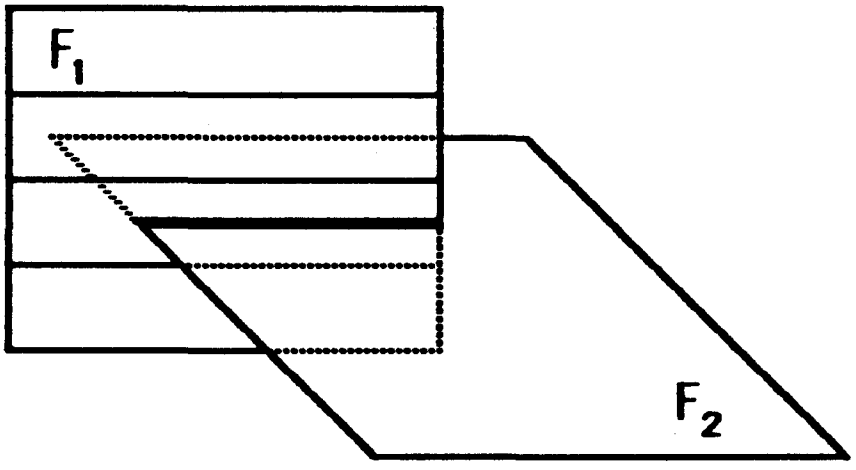

FIGURE 7.3: Using only u-curves of F_1, we miss the intersection edge.

miss large intersection-edges that are almost parallel to u-curves of F_1 (Figure 7.3).

The tangents to intersection points will be used as additional constraints for the approximation. The exact tangent to the intersection-curve at an intersection point P can be computed as the cross-product of the normals at P to S_1 and to S_2. Normals to standard surfaces are simple to obtain. When the two normals are colinear, the two surfaces are tangent, which implies a self-crossing of the intersection-edge. We shall not attempt to predict such situations, but rather isolate them during the matching process, as explained below.

7.1.3 Cellular approach

It is desirable to produce evenly spaced intersection points, and to provide an upper bound on the size of details missed by the approximation. The generators form a grid, which divides the patch F_1 into *cells*. The cell that corresponds to the parameter interval $[u_1, u_2] \times [v_1, v_2]$ is defined by the function $F(u,v)$ for $u \in [u_1, u_2]$ and $v \in [v_1, v_2]$. The "size" of the cell is controlled by the two increments $\delta u = u_2 - u_1$ and $\delta v = v_2 - v_1$, and the boundary of such a cell is composed of four segments: the segments of C_{u_1} and of C_{u_2} for $v \in [v_1, v_2]$, and the segments of C_{v_1} and of C_{v_2} for $u \in [u_1, u_2]$.

For general bi-parametric patches, it is difficult to control the maximum distance between two consecutive u-curves C_u and $C_{u+\delta u}$, by adjusting the increment δu. We adopt the following definition.

DEFINITION 7.1: **The *maximum distance* between two curves C_1 and C_2 is $m(C_1, C_2) = \inf r$ for $C_1 \subset C_2 \uparrow r$ and $C_2 \subset C_1 \uparrow r$.**

It follows from this definition and from the definition of expanded sets that any point of C_1 is closer than $m(C_1, C_2)$ to C_2 and *vice versa*. For standard surfaces, we can easily control the maximum distance between two consecutive generators. This allows us to control the "size" of the cells, i.e., the maximum distance from any point of a cell to the boundary of the same cell. Because cells on standard surfaces are bounded by lines or circles, given a point $F(u,v)$ in the cell defined by $[u_1, u_2] \times [v_1, v_2]$, the minimum distance from $F(u,v)$ to the boundary of the cell is less than or equal to the minimum distance d from $F(u,v)$ to the four points $F(u, v_1)$,

$F(u,v_2)$, $F(u_1,v)$, and $F(u_2,v)$. An upper bound for the size of the cell is provided by the maximum of d over all points of a cell. Using our "natural" parameterization of standard faces, the distance d is maximum when $u = \frac{u_1+u_2}{2}$ and $v = \frac{v_1+v_2}{2}$ and can be simply evaluated for each type of face.

7.1.4 Matching intersection points

The parameters v_i of the intersections of generators C_u of F_1 with S_2 are stored in an array $V[u]$, indexed by discrete values of u. Each element of the array contains at most four v values. The parameters u_i of the intersections of generators C_v of F_1 with S_2 are stored in a similar array $U[v]$, indexed by discrete values of v. Given a cell defined as $F_1(u,v)$ with $u \in [u_1,u_2]$ and $v \in [v_1,v_2]$, the intersections of its boundary with S_2 contains at most sixteen points, which can be found in four lists. Each list has at most four elements. The intersection points are:

- $F_1(u_1,v)$ with v stored in $V[u_1]$ and $v \in [v_1,v_2]$,
- $F_1(u_2,v)$ with v stored in $V[u_2]$ and $v \in [v_1,v_2]$,
- $F_1(u,v_1)$ with u stored in $U[v_1]$ and $u \in [u_1,u_2]$,
- $F_1(u,v_2)$ with u stored in $U[v_2]$ and $u \in [u_1,u_2]$,

If care is taken to count twice intersection points that are at the corner of a cell, and tangency points where a generator "touches" S_2, the boundary of each cell contains an even number of intersection points, because S_2 is a closed or infinite surface (see Figure 7.4), and therefore the curve of intersection between S_1 and S_2 cannot terminate (have an end–point) within a cell. If we have more than two points for a single cell, we propose to use adaptive subdivision to match these intersection points into pairs. We tried heuristic approaches based on the direction of the tangents associated with each point, but our experiments show that one cannot guarantee that such approaches will produce the appropriate results, especially near points where the two surfaces are almost tangent. An example is shown in Figure 7.5.

When the boundary of a cell contains more than two intersection points, or when the approximation generated does not satisfy the precision

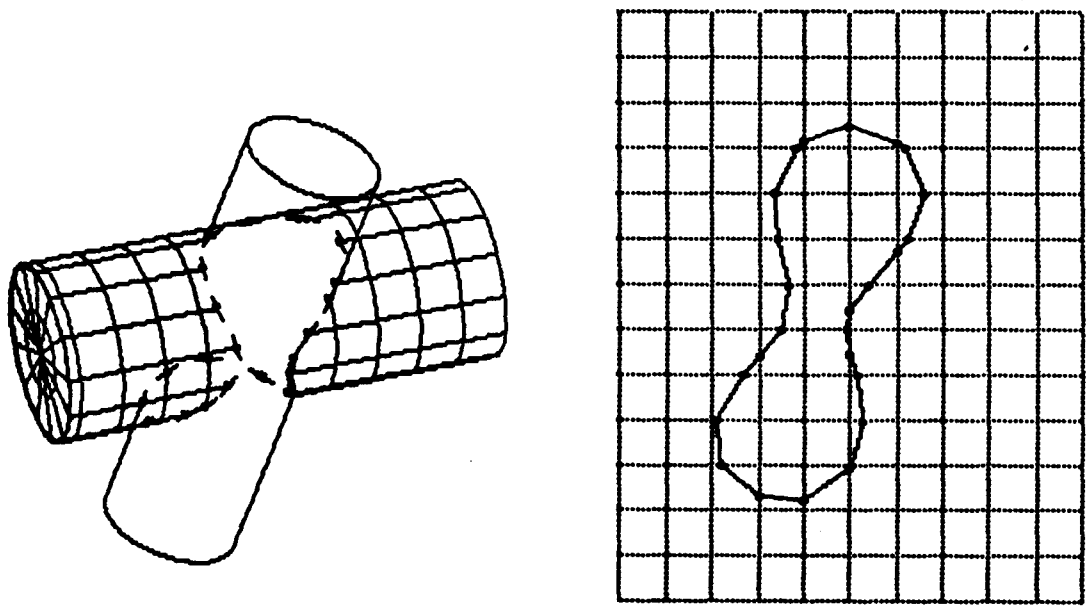

FIGURE 7.4: Intersection points are mapped into cells of the parametric grid.

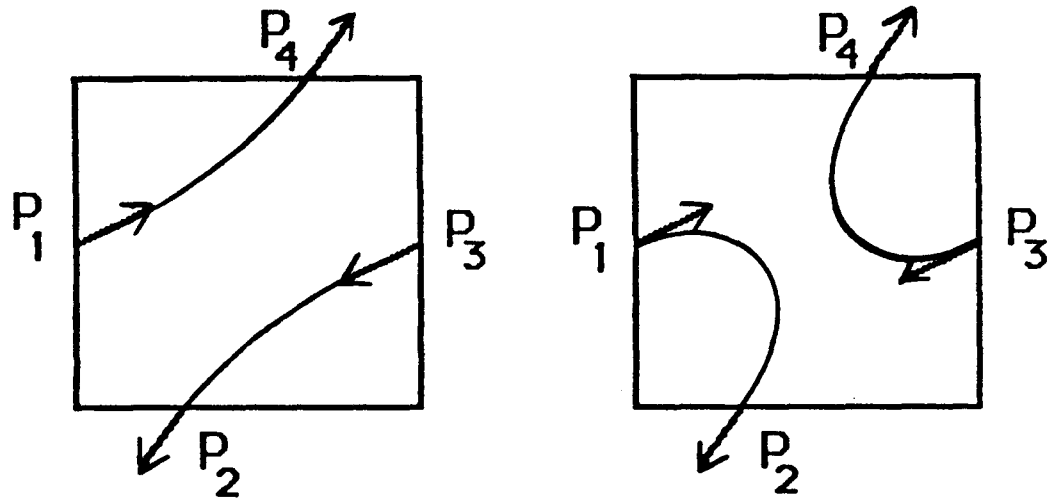

FIGURE 7.5: Given four intersection points P_i and the associated tangents (left), most approaches based on point position and tangent direction would indicate that the point P_1 should be matched with P_4. It is not the case in reality (right).

criterion discussed below, the cell can be recursively subdivided into four cells, until all precision criteria are met and all ambiguities resolved, or until the "size" of the cell is less than a specified limit. To subdivide a cell, we propose to use generators that correspond to mid-values of the parameters defining the cell. If the subdivision process reaches the cell's minimum size without resolving the matching ambiguities (more than two intersection points remain on the boundary of the cell), we could declare that the edge is self-intersecting, but it is easier to match the points so that the two induced edge segments do not intersect. There is not enough information to infer the correct topology of the intersection; we cannot distinguish between possible topologies on the basis of approximate geometric data. This seems to be of no consequence in solid modelling.

Our cell–based approach to the matching problem seems faster and more robust than for instance a similar method reported in [Varady 83, Jared 84] and used in the BUILD modeller. Varady's method uses only u-curves and an isosceles–triangle search to sort the intersection points.

7.1.5 Interpolation

After the matching process, each pair of points (with associated tangents) is interpolated by a *twisted bi-arc*. The generation of such bi-arcs is discussed in the next chapter. The result is a geometrically smooth piecewise circular or linear curve (PCC), which interpolates all intersection points and their tangents. We believe that the PCC is a better approximation than popular curves such as B-splines, which do not interpolate tangent directions. Although geometrically smooth, a PCC will not necessarily have continuous derivatives with respect to a specific parameterization. Its curvature is piecewise constant but not continuous, and its torsion is null except at junction points. Our approach is *local*: we need only to consider two consecutive intersection points at a time. This requires less computation than most global methods [Harada 82].

7.1.6 Precision

We do not have an explicit parameterization of edges of intersection of tori and therefore it is difficult to compute a bound on the maximum distance between the real edge and its PCC approximation. Instead, we define the error of the approximation as the maximum of the distances from all points of the PCC to the two surfaces that intersect at the approximated edge. This error measure is more meaningful than the distance between curves, because we must ensure that set membership classification algorithms classify an edge approximation as "on" the surfaces that intersect at the true edge.

Point/surface distance is simple to compute for standard surfaces, (see chapter nine), however computing its maximum over a circular or even linear arc is too expensive for practical use. Given a surface $F(u,v)$ and a curve segment $C(t)$, we must compute values of u, v, and t, for which the distance $\|C(t)-F(u,v)\|$ is extremum. Setting to zero the partial derivatives of the square of this distance with respect to each variable yields three non-linear equations. Initial investigation indicated that such an approach is not worth implementing. We propose to replace the exact maximum distance between an arc and a surface by an estimate computed at specific points of the interpolating curve, or of its convex hull. The optimum choice of such points and the derivation of an error bound remain open issues.

7.2 TORUS PROFILE EDGES

Line drawings of curved-face solids require the computation of *profile edges*, sometimes called silhouette edges, which are view-point dependent. Drawing a torus or a sphere should convince the reader of the importance of profile edges. Simple expressions for profile edges of standard quadrics are available for both perspective and parallel views. Profile edges of tori and therefore of piecewise toroidal approximations of canal faces are more complex. We approximate them by PCC's, computed as follows.

The profile of a smooth face F is the locus of all profile points P of F such that the normal $N(P,F)$ to F at P is orthogonal to the viewing vector $W(P)$. For parallel views, the viewing vector is constant. For perspective views $W = P - V$, where V is the view-point. We generate profile edges by a technique similar to that used for the intersection-edge

generation. We use only u-curves, which are the circular cross-sections of the torus. Each cross-section has zero or two profile points, and these can be computed easily. Unfortunately the tangents to the profile edge at those profile points are not directly available. We only have one constraint for such tangents: they must be orthogonal to $N(P,F)$. We developed a scheme for estimating the tangents to a curve that interpolates a series of points in space. The scheme is explained in chapter eight and can be used to estimate the tangents necessary for the computation of the PCC approximation of profile edges as follows. Given a sorted list of profile points, first obtain an estimated tangent T at each profile point P by the method of chapter eight. These tangents need not lie in the face F. Then, to guarantee that the approximation of the profile edge will pass through the profile points, and will be tangent to F at those points, we compute the projection of T on the plane tangent to F at P and use the projected tangents in the PCC fitting algorithm. This tangency constraint is important when profile edges have to be classified against solids that contain several instances of the same face F: the tangent can be used to derive edge neighborhoods for the combination of on/on classification results.

We have only two profile points per generator, therefore matching is much simpler that in the case of intersection-edges. Two profile curves (lists of profile points) are built incrementally, as we go from one u-curve to the next one.

Profile edges are used only for display purposes and the precision to which they are approximated is not crucial. However, the classification of loosely toleranced approximations can, in certain cases, produce wrong results, which correspond to unrealistic pictures. A similar problem arises in the classification of approximation curves for intersection edges, and is addressed in Chapter nine.

7.3 VERTICES

To support offsetting operations, we also need a list of vertices. They are used for computing point/solid distance, and for generating the bounding supersets of offset solids. Vertices can be obtained as ends of classified edges. However, since we use approximations for edges, we cannot in general obtain exact vertices that lie on three intersecting surfaces. For instance, given two surfaces S_1 and S_2, their common intersection-edge will be approximated by the PCC denoted E_{12}. Classifying E_{12} against the solid will produce segments of E_{12}. An end-point of such a segment will lie on a third surface S_3, but will probably not lie on S_1 nor on S_2, because E_{12} is guaranteed to lie on $S_1 \bigcap S_2$ only at intersection points. Similarly, the corresponding vertex of the edge E_{13}, intersection of S_1 with S_3, will lie on S_2, but not on S_1, nor on S_3. Therefore, the three edge-approximations E_{12}, E_{23}, and E_{13} will not meet at $S_1 \bigcap S_2 \bigcap S_3$, and three different vertices will result. Each of them will lie on one of the three surfaces and will be within specified tolerance from the two others. For n-offsetting or for point-solid distance computation we use the center of mass of the three vertices. Another, and more expensive, solution would be to compute $S_1 \bigcap S_2 \bigcap S_3$ directly (using iterative methods) and update the descriptions of the three edges so that they pass through the true vertex.

CHAPTER VIII

PIECEWISE CIRCULAR CURVES

In this chapter, we present a novel approach to curve approximation. The interpolation of an ordered set of *sampling* points in space with a smooth parametric curve is an important tool in many computer applications, especially in computer aided design and manufacturing. Most popular interpolation schemes in three dimensions are based on piecewise cubic curves. In order to simplify drastically the computations performed on curves in CAD/CAM systems, a new interpolating scheme was developed; it produces a smooth[9] piecewise circular or linear curve called "PCC". The whole curve passes through sampling points with specified tangent directions. If not available, the tangent directions can be derived from the position of the neighboring sampling points. The computational advantages of an approximation with simple arcs are important in solid modelling: PCC approximation allows simple classification of edges, and permits to approximate offsets of solids without domain extension (i.e., avoiding the necessity to introduce new surface–types).

8.1 INTERPOLATION

Much effort has been spent to develop methods that interpolate a sequence of sampling points in space by a three times differentiable, piecewise cubic space curve without undesirable waves [McLaughlin 83]. The quality of these interpolations is defined by a subjective visual criterion. A computational price for second order continuity is paid twice, first at the curve generation time (large matrix inversion for interpolating B–splines

[9] Smoothness refers to geometric continuity (G^1) [Barsky 84], which is parameterization independent (it is also called *visual smoothness*).

or Ferguson splines) [Schaffner 81], [Liou 76], and a second time when computing the geometric properties of the curve (intersections with surfaces, minimum distances between points and curves, and so on...). When continuity of tangent direction (G^1) is sufficient, a "smooth" piecewise conic — or even piecewise circular – curve can be used. Such curves can be efficiently generated and processed.

One method of generating a piecewise conic curve that passes through n sampling points P_i, $(i = 1 \ldots n)$ is to compute the $n+1$ control points C_i $(i = 0 \ldots n)$ of a quadratic B–spline. A quadratic B–spline curve generally does not pass through its control points, and therefore sampling points cannot be used as control points. A uniform quadratic B–spline passes through the mid–points of any two consecutive control points. Matching these mid–points with sampling points one obtains n constraints of the form: $C_i + C_{i-1} = 2P_i$, which define a family of quadratic B–splines that interpolate all n sampling points. Three degrees of freedom remain for the choice of the $n+1$ control points, and fixing any one of the control points C_i is sufficient to specify a particular curve (for instance one could choose C_0 to minimize the sum of the squares of the sides of the polygon $C_0, C_1, ..C_n$). Such a constraint influences the shape of the whole curve and therefore annihilates the *local control* property of B–splines; modifying a single sampling point will change the whole curve. (Lack of local control usually produces unpleasant waves.) The resulting curve can be decomposed into a series of smoothly joined pieces, such that each piece interpolates two consecutive sampling points in position but not in tangent direction. We show below that each piece is a quadratic Bezier curve and therefore is a parabolic segments (because the parabola is the only quadratic parametric curve).

A point $P(u)$ on a B–spline curve is defined as the weighted average of three consecutive control points (for example C_0, C_1, C_2, for u in $[0,1]$) as

$$P(u) = C_1 + \frac{(1-u)^2}{2}(C_0 - C_1) + \frac{u^2}{2}(C_2 - C_1)$$

By taking $B_0 = (C_0 + C_1)/2$, $B_1 = C_1$ and $B_2 = (C_1 + C_2)/2$ we can define the same parabolic arc as a Bezier curve

$$P(u) = B_1 + (1-u)^2(B_0 - B_1) + u^2(B_2 - B_1)$$

it is important to note that:

$$P(0) = B_0, \quad P(1) = B_2, \quad P'(0) = 2(B_1 - B_0), \quad P'(1) = 2(B_2 - B_1)$$

We call (B_0, B_1, B_2) the *control triangle* of the conic arc. The arc passes through B_0 and B_1, and is tangent to $B_1 - B_0$, and to $B_2 - B_1$.

We propose a method that, instead of a parabolic piece, uses a circular bi–arc. Such a bi–arc interpolates two sampling points in position and in tangent direction, which can be chosen to match the tangent to the interpolated intersection–edge.

The use of quadratic B–splines induces computational costs, which can be avoided with PCC's. For example, the closest distance between a point X and the parabola satisfies $(P(u) - X) \cdot P'(u) = 0$ where $P(u)$ is defined as above. The optimal values of the parameter u can be computed by finding the roots of a third degree polynomial. If we use a circular bi–arc instead of a parabola, the minimum distance problem can be solved more efficiently.

In addition, approximating the spine of a canal surface with a PCC provides a simple approximation of the canal surface, which is not the case when the spine is approximates with a quadratic B–spline.

8.2 TWISTED BI–ARCS

We showed in chapter seven that, to approximate the intersection–edge between two patches (or faces), we can proceed in three steps: generate a list of sampling points P_i and associated tangent directions T_i, order the points along the edge, and finally compute a PCC that interpolates the sampling points and tangents. We saw how the first step can be done by intersecting constant parameter curves of one face with the other surface, and how the second step can be optimized by mapping intersection points into cells in the two–dimensional parameter space.

The third step can be broken into a series of smaller independent sub–problems: how to fit a simple space curve through two end–points with specified tangent directions. This can be easily achieved with one cubic arc (Hermite or Bezier for example). Two smoothly joined conic arcs can

also be used to generate a bi–arc span. Such a method was developed for circular arcs in two dimensions [Sabin 77] and for parabolic twisted bi–arcs in three dimensions [Varady 83]. We developed independently a similar technique for twisted circular bi–arcs in three dimensions.

Given the boundary conditions (two points P_1 and P_2 and the associated tangent unit vectors T_1 and T_2) we wish to construct a smooth curve that is piecewise linear or circular, and that satisfies at its ends the boundary conditions. In general there is an infinite number of ways to construct such a curve with two smoothly connected arcs (linear or circular).

A circular arc of center O, radius r and end–points A and C can be described by its control triangle ABC (Figure 8.1). (If the angle (OA, OC) is larger than 180 degrees we use the control triangle that defines the complement of the arc in the circle.) The control triangle is isosceles, and the arc interpolates the points A and C and is tangent to the sides AB and CB. For arcs that are line segments, which correspond to circular arcs of infinite radius, we obtain a degenerate control triangle: the line segment AC, which coincides with the arc.

To define a smooth bi–arc that satisfies the boundary conditions, we build a control polygon made of two smoothly joined control triangles $A_1B_1C_1$ and $A_2B_2C_2$, as shown in Figure 8.2. Thus to determine the eighteen coordinates of the six points A_1, B_1, C_1, A_2, B_2, and C_2, we have the following constraints:

- The polygon satisfies the boundary conditions in position (six equations):

$$A_1 = P_1 \quad \text{and} \quad C_2 = P_2$$

- The polygon can be decomposed into two isosceles triangles (two equations):

$$\|B_1 - A_1\| = \|C_1 - B_1\| \quad \text{and} \quad \|B_2 - A_2\| = \|C_2 - B_2\|$$

- The two control triangles are smoothly connected at their common

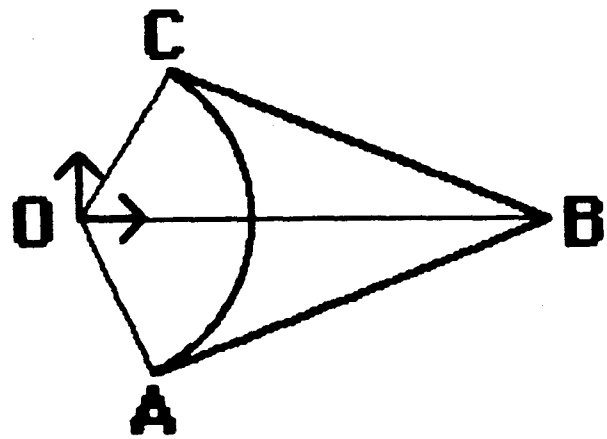

FIGURE 8.1: The circular arc is completely defined by its control triangle (A, B, C).

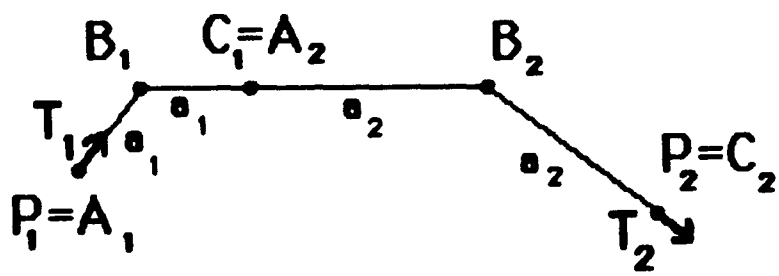

FIGURE 8.2: The end points P_1, P_2 and the tangents T_1, T_2 are interpolated by a control polygon (A_1, B_1, B_2, C_2).

- The polygon satisfies the boundary conditions in tangent directions (six equations and two new variables a_1 and a_2):

$$B_1 = P_1 + a_1 T_1 \quad \text{and} \quad B_2 = P_2 - a_2 T_2$$

It follows that we have twenty equations and twenty one variables, and therefore one degree of freedom. The whole system of equations can be simplified by noticing that the control polygon can be constructed as follows:

$$\begin{aligned} A_1 &= P_1 \\ C_2 &= P_2 \\ B_1 &= P_1 + a_1 T_1 \\ B_2 &= P_2 - a_2 T_2 \\ C_1 &= B_1 + \frac{a_1}{a_1 + a_2}(B_2 - B_1) \\ A_2 &= C_1 \end{aligned}$$

Two unknowns remain: a_1 and a_2. They are bound by a single equation:

$$\|B_2 - B_1\| = a_1 + a_2$$

which becomes:

$$\|(P_2 - a_2 T_2) - (P_1 + a_1 T_1)\| = a_1 + a_2$$

8.2.1 General equation

We can solve this equation for a_2 by first squaring both terms:

$$\|P_2 - P_1 - (a_2 T_2 + a_1 T_1)\|^2 = (a_1 + a_2)^2$$

which, with $S = P_2 - P_1$, yields:

$$\begin{aligned} \|S\|^2 - 2S \cdot (a_1 T_1 + a_2 T_2) + a_1^2 \|T_1\|^2 + a_2^2 \|T_2\|^2 + 2a_1 a_2 T_1 \cdot T_2 \\ = a_1^2 + a_2^2 + 2a_1 a_2 \end{aligned}$$

With $\|T_1\| = \|T_2\| = 1$ we obtain:

$$a_1 a_2 (T_1 \cdot T_2 - 1) + \frac{\|S\|^2}{2} = a_1 (S \cdot T_1) + a_2 (S \cdot T_2)$$

or

$$a_2 (a_1 (T_1 \cdot T_2 - 1) - S \cdot T_2) = a_1 S \cdot T_1 - \frac{\|S\|^2}{2}$$

8.2.2 Degenerate case

Let us consider the case where the right–hand side of this equation is null:

$$a_1 S \cdot T_1 = \frac{1}{2}\|S\|^2$$

This equation constrains B_1 to lie in the plane normal to S and passing through the mid–point $\frac{1}{2}(P_1 + P_2)$. In such case, if $a_1(T_1 \cdot T_2 - 1) \neq S \cdot T_2$ then we have one solution: $a_2 = 0$, otherwise, a_2 can take any value. We shall examine such cases, characterized by the two equations:

$$a_1(S \cdot T_1) = \frac{1}{2}\|S\|^2$$
$$a_1(T_1 \cdot T_2 - 1) = S \cdot T_2$$

Under those conditions, if $T_1 \cdot T_2 = 1$ then $T_1 = T_2$ and $S \cdot T_2 = 0$, which implies that $S \cdot T_1 = 0$ and that $\|S\| = 0$. This corresponds to a degenerate case, where the bi–arc is reduced to a single point ($P_1 = P_2$ and $T_1 = T_2$). Let us suppose now that $T_1 \neq T_2$. We can eliminate a_1 from the system of two equations and obtain

$$2(S \cdot T_2)(S \cdot T_1) = \|S\|^2(T_1 \cdot T_2 - 1)$$

Dividing this equation by $\|S\|^2$, and choosing a coordinate system such that S is aligned with the X–axis, we obtain:

$$2x_1x_2 = x_1x_2 + y_1y_2 + z_1z_2 - 1$$

or

$$x_1x_2 - y_1y_2 - z_1z_2 = -1$$

where $T_1 = (x_1, y_1, z_1)$ and $T_2 = (x_2, y_2, z_2)$. Define the real quantity g as:

$$g = (x_1 + x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2$$

Using $x_1^2 + y_1^2 + z_1^2 = 1$ and $x_2^2 + y_2^2 + z_2^2 = 1$, we obtain:

$$\begin{aligned} g &= x_1^2 + 2x_1x_2 + x_2^2 + y_1^2 - 2y_1y_2 + y_2^2 + z_1^2 - 2z_1z_2 + z_2^2 \\ &= x_1^2 + y_1^2 + z_1^2 + x_2^2 + y_2^2 + z_2^2 + 2(x_1x_2 - y_1y_2 - z_1z_2) \\ &= 1 + 1 - 2 = 0 \end{aligned}$$

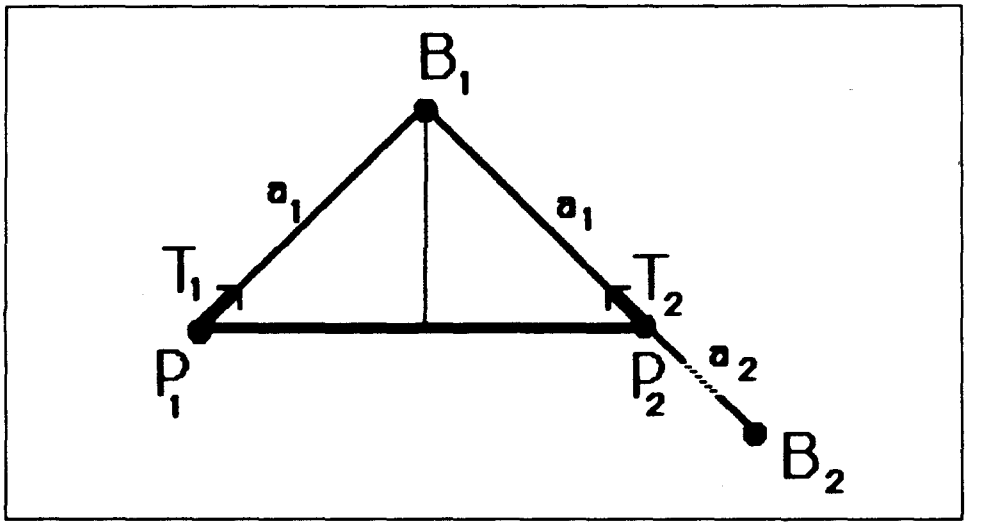

FIGURE 8.3: In the special case where $a_1(T_1 - T_2) = S$, the second triangle degenerates into a cusp and a_2 is unconstrained.

g is the sum of three positive terms, therefore each term must be null:

$$x_1 = -x_2$$
$$y_1 = y_2$$
$$z_1 = z_2$$

This shows that $S \cdot T_1 = -S \cdot T_2$. The vector $a_1(T_1 - T_2)$ is parallel to S and since $S \cdot (2a_1T_1) = \|S\|^2$ we obtain $a_1(T_1 - T_2) = S$. It follows that

$$B_1 = P_1 + a_1T_1 = P_1 + S + a_1T_2 = P_2 + a_1T_2$$

which implies that the first control triangle is (P_1, B_1, P_2). The second control triangle is (P_2, B_2, P_2), and it degenerates into a cusp at P_2, regardless of the value of a_2. This situation is depicted in Figure 8.3.

8.2.3 General solution

In the general case $a_1(T_1 \cdot T_2 - 1) \neq S \cdot T_2$ and one can compute a_2 for each a_1 as follows:

$$a_2 = \frac{a_1(S \cdot T_1) - \frac{1}{2}\|S\|^2}{a_1(T_1 \cdot T_2 - 1) - S \cdot T_2}$$

8.2.4 Equi–sided control polygon

One degree of freedom remains. a_1 can be chosen to minimize the curvature, the total arc length, the twist or any other characteristic of the generated bi–arc. We investigated several of these criteria and concluded that they lead to expensive computations and often have undesirable side effects. We chose to fix $a_1 = a_2 = a$, which produces very regular curves for all cases and leads to efficient evaluation.

With $a_1 = a_2 = a$, $S = P_2 - P_1$, and $T = T_1 + T_2$, the general equation is $\|S + aT\|^2 = 4a^2$, which can be written as $\|S\|^2 + a^2\|T\|^2 + 2a(S \cdot T) = 4a^2$, or

$$a^2(\|T\|^2 - 4) - 2a(S \cdot T) + \|S\|^2 = 0$$

The discriminant of this second degree equation in a is

$$d = (S \cdot T)^2 + \|S\|^2(4 - \|T\|^2)$$

and is never negative since $\|T\| \leq 2$. If $\|T\|^2 \neq 4$, then we obtain a positive solution:

$$a = \frac{\sqrt{d} - S \cdot T}{4 - \|T\|^2}$$

a is always positive since $d = (S \cdot T)^2 + \|S\|^2(4 - \|T\|^2) \geq (S \cdot T)^2$.

Let us suppose now that we are in the special case where $\|T\|^2 = 4$, which is equivalent to $T_1 \cdot T_2 = 1$ or $T_1 = T_2$. In this case, if $S \cdot T_1 \neq 0$ then

$$a = \frac{\|S\|^2}{4(S \cdot T_1)}$$

If $S \cdot T_1 < 0$ then a is negative, but we can still obtain an interpolation without cusps by using the complement of the circular arcs defined by the control polygon (see Figure 8.4).

If $T_1 = T_2$ and $S \cdot T_1 = 0$, we use half circles as shown in Figure 8.5.

8.3 REPRESENTATION

Alternative schemes for representing PCC's are discussed in the following subsections.

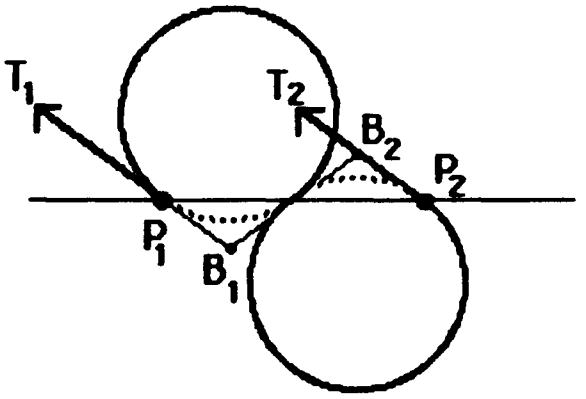

FIGURE 8.4: If the coefficient a is negative, we use the complements of the circular arcs defined by the control polygon.

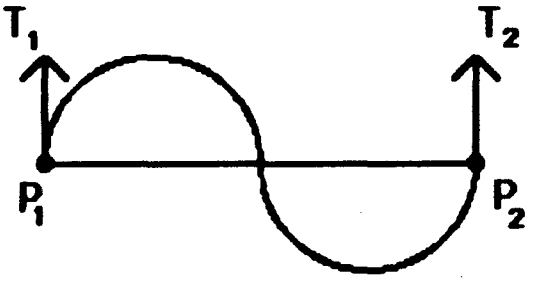

FIGURE 8.5: When $S \cdot T_1 = 0$, we use half circles.

8.3.1 Minimal representation

A PCC is completely defined by its *sampling points*, and the associated tangent directions, therefore, the *minimal representation* of a PCC that interpolates n sampling points requires $5n$ real numbers (3 for position and 2 for tangent direction per sampling point). One could add, to the minimal representation the coefficient a of each bi–arc. This would require $6n-1$ real numbers, but would save the cost of recomputing a each time the bi–arc is processed.

8.3.2 Control polygon

Smooth projections of PCC edges can be efficiently produced by our display procedure which recursively subdivides the control triangle of each arc into two equal parts until the projection of the control triangle on the screen is sufficiently small or almost linear. The computation of the two new control triangles is simple and economical — see Chapter nine.

Given the control polygon (P_1, B_1, B_2, P_2) of a bi–arc, the two control triangles (P_1, B_1, M) and (M, B_2, P_2) can be simply obtained by computing $M = \frac{B_1+B_2}{2}$. Therefore, PCC representation in terms of the control polygons seems attractive. It is obtained from the *minimal representation* described above, by computing for each pair of sampling points (P_{i1}, P_{i2}) the coefficient "a_i", and by generating the associated control polygon $(P_{i1}, B_{i1}, B_{i2}, P_{i2})$. Since P_{i2} is also the starting point of the control polygon for the next bi–arc, we need to store only 3 points per bi–arc (plus the last intersection point of the PCC), which amounts to $3n-2$ points, or $9n-6$ real numbers. (Cases where the coefficient a is negative, or where the arcs are half–circles may be detected and treated separately, but do not require additional storage.)

The redundancy of the above scheme can be reduced if one does not store the sampling points. For each bi–arc (P_1, B_1, B_2, P_2), but the first and the last one, one would only store the vertices B_1 and B_2. This approach only requires $6n$ real numbers for representing a PCC that interpolates n sampling points. Unfortunately this *control polygon* representation does not contain an explicit representation of the sampling points, which can only be obtained *incrementally* by traversing the whole curve as follows. Let suppose that we represent the PCC by the control polygon

$C_1, C_2, \ldots, C_{2n}$ (see Figure 8.6). The control polygon for the first bi–arc $(P_{11}, B_{11}, B_{12}, P_{12})$ is obtained by:

$$\begin{aligned} P_{11} &= C_1 \\ B_{11} &= C_2 \\ B_{12} &= C_3 \\ a_1 &= \|C_2 - C_1\| \\ l_1 &= \|C_4 - C_3\| \\ P_{12} &= C_3 + a_1 \frac{C_4 - C_3}{l_1} \end{aligned}$$

Then we compute the control polygon $(P_{21}, B_{21}, B_{22}, P_{22})$ for the second bi–arcs as follows:

$$\begin{aligned} P_{21} &= P_{12} \\ B_{21} &= C_4 \\ B_{22} &= C_5 \\ a_2 &= l_1 - a_1 \\ l_2 &= \|C_6 - C_5\| \\ P_{22} &= C_5 + a_2 \frac{C_6 - C_5}{l_2} \end{aligned}$$

and so on, for all bi–arcs.

The need for an incremental evaluation of the sampling points is not a major drawback for display algorithms, which process the curve sequentially.

8.3.3 Explicit trigonometric representation

Control polygon representation is not convenient for the analytic computation of the intersections of the arcs with standard surfaces. For such applications, we generate the control triangle for each arc and convert it into a standard trigonometric representation of a circular arc in terms of its radius, its angle, and the associated rigid motion. Such conversions are discussed in chapter nine. The rigid motion is defined by 6 numbers, but usually is represented by the corresponding 3 by 4 matrix. Therefore the *explicit trigonometric* representation of a PCC that interpolates n sampling points (i.e., $2(n-1)$ arcs) requires $28(n-1)$ real numbers.

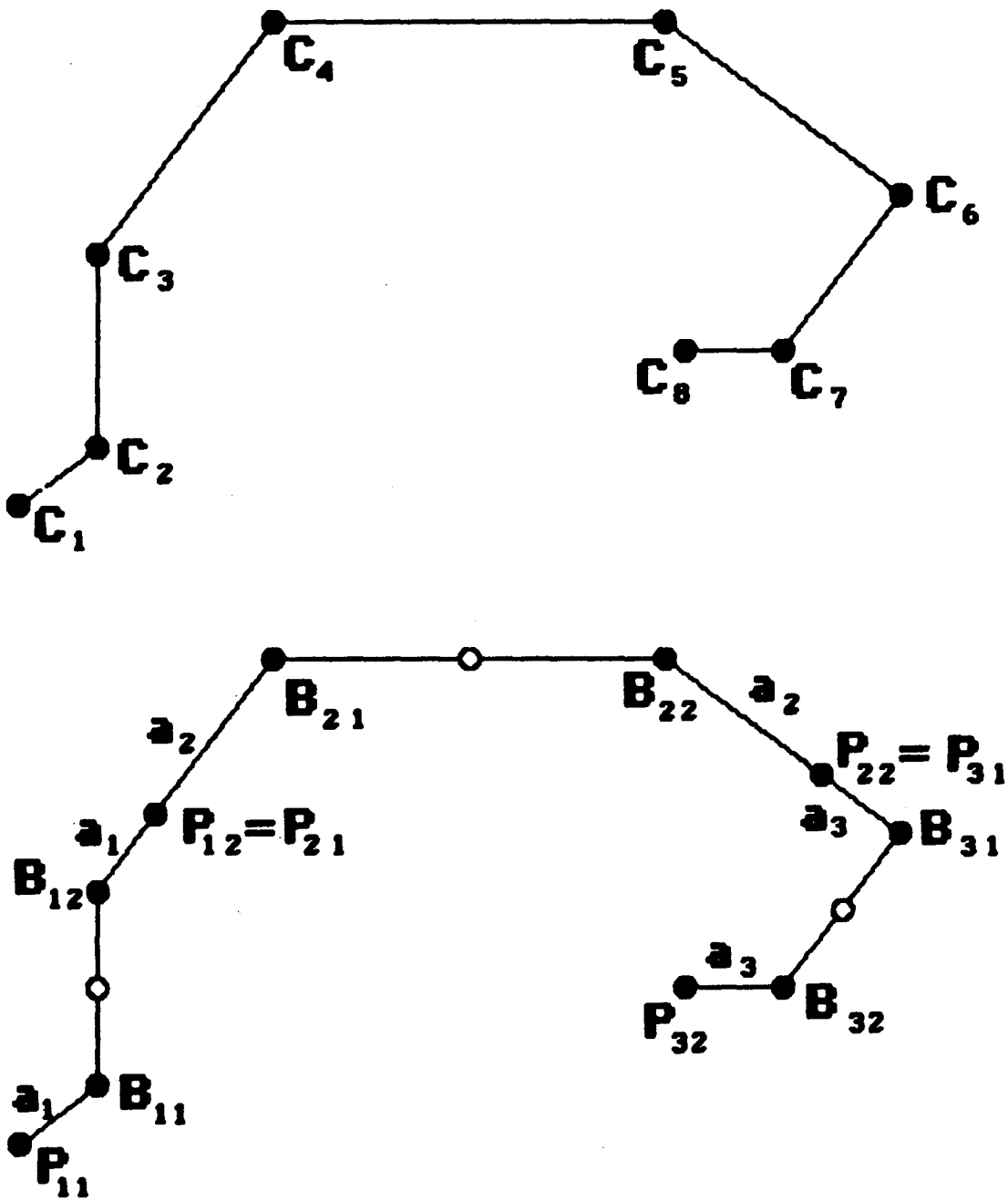

FIGURE 8.6: The PCC is represented by its control vertices C_i, from which we can compute the control polygons of the individual bi-arcs.

8.3.4 Representation of curve segments

The result of curve/solid classification is usually a list of segments (subsets of a curve). A convenient way to represent such segments is to add a list of parameter intervals to a parametric representation of the curve.

To use this approach we need to infer a parameterization $C(t)$ for our PCC representation. The most convenient parameterization is in terms of arc length. It can be easily obtained from the trigonometric representation of the PCC, which contains explicitly the radius and angle of each arc.

8.3.5 A convenient representation

In order to achieve a compromise between time and storage costs we chose the following representation for a PCC. For each sampling point P_i we store the three coordinates of the point, the three coordinates of the associated unit tangent, the coefficient a_i, and the parameter t_i such that $C(t_i) = P_i$. Conversion into explicit trigonometric representation is done only when needed for segment/surface intersection or point/edge projection.

8.4 INFERRING TANGENTS

To interpolate an ordered set of sampling n points P_i $(i = 1 \ldots n)$ by a PCC, the tangents T_i of the curve at the points P_i must be known. When the sampling points lie on a known curve or on the intersection of two surfaces, the exact associated tangents can be computed. But in other applications (free form curve design or profile edge approximation) only the sampling points are known, and tangents must be approximated. points P_{l-1}, P_l, P_{l+1} define one circle C_l. The center O_l and radius r_l of the circle, as well as the normal N_l to the plane that contains the circle

can be computed as follows:

$$
\begin{aligned}
V &= P_{l-1} - P_l \\
W &= P_{l+1} - P_l \\
N_l &= \|V \times W\| \\
R &= \frac{\|V\|^2}{2}(W \times N_l) + \frac{\|W\|^2}{2}(N_l \times V) \\
O_l &= P_l + R \\
r_l &= \|R\|
\end{aligned}
$$

Given the circle C_l lying in the plane normal to N_l, the tangent $TANG(C_l, S)$ to C_l at a point S is $N_l \times (O_l - S)$, where O_l is the center of the circle. The tangent T_l associated with the sampling point P_l could be approximated by $TANG(C_l, P_l)$; however this method is too local and can produce somewhat "unnatural" waves. Much better results can be obtained by using a weighted average of tangents to three circles constructed over consecutive points (Figure 8.7):

$$T_l = w_{l-1}TANG(C_{l-1}, P_l) + w_l TANG(C_l, P_l) + w_{l+1}TANG(C_{l+1}, P_l)$$

for $l = 3 \ldots n-2$ and where C_j is the circle passing by the points P_{j-1}, P_j, P_{j+1} for $j = 2 \ldots n-1$. For non–cyclic curves, tangents T_1, T_2, T_{k-1}, and T_k are inferred from only one or two circles.

When the weights w_j are equal, a smooth curve with little curvature variations is obtained. In order to approximate correctly straight lines defined by three or more colinear points, we make the weights proportional to the corresponding radii: $w_l = r_l$, which produces infinite weights for colinear points (Figure 8.8). Another weight formula is proposed in [Harada 82], where the coefficients w_l are slightly more complicated, but do not seem to produce better results than our approach.

8.5 APPLICATIONS OF PCC'S

PCC's have many applications. We shall only illustrate a few of them here.

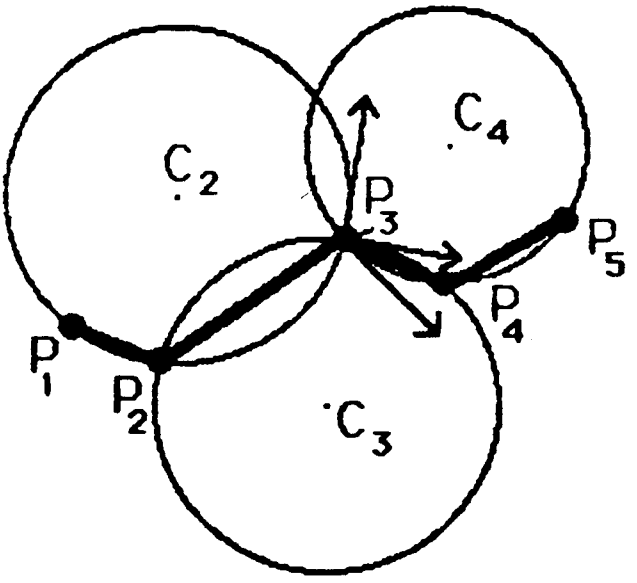

FIGURE 8.7: Tangents are weighted averages of tangent directions induced by three circles.

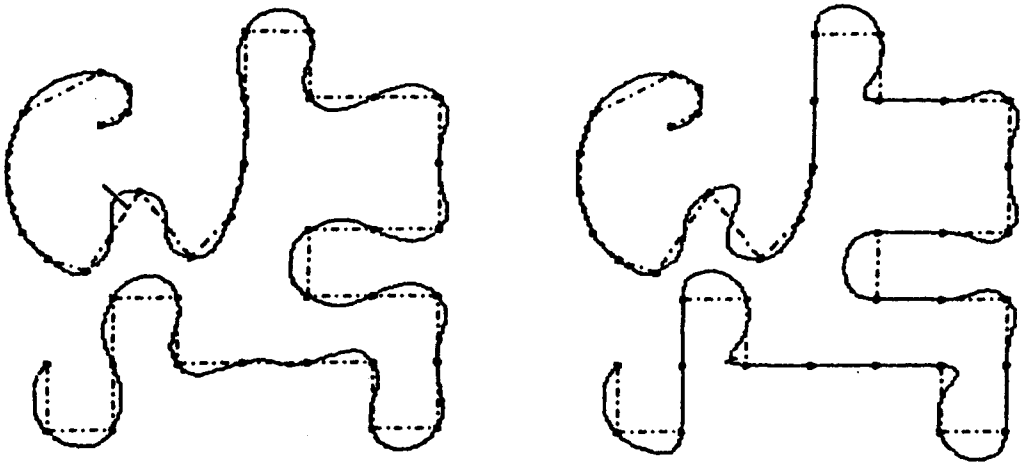

FIGURE 8.8: Approximating tangents using equal weights produces a smooth curve (left). To interpolate three colinear points with a straight line we introduce weights that are proportional to the radii of the corresponding circles (right).

8.5.1 Approximations of torus edges

Exact intersection–edges of a torus with any other standard surface are very expensive to generate, represent and classify. We propose to address this problem by using PCC approximations for such edges.

8.5.2 Support of offsetting operations

Canal faces, which are necessary to model the boundary of offset solids, can be approximated by a set of smoothly joined toroidal or cylindrical faces. This requires that their spine be approximated with PCC's. In addition, the use of PCC approximations for intersection–edges simplifies set membership classification with respect to solids defined in CSGO. Our experimental solid modeller approximates all intersection–edges and spines with PCC's.

8.5.3 Two–dimensional contouring

To produce illustrations for offsets, the author developed an experimental system for two dimensional contour definition. The contour is specified and modified graphically in terms of control points which define line segments, fillets for corners, or points that will be interpolated with a smooth PCC. Such contours contain only circular or linear segments and therefore are closed under n–offsetting (see Chapter four). These contours can be viewed as boundaries of two dimensional solids. Offsets of such solids can be used to drive numerically controlled cutting tools in computer aided manufacturing systems. Recursive offsetting (Figure 8.9) can be used as a basis for pocketing algorithms. We conjecture that it could also be used for the automatic generation of two–dimensional meshes for finite element analysis; such meshes would have the advantage of following the geometry of the boundary.

8.5.4 Three–dimensional curves

PCC's can be used for the design of free form curves. Interpolated control points of PCC's can be used in the same way as control points of B–splines to specify and modify the shape of the curve. PCC's retain the important local control feature of B–splines and interpolate control points.

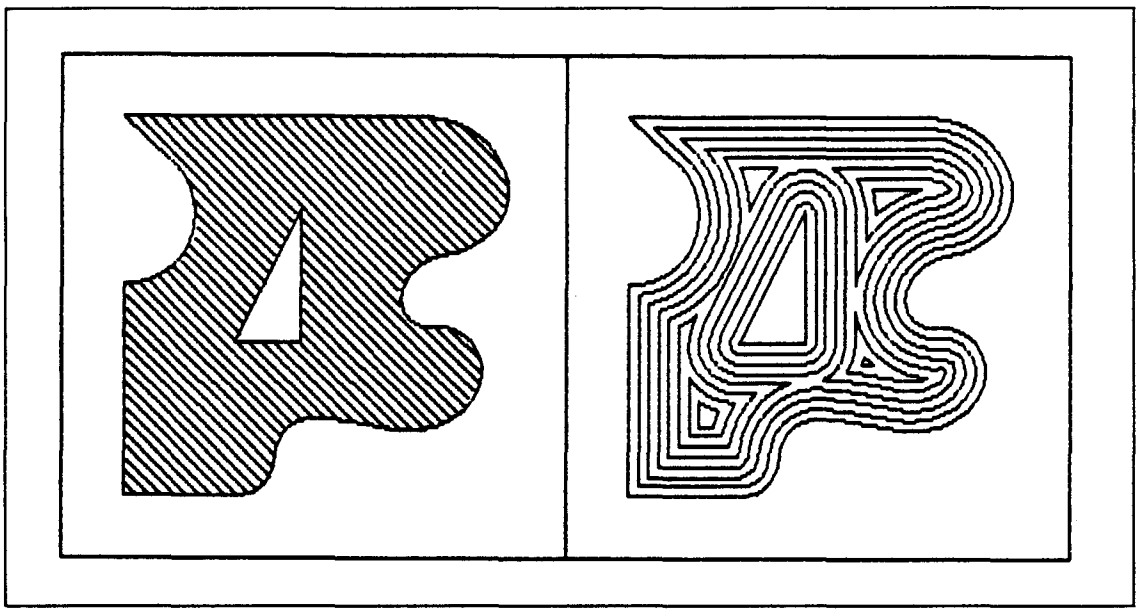

FIGURE 8.9: The contour (left) can be recursively contracted (right). The resulting family of contours can be used in pocketing or meshing algorithms.

8.5.5 Cutter path modelling

PCC's also can be used to define cutter trajectories for machine tools that allow linear and circular trajectories in space.

8.5.6 Sweeps

Combining PCC 2–D contours with PCC 3-D curves, one can incorporate sweeps into solid modellers that support only the standard surfaces. Sweeping along a PCC spine a constant cross–section bounded by contours that are piecewise linear or circular produces solids whose boundary is composed of standard patches, because the sweep can be decomposed into rotational sweeps (when the corresponding spine segment is circular) and translational sweeps (when the corresponding spine segment is linear).

Classification on such sweeps could be organized as follows:

- To classify a point P, one computes the normal projections P_i of P on the spine, then classifies P against the two dimensional contours, whose position and orientation are defined by each P_i. The two di-

- To classify a curve, one computes all the intersections of the curve with the set of patches that bound the sweep, and then classifies an intermediate point of each curve segment defined by two consecutive intersecting points.

8.5.7 Wire frame display

PCC's offer an economical representation of smooth approximations for edges. Such a representation could be stored in display devices to allow realtime transformations. Recursive subdivision of the control triangle (described in chapter nine) could presumably be used to drive micro–code or hardware implementation of display procedures.

Hidden–edge detection requires the intersection of edge projections in the picture plane. If edges are approximated with PCC's, such intersections are reduced to the intersections of ellipses and lines.

CHAPTER IX

COMPUTATION WITH PCC'S

We report in this chapter the mathematical results that were used in our experimental modeller to implement most of the algorithms that operate on PCC's. We derive formulae for converting PCC representations, for evaluating point/edge and point/face minimum distance, and for computing arc/surface intersections.

9.1 REPRESENTATION CONVERSION

Let C be a circle of radius r centered at the origin and lying in the XY plane. We can represent a circular segment in C by the trigonometric parametric expression

$$P(u) = O + r\cos(u)X + r\sin(u)Y$$

For $u \in [-t, t]$, the segment is positioned symmetrically around the X axis, and the half–angle is t (Figure 9.1). To define an arc at an arbitrary position and orientation in space, one associates with the arc a rigid motion represented by a 3×4 matrix RM. Points on the arc, at its final position, are defined by $RM(P'(u))$, where $P'(u)$ is a homogeneous representation of $P(u)$, obtained by adding a fourth coordinate equal to 1. The triple (r, t, RM) defines the *trigonometric representation* of the arc.

Procedures that compute PCC approximations produce a set of circular arcs represented by their control triangles. The isosceles non degenerate triangle (P_1, P_2, P_3) defines a unique circle and splits it into two complementary arcs.

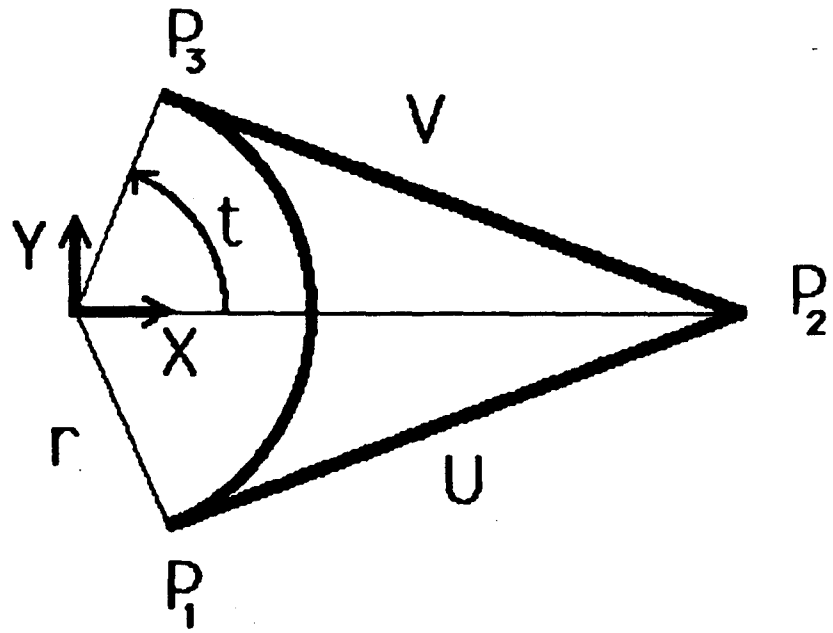

FIGURE 9.1: The arc can be represented in its parametric form or by the control polygon (P_1, P_2, P_3).

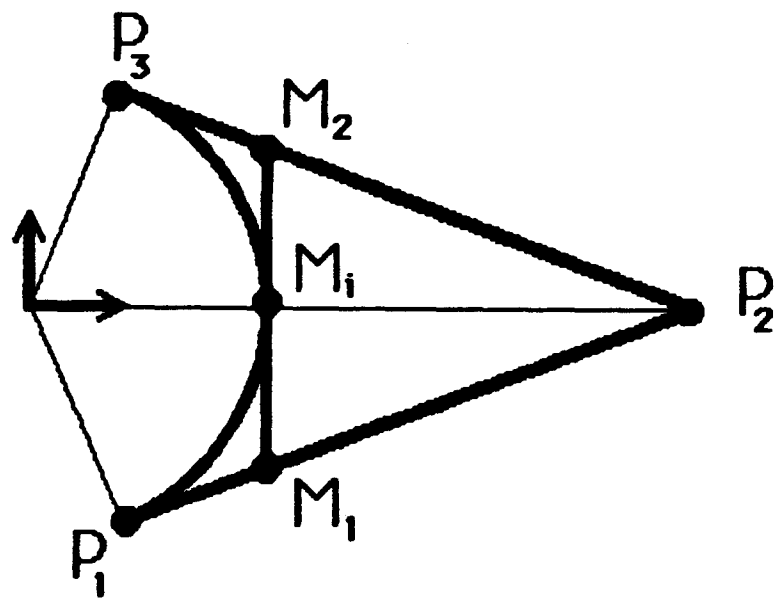

FIGURE 9.2: The arc defined by (P_1, P_2, P_3) is subdivided into two arcs defined by (P_1, M_1, M_i) and (M_i, M_2, P_3).

Usually we are only interested in the smaller arc (less than half a circle), but in certain cases, achieving smoothness of the PCC interpolation requires the use of the larger arc, which may be distinguished by some additional flag or sign, but is represented by the control triangle of its complement. The trigonometric representations of an arc and of its complement can be simply computed one from the other, and therefore we shall deal only with arcs that are less than half a circle.

The special case of semi–circles should also be treated separately. They can be represented by a control triangle, in which the tip point is replaced by a point at infinity.

For some applications (curve/surface intersection for example), we need the trigonometric representation (r, t, RM) of an arc. It can be evaluated from control triangle (P_1, P_2, P_3). by the following sequence of assignments:

$$
\begin{aligned}
U &= P_1 - P_2 \\
V &= P_3 - P_2 \\
Y &= P_3 - P_1 \\
X &= U + V \\
l &= \|X\| \\
a &= \|U\| \\
h &= \|Y\| \\
r &= \frac{ah}{l} \\
t &= \pi/2 - \arctan(h/l) \\
O &= P_2 - 2a^2 X \\
X &= -\frac{X}{l} \\
Y &= \frac{Y}{h} \\
Z &= X \times Y \\
RM &= (X, Y, Z, O)
\end{aligned}
$$

The inverse conversion is very simple. Given a trigonometric representation (r, t, RM), the points of the corresponding control triangle

(P_1, P_2, P_3) are computed as follows:

$$
\begin{aligned}
c &= r\cos t \\
s &= r\sin t \\
d &= r^2/c
\end{aligned}
$$

$$
(P_1 \quad P_2 \quad P_3) = (RM)\begin{pmatrix} c & c & d \\ s & -s & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}
$$

9.2 ARC SUBDIVISION

For display or classification purposes, it is often useful to be able to subdivide an arc into two parts. Representation in terms of control triangles is well suited for such subdivision. Given a control triangle (P_1, P_2, P_3), we produce the two control triangles (P_1, M_1, M_i) and (M_i, M_2, P_3) (see Figure 9.2) of the two half-arcs by the following sequence of assignments:

$$
\begin{aligned}
U &= P_1 - P_2 \\
V &= P_3 - P_2 \\
Y &= P_3 - P_1 \\
a &= \|U\| \\
h &= \frac{\|Y\|}{2} \\
m &= \frac{a}{a+h} \\
M_1 &= mU + P_2 \\
M_2 &= mV + P_2 \\
M_i &= \frac{M_1 + M_2}{2}
\end{aligned}
$$

For simplicity we assume that $a + h \neq 0$. Degenerate arcs corresponding to $a + h = 0$ are treated separately.

9.3 POINT EDGE PROJECTION AND DISTANCE

In our experimental modeller, described in chapter ten, we need routines to compute the normal projection H_i of a point P on PCC's. The normal projections of P on the linear or circular arcs of the PCC, can be easily computed from the coordinates (x, y, z) of the point P in the origin–centered coordinates system of the arc.

Linear arcs in their origin–centered coordinate systems correspond to the Z–axis, and therefore the only normal projection is $H = (0, 0, z)$. The minimum distance between P and the line is $\sqrt{x^2 + y^2}$.

For the circle of radius r, centered around the origin, and lying in the XY plane, if P is not on the Z–axis there are two normal projection points H_1 and H_2, having for coordinates $(qx, qy, 0)$ and $(-qx, -qy, 0)$, where $q = r/\sqrt{x^2 + y^2}$. The minimum distance between P and the circle is $\|P - H_1\|$. If P is on the Z–axis, all the points of the arc are at distance $\sqrt{r^2 + z^2}$ from P.

9.4 LINE/SURFACE INTERSECTION

To classify PCC's we need to compute their intersections with standard surfaces. Such computation can be divided into sub–problems of finding the intersection of a simple arc with the surface.

Intersections of the line $L(t) = P + tD$ with the surface of implicit equation $F(x, y, z) = 0$ can be obtained by replacing x, y, and z with the coordinates of a point on the line, and by solving for t the resulting polynomial equation.

9.5 CIRCLE/SURFACE INTERSECTION

The same method applied to the circle/surface intersection requires additional work in order to obtain a polynomial form for the resulting equation. We develop here a method for deriving systematically such polynomial forms. As a result, we obtain a fourth degree polynomial for the circle/torus intersection.

9.5.1 Conversion of trigonometric expressions

In origin–centered coordinates, a point Q on a circle of radius r is defined parametrically by: $Q = [r\cos(\theta), r\sin(\theta), 0]$. Applying to Q a rigid motion M (corresponding to a matrix $[M] = [UVWO]$)[10] we obtain $P = [P_x, P_y, P_z]$ which can be expressed in vector form by:

$$P = rcU + rsV + O$$

where $c = \cos(\theta)$ and $s = \sin(\theta)$. The coordinates of P are of the form:

$$\begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} = rc \begin{pmatrix} U_x \\ U_y \\ U_z \end{pmatrix} + rs \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} + \begin{pmatrix} O_x \\ O_y \\ O_z \end{pmatrix}$$

where U_x, U_y, and U_z are the coordinates of U, and a similar notation is used for V, W, O, and P.

In order to find the intersection point between the circle and a quadric surface, we need to convert into polynomial forms equations that contain P_x, P_y, P_z and P_x^2, P_y^2, P_z^2. The intersection of the circle with a torus requires in addition the conversion of $\|P\|^2$ and $\|P\|^4$ to such polynomials. For all these cases, a fourth degree polynomial equation is sufficient. We developed a systematic approach to compute the coefficients of such polynomials.

All the circle/surface intersection problems mentioned above can be expressed as the solution of the equation $E = 0$, which contains squares of terms of the form: $x = cI + sJ + K$. Squaring such a term we obtain:

$$x^2 = c^2\|I\|^2 + s^2\|J\|^2 + 2sc(I \cdot J) + 2c(I \cdot K) + 2s(J \cdot K) + \|K\|^2$$

and using $t = \tan(\theta/2)$, $c = \frac{1-t^2}{1+t^2}$ and $s = \frac{2t}{1+t^2}$ we can reduce $(1+t^2)^2x^2$ to a polynomial in t by replacing all terms by their polynomial forms:

$$\begin{aligned}
c^2\|I\|^2(1+t^2)^2 &= (t^4 - 2t^2 + 1)\|I\|^2 \\
s^2\|J\|^2(1+t^2)^2 &= 4t^2\|J\|^2 \\
2cs(I \cdot J)(1+t^2)^2 &= 2(-2t^3 + 2t)(I \cdot J) \\
2c(I \cdot K)(1+t^2)^2 &= 2(-t^4 + 1)(I \cdot K) \\
2s(J \cdot K)(1+t^2)^2 &= 2(2t^3 + 2t)(J \cdot K) \\
\|K\|^2(1+t^2)^2 &= (t^4 + 2t^2 + 1)\|K\|^2
\end{aligned}$$

[10] We use homogeneous coordinates to transform points by rigid motions.

and sorting powers of t in the expression of $(1+t^2)^2x^2$ yields:

$$(1+t^2)^2x^2 = a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0$$

where

$$\begin{aligned} a_4 &= \|K-I\|^2 \\ a_3 &= 4J\cdot(K-I) \\ a_2 &= 4\|J\|^2 + 2(K-I)\cdot(K+I) \\ a_1 &= 4J\cdot(K+I) \\ a_0 &= \|K+I\|^2 \end{aligned}$$

We introduce a matrix notation, where $[A(x^2)]$ defines the vector of coefficients a_i, which depend on I, J, and K, as indicated above: $[A(x^2)] = [a_4\ a_3\ a_2\ a_1\ a_0]$. The polynomial becomes:

$$(1+t^2)^2x^2 = [A(x^2)]\begin{pmatrix} t^4 \\ t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

The coefficients a_i can be efficiently computed by the following sequence of assignments:

$$\begin{aligned} K_{MI} &= K-I \\ K_{PI} &= K+I \\ a_4 &= \|K_{MI}\|^2 \\ a_3 &= 4J\cdot K_{MI} \\ a_2 &= 4\|J\|^2 + 2K_{MI}\cdot K_{PI} \\ a_1 &= 4J\cdot K_{PI} \\ a_0 &= \|K_{PI}\|^2 \end{aligned}$$

9.5.2 Applications to circle–surface intersection

We apply the above transformations to P_x^2, P_y^2, P_z^2, $\|P\|^2$, $\|P\|^4$ and to constants. For P_x^2 we have:

$$\begin{aligned} I_x &= rU_x \\ J_x &= rV_x \\ K_x &= O_x \end{aligned}$$

The corresponding matrix $[A]$ will be denoted $[A(P_x^2)]$. We obtain:

$$(1+t^2)^2 P_x^2 = [A(P_x^2)] \begin{pmatrix} t^4 \\ t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

Similar expressions are obtained for P_y^2 and P_z^2 using the notation $[A(P_y^2)]$ and $[A(P_z^2)]$ instead of $[A(P_x^2)]$.

In the same manner, we can express $\|P\|^4$, as a fourth degree polynomial. Using $U \cdot V = 0$ we get:

$$\|P\|^2 = 2rc(O \cdot U) + 2rs(O \cdot V) + \|O\|^2 + r^2$$

The intermediate coefficients are:

$$\begin{aligned} I_P &= 2r(O \cdot U) \\ J_P &= 2r(O \cdot V) \\ K_P &= \|O\|^2 + r^2 \\ K_{PI} &= \|O + rU\|^2 \\ K_{MI} &= \|O - rU\|^2 \end{aligned}$$

and denoting $[A]$ by $[A(P^4)]$:

$$(1+t^2)^2 \|P\|^4 = [A(P^4)] \begin{pmatrix} t^4 \\ t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

To convert $\|P\|^2$ into a similar polynomial form, one could use $\|P\|^2 = x^2 + y^2 + z^2$, which yields:

$$(1+t^2)^2 \|P\|^2 = [A(P^2)] \begin{pmatrix} t^4 \\ t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

where $[A(P^2)] = [A(P_x^2)] + [A(P_y^2)] + [A(P_z^2)]$. However, since $\|U\| = \|V\| = 1$ and $U \cdot V = 0$, the coefficients of $[A(P^2)]$ can be combined into a simpler vector form:

$$\begin{aligned} a_4 &= \|O - rU\|^2 \\ a_3 &= 4r(V \cdot O) \\ a_2 &= 2\|O\|^2 + 2r^2 \\ a_1 &= 4r(V \cdot O) \\ a_0 &= \|O + rU\|^2 \end{aligned}$$

To express the constant we develop $(1+t^2)^2$, and obtain:

$$(1+t^2)^2 = [A(1)] \begin{pmatrix} t^4 \\ t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

where $[A(1)] = [1\ 0\ 2\ 0\ 1]$.

9.5.3 Circle–cylinder intersection

The equation of the cylinder is: $x^2+y^2-a^2 = 0$. The circle is expressed in the local system of the cylinder. We multiply the cylinder equation by $(1+t^2)^2$:

$$(1+t^2)^2x^2 + (1+t^2)^2y^2 - (1+t^2)^2a^2 = 0$$

and substituting $(1+t^2)^2x^2$, $(1+t^2)^2y^2$, and $(1+t^2)^2a^2$ by the equivalent

$$\left([A(P_x^2)] + [A(P_y^2)] - a^2[A(1)]\right) \begin{pmatrix} t^4 \\ t^3 \\ t^2 \\ t \\ 1 \end{pmatrix} = 0$$

9.5.4 Circle–cone intersection

The equation of the cone is: $x^2 + y^2 - b^2 z^2 = 0$, where b is the tangent of the half–angle. This yields the polynomial:

$$\left([A(P_x^2)] + [A(P_y^2)] - b^2[A(P_z^2)]\right) \begin{pmatrix} t^4 \\ t^3 \\ t^2 \\ t \\ 1 \end{pmatrix} = 0$$

9.5.5 Circle–torus intersection

The equation of the torus is:

$$(\sqrt{\|P\|^2 - z^2} - b)^2 + z^2 = a^2$$

where a is the cross–section radius and b the radius of the spine. To eliminate the square root, we first obtain:

$$\|P\|^2 + b^2 - a^2 = 2b\sqrt{\|P\|^2 - z^2}$$

and by squaring both terms:

$$(\|P\|^2 + b^2 - a^2)^2 - 4b^2(\|P\|^2 - z^2) = 0$$

which is

$$\|P\|^4 - 2(b^2 + a^2)\|P\|^2 + 4b^2 z^2 + (b^2 - a^2)^2 = 0$$

Using previous results, the corresponding polynomial is:

$$\left([A(P^4)] - 2(b^2 + a^2)[A(P^2)] + 4b^2[A(P_z^2)] + (b^2 - a^2)^2[A(1)]\right) \begin{pmatrix} t^4 \\ t^3 \\ t^2 \\ t \\ 1 \end{pmatrix} = 0$$

9.5.6 Circle–sphere intersection

The equation of the sphere is:

$$\|P\|^2 - a^2 = 0$$

We multiply the equation by $(1+t^2)^2$ which yields:

$$\Big([A(P^2)] - a^2[A(1)]\Big) \begin{pmatrix} t^4 \\ t^3 \\ t^2 \\ t \\ 1 \end{pmatrix} = 0$$

However, the sphere does not require a fourth degree polynomial solution. Let C be the circle of radius r, centered at the origin O in the XY–plane. Let S be the center of the sphere of radius a. If the sphere intersect C then it also intersects the XY–plane in a circle C_s of radius $b = \sqrt{a^2 - C_z^2}$, and center S_{xy}, which is the projection of S on the XY–plane. We can rotate the coordinate system around the Z–axis by the angle θ formed between the vector OS_{xy} and the X–axis. In this new coordinate system, the two intersections I_1 and I_2 (if they exist) are placed symmetrically around the X–axis at an angle d from it (see Figure 9.3). The angle is given by

$$\cos d = \frac{r^2 + \|O_{xy}\|^2 - a^2}{2r\|O_{xy}\|}$$

The points I_i are given in this local coordinate system by:

$$I_1 = (r\cos(d), r\sin(d), 0) \text{ and } I_2 = (r\cos(d), -r\sin(d), 0)$$

The actual intersection points are obtained by rotating I_i by θ around the Z–axis.

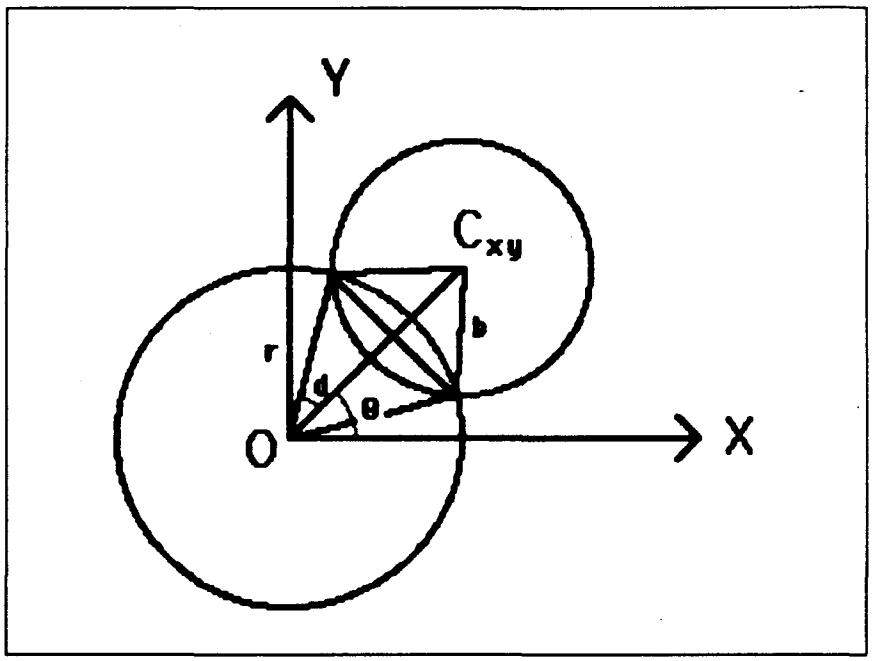

FIGURE 9.3: The intersection between a sphere and a circle can be calculated as a circle–circle intersection.

9.5.7 Circle plane intersection

For the plane $z = 0$, we obtain the following polynomial:

$$(1-t^2)rU_z + 2trV_z + (1+t^2)O_z = 0$$

ordering in t yields:

$$(O_z - rU_z)t^2 + 2trV_z + O_z - rU_z = 0$$

which (if $O_z \neq rU_z$) has for zeros:

$$t_1 = -A + \sqrt{(A^2-1)}$$
$$t_2 = -A - \sqrt{(A^2-1)}$$

where

$$A = \frac{V_z}{O_z/r - U_z}$$

Clearly there is no intersection if $|A| < 1$.

CHAPTER X

EXPERIMENTAL IMPLEMENTATION

To test and illustrate the concepts developed throughout this thesis, we designed and implemented an experimental solid modeller, SMOC, that incorporates offsetting operations in Constructive Solid Geometry. In order to support canal faces and intersection–edges of tori, all edges in SMOC are approximated with PCC's, and therefore we only have to support one edge type. We propose here a high level outline of our implementation of some of the important algorithms discussed in Chapter six. In order to help the reader understand the operations involved in those algorithms, we also describe some of the important representation schemes.

10.1 BOUNDARY EVALUATION

Given a solid represented in CSGO by the tree S, we create a dual representation for S by attaching lists of edges, vertices and patches to arguments of offset nodes. This is done incrementally by a recursive boundary evaluation procedure. We start at the node S; the routine returns the set of edges E, and the set of tentative faces F for the current node N. We outline below the steps that such a routine performs at each node. The specific actions depend on the type of the node.

- For *PRIMITIVE* nodes, we return E, the set of edges, and B, the set of faces (patches) of the primitive. The edges are marked as convex. The direction of the normals to patches of B are oriented towards the outside of the primitive.

- For *MOTION* nodes, through a recursive call, we evaluate the edges E_l and the patches F_l for the left son, transform them by the rigid motion associated with the current node, and return the result.

- For a *BOOLEAN OPERATOR* node, through recursive calls, we evaluate E_l and F_l for the left son L, and E_r and F_r for the right son R of the current node. If the node is a $-^*$ operator, we invert the concavity of the edges of E_r and the orientation of the patches of F_r. If the operator is $\bigcup^*$, we reject[11] from E_r all portions of edges that are in L, else we reject from E_r all portions of edges that are out of L. In the same manner, if the operator is $\bigcap^*$, then we reject from E_l all portions of edges that are out of R, else we reject from E_r all portions of edges that are in R[12].

  We also compute E_c, the set of intersection–edges[13] of each patch of F_l with each patch of F_r. If the operator is $\bigcup^*$, we mark those edges as concave, otherwise we mark them as convex. We reject from E_c all portions of edges that are not on L or not on R.

  We return patches, made of the union of F_l with F_r, and the set of edges, made by combining what remains from E_l, E_r, and E_c after the classifications mentioned above.

- For an *OFFSET* node, through a recursive call, we evaluate the edges E_l and the patches F_l for the left son L. We construct a set of vertices V_l from the ends of the edges of E_l, and replace the multiple versions of the same vertex by their centroid. We add to V_l the singular points of the patches of F_l that lie on L. Finally, we attach (E_l, F_l, and V_l) to the node L (it will be used later for PMC with respect to the current node).

  Given the offset distance r attached to the current node, we construct the set of patches F, which contains the n–offsets by r of the patches of F_l, the canal faces of radius r around the edges of E_l,

[12] The difference operator is considered implicitly in this formulation.

[13] Such intersection edges are PCC's generated by the method described in chapter seven.

We also build tentative intersection–edges E between all pairs of the patches of F and reject all edges of E that are not on the current node. We return F and E.

The edges of S returned by this routine can be used for wire–frame displays. The patches F, can be used for the generation of shaded pictures, either by *ray casting* or by *depth buffer techniques* [Rossignac 85]. This high level description of our boundary evaluation algorithm refers to several other procedures and uses various representations, which we shall discuss below.

10.2 PATCHES

This section describes how we represent and generate patches that constitute supersets of boundaries. Patches correspond to the faces of primitives or to n–offsets of other patches, edges or vertices. N-offsetting PCC edges yields piecewise cylindrical or toroidal surfaces. Patches are subsets of standard surfaces, and can be associated with a vector–valued parametric function $F(u,v)$. The surface containing a patch is defined by its type, its parameters and the associated rigid motion, which maps points of the surface in its origin–centered position into corresponding points of the surface in its final position. Patches are represented by their type, by a description of the associated surface (parameters, rigid motion), and by the limits (u_0, u_1, v_0, v_1) of the parameters u and v. Those limits define the extent of the patch. We list below the semantics of such representations for each type of patch.

- *Rectangle*: defined in the plane $z = 0$ by $F(u,v) = (u,v,0)$. There is no parameter. $u_0 = 0$, $v_0 = 0$, and u_1 and v_1 indicate the extent of the patch in the X and Y directions.

- *Disk*: defined in the plane $z = 0$ by $F(u,v) = (v\cos(u), v\sin(u), 0)$. There is no parameter. $u_0 = -\pi$, $u_1 = \pi$, $v_0 = 0$, and v_1 is the radius of the disk.

- *Cylinder*: defined in the surface $x^2 + y^2 = r^2$ by

$$F(u,v) = (r\cos(u), r\sin(u), v)$$

The only parameter is the radius r of the cylinder. $u_0 = -\pi$, $u_1 = \pi$, $v_0 = 0$, and v_1 is the length of the cylinder.

- *Sphere*: defined in the surface $x^2 + y^2 + z^2 = r^2$ by

$$F(u,v) = (r\cos(u)\cos(v), r\sin(u), -r\cos(u)\sin(v))$$

The only parameter is the radius r of the sphere. $u_0 = -\pi$, $u_1 = \pi$, $v_0 = -\pi/2$, and $v_1 = \pi/2$.

- *Cone*: defined in the surface $x^2 + y^2 = p^2 z^2$ by

$$F(u,v) = (v\cos(u)\sin(t), v\sin(u)\sin(t), v\cos(t))$$

The only parameter is the tangent p of the half-angle t of the cone. $u_0 = -\pi$, $u_1 = \pi$, $v_0 = 0$, and $v_1 = l\sqrt{1+p^2}$, where l is the length of the cone along the Z–axis.

- *Elbow*: defined in the toroidal surface $\left(\sqrt{x^2+y^2} - R\right)^2 + z^2 = r^2$ by

$$F(u,v) = R_z(u) \star T_x(R) \star R_x(-\pi/2) \star R_z(v) \begin{pmatrix} r \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

where $R_x(b)$ and $R_z(b)$ denote respectively the rotations by an angle b around the X–axis and around the Z–axis, and where $T_x(b)$ is a translation along the X–axis by distance b. $u_0 = -t$, $u_1 = t$ where t is the half–angle defining the extent of the elbow. $v_0 = -\pi$, $v_1 = \pi$.

10.3 PROJECTION ON STANDARD SURFACES

To determine the normal projections H_i of a point $P = (x,y,z)$ on a standard surface S, we assume that P is expressed in the origin–centered coordinate system of the surface.

For the *plane* $z = 0$, the only normal projection is $H(x,y,0)$.

For the *sphere* of radius r and center O (the origin): if $P \neq O$, then $H_1 = qP$ and $H_2 = -qP$ where $q = r/\|P\|$. If $P = O$, then all points of the sphere are at the same distance r from P.

If P is not on the Z–axis, the normal projections of P on the *cylinder* of radius r centered around the Z–axis are the points $H_1 = (xq, yq, z)$ and $H_2 = (-xq, -yq, z)$, where $q = r/\sqrt{x^2 + y^2}$. If P is on the Z–axis, the minimum distance from P to S is r, and is attained at all points that lie on the circle of radius r and center P, parallel to the XY plane.

Let K be a *cone* centered around the Z–axis, such that its tip is at the origin and its tip half–angle is t. If P is not on the Z–axis, the normal projections of P on K are the points

$$H_1 = \frac{z + pr}{r(1 + p^2)}(px, py, r)$$
$$H_2 = \frac{z - pr}{r(1 + p^2)}(-px, -py, r)$$

where $P = \tan(t)$ and $r = \sqrt{x^2 + y^2}$. The distance from P to the cone is $\|P - H_1\| = |r - zp|/\sqrt{1 + p^2}$. If $P = (0, 0, z)$, then $d(P, S) = |z|p/\sqrt{1 + p^2}$.

Let S be a *torus* of cross–section radius r, and having for spine, in the XY plane, the circle C of radius R and center O. If P is not on the Z–axis, to compute the normal projections of P on S, we first compute the normal projections P_1 and P_2 of P on C. The four projections of P on S are at distance r from P_i along the vectors $P - P_i$ for $i = 1, 2$.

10.4 PRIMITIVES

Patch representations are directly computed from primitives. A primitive P is represented by its parameters; for example, a cylinder can be represented by its radius and its length. Primitives at the leaves of the CSGO tree are assumed to be in their origin–centered position, which we chose so as to simplify computation. We support the following primitives (Figure 10.1): Block, sphere, cylinder, cone, elbow.

In, **on**, or **out** results of the point membership classification procedure of a point P with respect to a primitive S can be elegantly expressed by the results of two calls to a simpler routine: **pclass**(P, S), whose result is true if $P \in kS$ and false otherwise. Specifically:

$$P \in iS \quad \Leftrightarrow \quad \textbf{pclass}(P, S) \quad AND \quad NOT \quad \textbf{pclass}(P, \overline{S})$$
$$P \in \partial S \quad \Leftrightarrow \quad \textbf{pclass}(P, S) \quad AND \quad \textbf{pclass}(P, \overline{S})$$
$$P \in \overline{kS} \quad \Leftrightarrow \quad NOT \quad \textbf{pclass}(P, S) \quad AND \quad \textbf{pclass}(P, \overline{S})$$

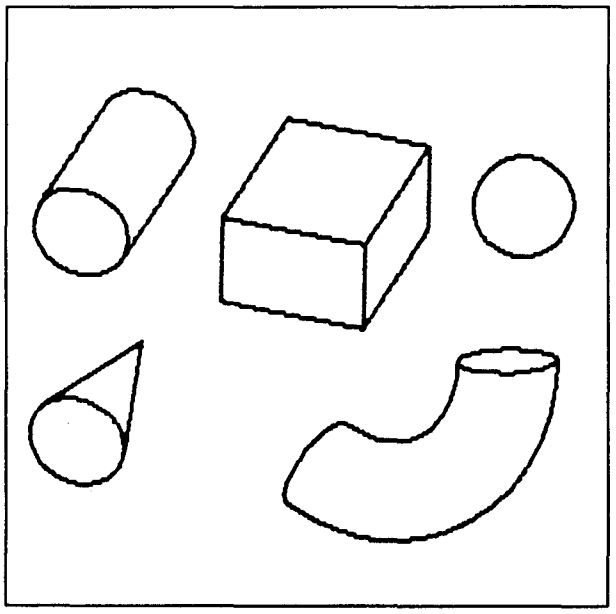

FIGURE 10.1: Standard primitives: cylinder, block, sphere, cone, toroidal elbow.

The combination of **pclass**(P, S) with **pclass**$(P, \overline{S})$ in a single routine is very simple and permits in certain cases to avoid evaluating twice the same expressions. For the sake of clarity, we shall only outline the simple version of **pclass**.

Each primitive, defined by a small number of constraints, corresponds to the intersection of half–spaces (except for the case of the torus–elbow). **pclass**(P, S) classifies the point P with respect to the intersection of versions of those half–spaces expanded by ϵ. **pclass**$(P, \overline{S})$ classifies P with respect to the union of the expanded versions of the complements of the half–spaces defining S. Therefore **pclass**$(P, \overline{S})$ is the complement of **pclass**(P, S) in which we use shrunk and not expanded versions of primitives. This modification can be simply performed by changing the sign of ϵ in the formulae used for point/half–space classification. We provide support for **pclass**$(P, \overline{S})$ by simply passing ϵ as a parameter to **pclass**. The formulae proposed below hold for both positive and negative epsilon, but **pclass** inverts the final result if $\epsilon < 0$, thus generating classification with respect to the complement of S.

This approach corresponds to classifying P against a primitive whose boundary has a thickness of 2ϵ. Choosing ϵ allows us to compensate for computer round–off errors and also provides a tool for dealing with errors introduced by our approximation schemes.

For each type of the primitive S, we present below the inequalities used in **pclass**(P,S) to classify against kS the point P, whose coordinates are (x,y,z).

- The *BLOCK* defined by its length l, height h, and depth d, contains the points that satisfy the six constraints:

$$-\epsilon < x < l + \epsilon$$
$$-\epsilon < y < h + \epsilon$$
$$-\epsilon < z < d + \epsilon$$

- The *SPHERE* defined by its radius r, contains the points that satisfy the constraint:

$$x^2 + y^2 + z^2 < r^2 + \epsilon$$

- The *CYLINDER* defined by its length l and radius r, contains the points that satisfy the three constraints:

$$x^2 + y^2 < (r + \epsilon)^2$$
$$-\epsilon < z < l + \epsilon$$

- The truncated *HALF-CONE* defined by its length l, and maximum radius r, contains the points that satisfy the three constraints:

$$x^2 + y^2 < \left(rz/l + \epsilon\sqrt{1 + \left(\frac{r}{l}\right)^2} \right)^2$$
$$-\epsilon < z < l + \epsilon$$

- The *ELBOW* (or truncated torus) defined by the radius r of its cross-section, and by the radius R and half-angle t of its spine (circular-arc), contains the points that satisfy the three constraints:

$$-t - \frac{\epsilon}{\sqrt{x^2+y^2}} < angle(x,y) < t + \frac{\epsilon}{\sqrt{x^2+y^2}}$$
$$z^2 + (\sqrt{x^2+y^2} - R)^2 < (r + \epsilon)^2$$

In these calculations we assume that $\epsilon \ll \sqrt{x^2+y^2}$, and we use the procedure *angle*(x,y) which returns the angle between the vector (x,y) and the X-axis.

10.5 BOUNDARY OF PRIMITIVES

We know exactly the boundary of primitives and use their exact faces in lieu of patches. The parameters, limits, and rigid motions of such patches can be simply derived from the parameters of the primitives. For instance, in the case of a cylinder of length l and radius r, we obtain three patches:

- The front disk with $v_1 = r$, and the rigid motion $M = T_z(l)$.
- The back disk with $v_1 = r$, and $M = R_y(\pi)$ to orient the normal out of the primitive.
- The cylindrical patch with r for parameter, and with $v_1 = l$. M is the identity transformation.

The edges of standard primitives are line segments or circles. For instance, the edges of the cylinder of radius r and length l are two circles of radius r, centered around the Z–axis and lying in planes $z = 0$ and $z = l$. To generate representations of such edges, we use routines that generate constant parameter curves for standard patches. For instance, the edges of the cylinder can be obtained as the v–curves $F(u, r)$ of the disks.

10.6 PMC

We propose a classification routine **class**(P, S), which returns a Boolean value that is TRUE if the point P is in the closure of a solid S defined in CSGO, and FALSE otherwise. As in the case of **pclass**, the routines **class**(P, S) and **class**$(P, \overline{S})$ can be combined to provide more efficient standard classification routines. **class**$(P, \overline{S})$ is implemented by passing S and a signed ϵ as parameters to **class**.

The procedure **class**(P, S) uses a recursive formulation. Let L and R denote respectively the left and the right sons (when they exist) of the current node N. Depending on the type of the current node, we perform the following computations:

- For *motion* nodes, we return **class**(P', L), where P' is the transformed version of P by the inverse of the rigid motion attached to N.

- For *primitive* nodes, we return **pclass**(P,N).
- For *offset* nodes, we return **oclass**(P,L,r), where r is the *signed* distance associated with the offset node. The procedure **oclass**(P,L,r) is described below.
- For nodes that represent *Boolean operations*, if the operation is $\bigcup^*$, we return (**class**(P,L) *OR* **class**(P,R)); if the operation is $\bigcap^*$, we return (**class**(P,L) *AND* **class**(P,R)); and if the operation is $-^*$, we return (**class**(P,L) *AND* **class**$(P,\overline{R})$).

10.7 CLASSIFICATION AGAINST OFFSETS

If $r > 0$, the procedure **oclass**(P,N,r) is used to find whether the point P is in or on the $N\uparrow r$. If $r < 0$, the procedure **oclass**(P,N,r) is used to find whether the point P is in or on $N\downarrow^* r$

If $r > 0$, the result of **oclass**(P,N,r) is **pclass**(P,N) *OR* (**dist**$(P,\partial N) < R+\epsilon$), where **dist**$(P,\partial N)$ returns the distance from P to the boundary of N. Such a distance is computed as explained in chapter six. If $r < 0$, the result of **oclass**(P,N,r) is **pclass**$(P,\overline{N})$ *AND* (**dist**$(P,\partial N) > R-\epsilon$).

The classification **oclass**$(P,\overline{N},r)$ of P against the offset by r of the complements of N is simply deduced from **oclass**(P,N,r) performed with a negative ϵ.

10.8 PCC CLASSIFICATION

The simplest approach for classifying a PCC curve C with respect to a solid S is to generate the explicit trigonometric representation for each arc of C, and classify the arc with respect to S. Arc classification, as discussed in chapter six, is done by segmenting the arc into subsets of constant classification, and by classifying an intermediate point in each segment. Arc segmentation is done by computing the intersections of the arc with all the patches of S.

Arc/surface intersection may require finding the roots of a fourth degree polynomial (see chapter nine). Computational costs are reduced by the following speed–ups.

The accuracy of the approximation for PCC is controlled by the *maximum size* s of the cells used in our parametric grid method (see Chapter seven). It follows that the distance between the two end–points of an arc is less than $\sqrt{2}s$. Our approach does not guarantee to detect intersection edges that are smaller than s, therefore, we could also ignore edge classification changes that are local to a single cell. This tolerance can be used as follows. If the two end–points of a bi–arc lie on the same side of a surface A one needs not produce the trigonometric representations of both arcs and compute their intersection with A. The speed-up can be carried even further. To classify a PCC curve C with respect to a CSGO solid S we classify all sampling points of C first. If between two consecutive sampling points the classification does not change, we consider that the bi–arc defined by these two points and the associated tangent directions has the same classification as both points. (Since the sampling points lie on the edge and we know the exact edge tangent at these points, edge–neighborhood evaluations as discussed in chapter six can be supported.) If two consecutive sampling points have a different classification with respect to the solid, the trigonometric representations of the two arcs are constructed and classified by the method described in chapter eight. This involves computing intersections of each arc with the host surfaces of patches of S, and classifying mid–points of the resulting segments.

To represent classification results, we use the edge parameterization described in chapter six. A parameter list can be attached to each arc, or to the whole edge. Vertices are obtained as edge points that correspond to ends of parameter intervals.

10.9 EXPERIMENTAL RESULTS

Our experience with PCC approximations for edges was successful. Only a small number of segments is necessary to approximate complicated edges with PCC's that cannot be visually distinguished from the exact edges.

Timing results are promising. Although speed was not an objective for our simple implementation, we found that PCC–based algorithms

compare advantageously with other non–analytic schemes. Wireframing a simple object with pixel accuracy (Figure 10.2) takes only a few seconds, and edges of more complex objects (Figure 10.3) can be generated and classified in a couple of minutes.

Unfortunately, performance decreases significantly when offsetting operations are used. Imagine expanding by r a simple solid S that has p patches, v vertices, and e edges, each being approximated with b bi–arcs. Offsetting S will produce a solid whose boundary is contained in $p+v+be$ patches. To generate the edges of $S\uparrow r$ one needs to compute all pairwise intersections of its patches. This requires computing the intersections of the generators of each patch with half of the other patches, on average. For example, a large n–offset patch might contain a hundred generators, and all of them must be intersected with be toroidal patches obtained by n–offsetting the edges of S. Similarly, shading such offset solids (see Figure 10.4), with a depth–buffer technique adapted to CSGO [Rossignac 85], requires classifying hundreds of thousands of points with respect to the offset solid. This implies computing the normal projections of all these points on the arcs of the PCC approximation of each edge. Our experimental implementation shows that the efficiency of boundary evaluation for offset and blended solids must be improved significantly to achieve speeds suitable for interactive operations.

Costs of computing edges could be reduced in the following two ways: (1) use boxes around patches (or other efficiency–enhancement techniques normally used in modellers) to avoid computing intersections of patches that are far one from another, and (2) while computing the intersections of two patches, use the generators of the smallest one. Similarly, costs for point/edge projection can be reduced by using rejection tests based on the convex hulls (triangles) of the arcs of the PCC approximation of the edge. And several other speed–up techniques undoubtedly deserve investigation.

Although we did not test such speed–ups, we fear that boundary evaluation costs will remain high. However, since to the best of our knowledge, no other method for supporting offsetting operations on complex solids is currently available, our simple implementation has the merit of providing an experimental tool and a reference for further development of more efficient systems.

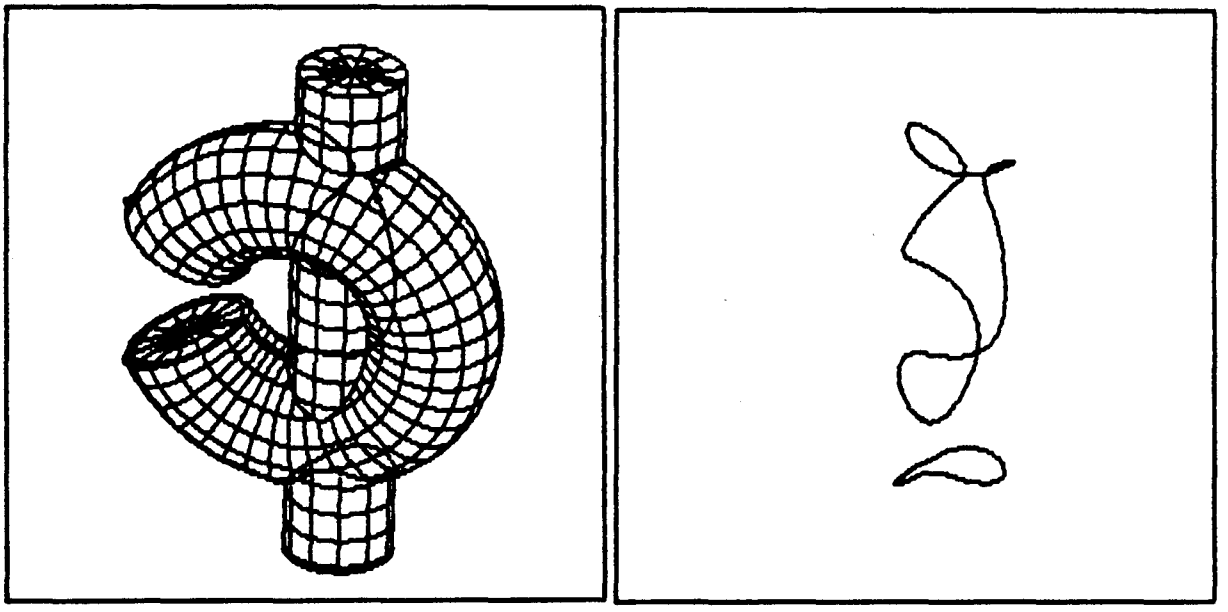

FIGURE 10.2: The intersection edge of a toroidal and a cylindrical face (left) was generated (right) in three seconds.

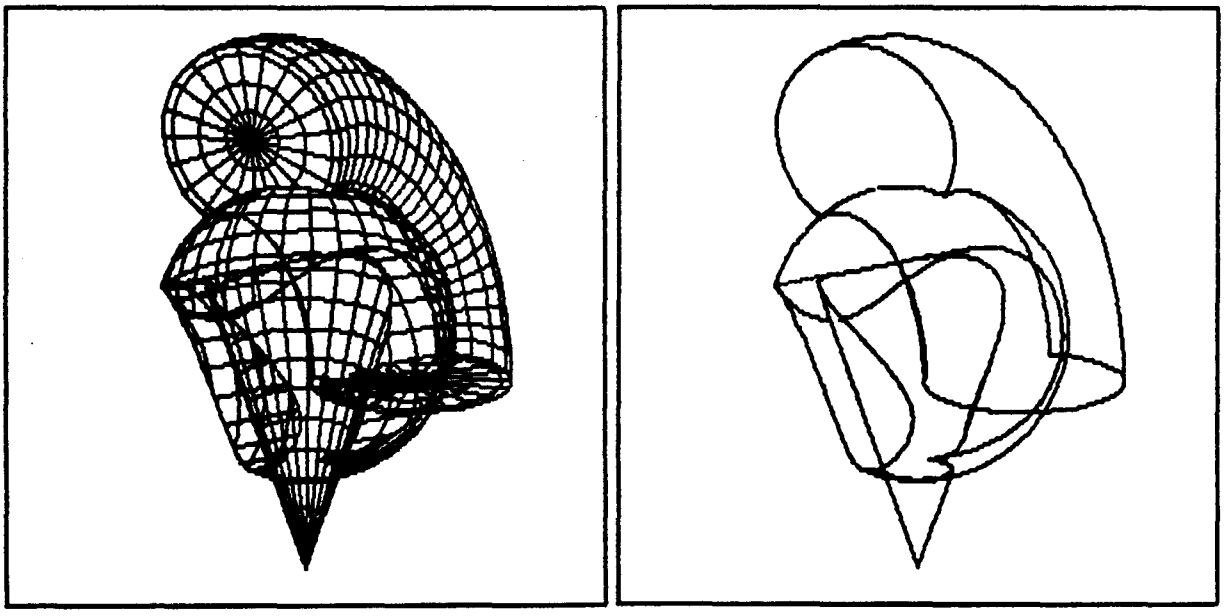

FIGURE 10.3: More complicated wireframes require longer, but acceptable delays.

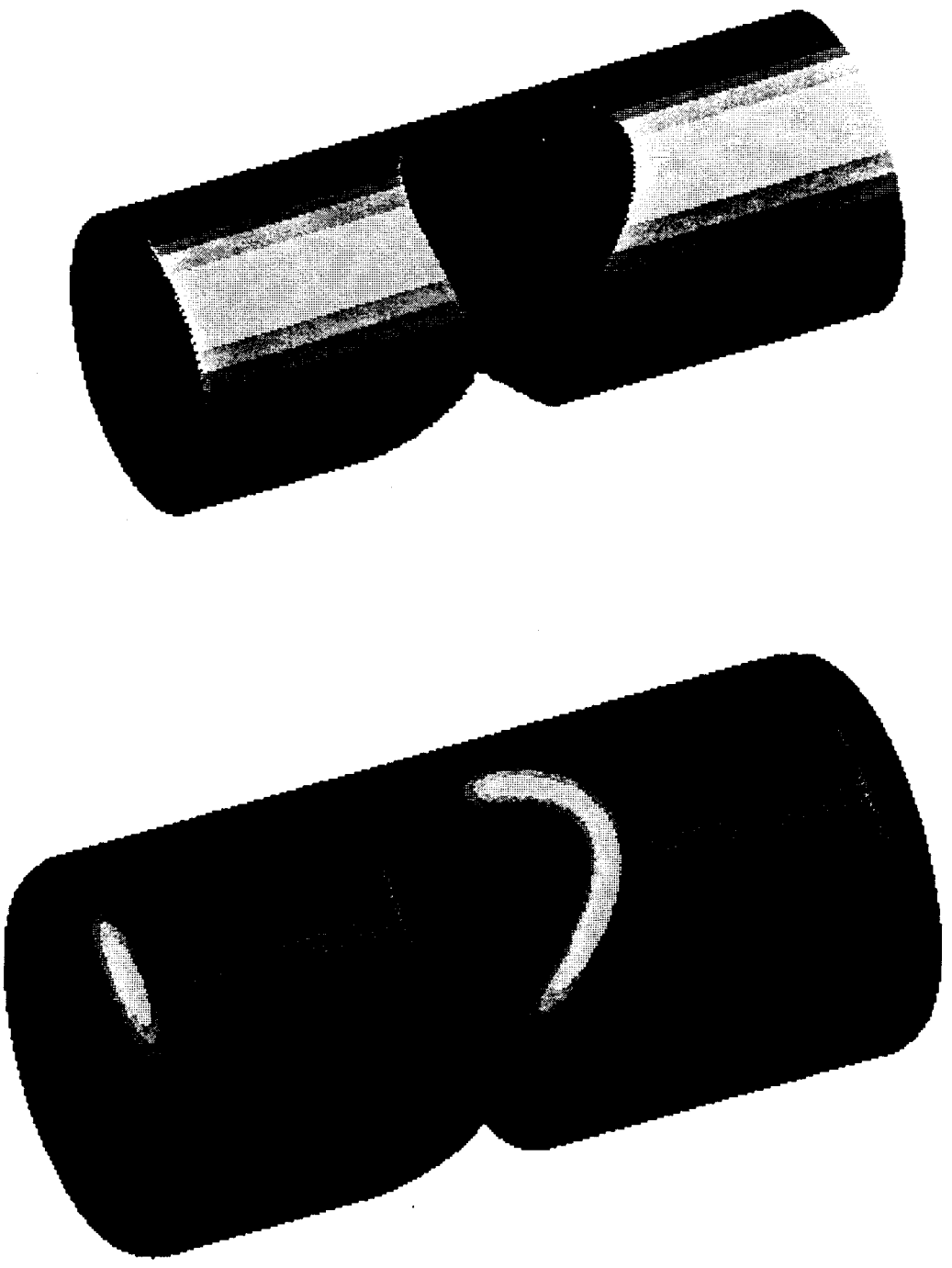

FIGURE 10.4: The difference of two cylinders (top) is expanded (bottom).

Boundary evaluation procedures for tori are currently being implemented in the PADL–2 system at the Production Automation Project, by using PCC approximations and the algorithms described here. The PADL–2 implementation should produce useful information on efficiency issues.

CHAPTER XI

CONCLUSION

11.1 BLENDING

The initial objective of this work was to attack the under–specified problem of providing blending facilities in modern solid modellers. After studying the semantics of blend specification, we decided to focus our attention on constant radius blends because they are the most common in mechanical parts.

Instead of trying to develop rules for inferring user's intentions from ambiguous specifications of blends, we decided to provide the user with a simple and powerful tool for describing solids that may contain complex blends, but are mathematically well defined. This was achieved by integrating global blending operations in a Constructive Solid Geometry (CSG) specification. Our approach has three major advantages:

— Models of blended solids (or even parametrized families of solids) can be specified in a compact, user accessible language, allowing simple archiving and modification of part descriptions, and providing a clean and flexible interface between the solid modeller and application programs.

— Specification of complex blends, or of fillets on sequences of edges (e.g., the concave edges of a pocket), can often be reduced to simple operations. Sequencing, successive blending, and Boolean operations offer wide possibilities for describing unambiguously interfering blends and for trimming the blends near complicated vertices.

— Application algorithms for CSG, based on divide and conquer methods, can be extended to blended solids. CSG–based algorithms for set membership classification are simpler and often more robust than their counterparts in boundary–based systems.

11.2 OFFSETTING

We discovered that constant radius blending operations can be modelled by a combination of two offsetting operations (expanding and shrinking). Offsetting per se has numerous applications in solid modelling. For example, automatic planning of cutter paths in NC machining or of collision free robot motions rely on offsetting. Therefore we decided to study offsets and provide the theory and algorithms to support offsetting operations in CSG.

In two dimensions, offsetting areas that are bounded by circular and linear edges can be done without domain extension. Unfortunately, in three dimensions, offsetting Boolean combinations of standard primitives (blocks, spheres, cylinders, cones and tori) produces solids that are partly bounded by canal faces, not supported in current modellers.

11.3 PCC'S

Providing exact representations and algorithms for canal surfaces and their intersection edges seemed very difficult, if not impossible, and we opted for an approximation technique. Our method is based on a new curve interpolation technique, which approximates intersection–edges by smooth piecewise linear or circular curves (PCC's). Approximating the spines of canal surfaces with PCCs implies that canal surfaces are approximated by smoothly joined (G^1) pieces of cylinders and tori. We developed algorithms to generate PCC approximations for intersection–edges between any two standard faces, and therefore also for torus intersection–edges. PCC approximations greatly facilitate the computations required to evaluate edge/surface intersections and minimum point/solid distances. PCC's can also be used in CAD/CAM to design smooth free–form space curves, and to model, without domain extension, sweeps of a constant cross–section along any trajectory in space.

11.4 EXPERIMENTAL IMPLEMENTATION

To test and illustrate the theory and algorithms developed in this thesis we designed and implemented an experimental solid modeller based on a CSGO representation. It supports all standard primitives combined through Boolean and offset operations. To support the torus and canal faces necessary to model boundaries of offset solids, we approximate all edges with PCC. This considerably reduced the amount of code necessary to compute and process intersection edges.

To improve performance and reliability, our approach must be enhanced in two ways.

1) We must devise efficient ways to estimate a tight bound for the maximum distance between points on a linear or circular arc and a standard surface. (As a first cut one might try to detect whether a maximum is attained for interior points of an arc.)

2) We need efficient rejection tests to speed–up tentative edge generation and point membership classification. Standard efficiency–enhancement techniques for geometric algorithms seem appropriate.

11.5 SUMMARY OF CONTRIBUTIONS

Finally, our main contributions can be summarized as follows.

- a CSG compatible approach to blending through global blending operations,
- precise definitions of solid offsetting operations and a study of their properties,
- the study of the boundaries of offset solids (canal surfaces),
- the definition of blending through offsetting operations,
- algorithms for boundary evaluation and other applications in CSGO,
- PCC approximations for space curves,

- a cell based sorting method for matching intersection points in edge approximation algorithms,
- a simplified approach to the computation of curve/surface intersections.

APPENDIX A

CANAL SURFACES

We defined a canal face as the *sweep of a circular cross–section* of radius r (called *generating circle*) along a smooth *spine* C. Canal faces are also included in *canal surfaces* of spheres of constant radius. The definitions and properties of canal surfaces – as well as their relations to the boundaries of sphere–sweeps – will be studied in this appendix.

Usually a smooth curve C is either an infinite or a closed curve (degenerate cases of geometrically discontinuous curves are not considered here). However, in solid modelling we must deal with edges that are *segments* of continuous curves.

A.1 DEFINITION

DEFINITION A.1: **A *canal surface of spheres of constant radius* (hereafter called simply *canal surface*) is defined mathematically [Monge 1849] as the *envelope* of a family of spheres of a constant radius whose centers describe a smooth curve C (called *spine*).**

Such surfaces have a *tubular* shape (figure A.1) and are classified as *Right Circular Constant Generalized Cylinders* by Shafer [Shafer 83], who defines them as the sweep of a circular (normal) cross–section centered on the spine. Both definitions are equivalent as it will be shown below.

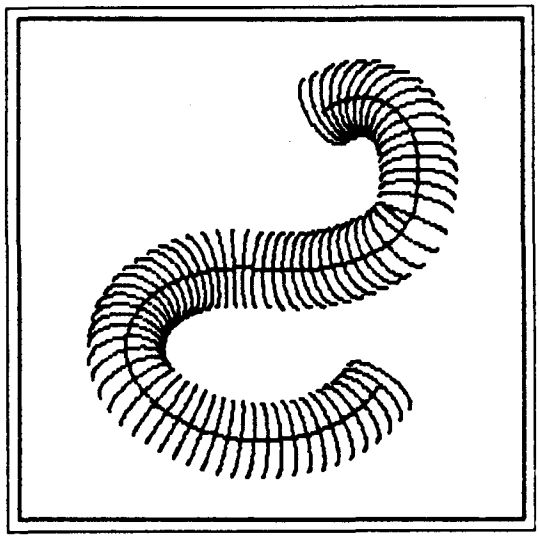

FIGURE A.1 : Canal surface of spheres of constant radius swept along a spine.

A.2 EQUATION

The derivation of the equations that define canal surfaces can be found in standard books on analytic and differential geometry [Salmon 1882]. We shall summarize results useful for our work. For simplicity, we use an arc length parametrization $C(t)$ for the spine.

Point P on a sphere centered at $C(t)$ must verify the equation:

$$\|P - C(t)\| = r$$

Such a sphere can be associated with the parameter t. Its implicit equation depends on t and can be expressed as

$$F(P,t) = 0$$

where

$$F(P,t) = \|P - C(t)\|^2 - r^2$$

We shall use the standard methods of calculus to find the equation of the envelope of a family of parametrized surfaces: we shall consider a second sphere, $F(P,t+dt)$, *infinitely close* to the first one ($dt \to 0$); both spheres will intersect in a circle that defines the contact curve between the two spheres and their envelope. This curve is called the *characteristic*. A

point P lies on the characteristic, and therefore on the envelope, if — for t tending to 0 — it simultaneously verifies the two equations:

$$F(P,t) = 0$$
$$F(P,t+dt) = 0$$

Combining these two equations when t tends to zero, we obtain:

$$\lim_{dt \to 0} \frac{\Big(F(P,t+dt) - F(P,t)\Big)}{dt} = 0$$

which is

$$\frac{\partial F(P,t)}{\partial t} = 0$$

The canal surface is defined as the locus of these characteristic curves defined by the two equations:

$$F(P,t) = 0$$
$$\frac{\partial F(P,t)}{\partial t} = 0$$

By eliminating t from these two equations, we can obtain the implicit equation of a canal surface in the form

$$S(P) = 0$$

This can be achieved for simple cases, where the spine is a simple planar curve (a linear spine produces a cylindrical surface, a circular spine produces a toroidal surface). However, the elimination of t in the general case may be extremely complex (or perhaps impossible) and the resulting surface equation might be expensive to represent and to evaluate.

The second equation implies that $F(P,t)$ passes by an extremum while t is varying. The geometrical interpretation is revealed by analyzing the form of the second equation in the case of the sphere. Partial differentiation with respect to t yields:

$$-2\left(\frac{dC(t)}{dt}\right) \cdot \Big(P - C(t)\Big) = 0$$

Here "$\cdot$" denotes the scalar product and $dC(t)/dt$ denotes the tangent $T(t)$ to the spine at $C(t)$. It follows that the vector $\Big(P - C(t)\Big)$ is orthogonal to this tangent; consequently:

PROPERTY A.1: **The characteristic of every sphere of a canal surface is a circle that lies in the plane orthogonal to the spine at the center of the circle.**

This property demonstrates that a canal surface can be obtained by *sweeping a circular cross-section along the spine.*

A.3 PARAMETRIZATION

We showed that the canal surface can be viewed as a sweep of a circular cross-section along the spine; therefore, the parametrization of the spine defines a parametrization of the surface in the *longitudinal* direction. The parametrization of the cross-section is more subtle, and requires the choice of an origin. We shall assume that such an origin exists at each point of the spine.

We shall use a parametrization that is associated with the Frenet's moving trihedron $\Big(T(t), N(t), B(t)\Big)$ [Struik 50], and will be undefined for portions of the spine where the curvature is equal to zero. These cases lead to a locally cylindrical canal surface which can be dealt with in other ways.

A point P on a canal surface will be parametrized by t, the arc-length position of the center of a characteristic on the spine, and by θ, the angle along the characteristic.

$$P(t,\theta) = C(t) + r\Big(\cos(\theta)N(t) + \sin(\theta)B(t)\Big)$$

A.4 NORMAL TO THE CANAL SURFACE

The normal vector at P to the canal surface S can be defined as the cross-product

$$N(P,S) = \left(\frac{\partial P(t,\theta)}{\partial t}\right) \times \left(\frac{\partial P(t,\theta)}{\partial \theta}\right)$$

if these partial derivatives exist.

$$\left(\frac{\partial P(t,\theta)}{\partial t}\right) = T(t) + r\left(\cos\theta\frac{\partial N(t)}{\partial t} + \sin\theta\frac{\partial B(t)}{\partial t}\right)$$

and since

$$\frac{\partial N(t)}{\partial t} = -\kappa T(t) + \tau B(t)$$

and

$$\frac{\partial B(t)}{\partial t} = -\tau N(t)$$

where τ is the torsion and κ the curvature, we have

$$\left(\frac{\partial P(t,\theta)}{\partial t}\right) = (1 - \kappa r \cos\theta)T(t) - \tau r \sin\theta N(t) + \tau r \cos\theta B(t)$$

we also have

$$\left(\frac{\partial P(t,\theta)}{\partial \theta}\right) = -r \sin\theta N(t) + r \cos\theta B(t)$$

And the normal to S at P is:

$$N(P,S) = r(1 - \kappa r \cos\theta)\Big(\cos\theta N(t) + \sin\theta B(t)\Big)$$

The normal is defined at points where

$$1 \neq \kappa r \cos\theta$$

κ is the curvature and we define g as the radius of curvature ($g = 1/\kappa$). The normal is defined everywhere if $g \neq r\cos\theta$.

Since $\cos\theta$ varies between -1 and 1 along a characteristic, the normal will be defined everywhere along this characteristic if and only if

$$g > r$$

Since the above expression of the normal is differentiable in terms of t and of θ, we have shown that:

PROPERTY A.2: **If the Frenet's moving trihedron is defined for a curve C, the canal surface of radius r will be smooth everywhere if the radius of curvature of C is larger than r.**

Points where

$$g = r\cos\theta$$

are called *focal points.* They are also defined as the intersection of any two infinitely close characteristics.

The two focal points of the same characteristic can be intuitively considered as defining the axis of instantaneous rotation of the circular cross–section. They divide the cross–section in two regions, one moves *forward* along the spine while the other one moves *backward* (regresses). This is shown by studying the sign of the projection of $(\partial P(t,\theta)/\partial t)$ on $T(t)$:

$$\left(\frac{\partial P(t,\theta)}{\partial t}\right)\cdot T(t) = (1 - \kappa r\cos\theta)$$

The focal points form a curve called *edge of regression.* The equation of the edge of regression can be derived from the three equations:

$$F(P,t) = 0$$
$$F'(P,t) = 0$$
$$F''(P,t) = 0$$

where F' and F'' denote the first and second partial derivatives of F with respect to t.

The third equation $F''(P,t) = 0$ applied to the sphere is:

$$-2\frac{\partial^2 C(t)}{\partial t^2}\cdot\Big(P - C(t)\Big) + 2 = 0$$

Here $\partial^2 C(t)/\partial t^2$ is the derivative of the tangent to the spine and can be written as kN, where N is the principal normal to the spine at $C(t)$. This is equivalent to:

$$\Big(P - C(t)\Big)\cdot N = g$$

Here g is the radius of curvature of the spine C. It follows that:

PROPERTY A.3: **If g (the radius of curvature of the curve C) is greater than r then the canal surface of radius r and spine C has no edge of regression.**

From the expression of the normal in the Frenet's moving trihedron we see that:

PROPERTY A.4: **The normal to the canal surface at a point P of the characteristic K passes by the center O of this characteristic and is orthogonal to the tangent to the spine at O.**

Even when it is smooth, a canal surface can intersect itself, and these self–intersection edges are different from the edges of regression. When two parts of the canal–surface are intersecting, it can be a *local* intersection due to a curvature that is larger than $1/r$ or a *global* intersection due to the fact that separate sections of the spine are too close to each other. In both cases, each one of the two parts of the surface is smooth – even though a self–intersection edge exists. When dealing with the boundary of sweeps of spheres, we are interested in the outer part of the surface, and the self–crossing edges become singularities of the boundary of the sweep.

REFERENCES

[Barnhill 74] R. E. Barnhill and R. F. Riesenfeld, Eds., *Computer Aided Geometric Design.* New York: Academic Press, 1974.

[Barsky 84] B. A. Barsky and T. D. DeRose, "Geometric continuity of parametric curves" Report No. UCB/CSD 84/205, Computer Science Dept., Univ. of California, Berkeley, October 1984.

[Barton 80] E. E. Barton and I. Buchanan, "The polygon package", *Computer Aided Design*, vol. 12, no. 1, pp. 3–11, January 1980.

[Braid 80] I. C. Braid, "Superficial blends in geometric modelling", C.A.D. Group Document No. 105, Univ. of Cambridge, February 1980.

[Brown 82] C. M. Brown, "PADL–2: A technical summary", *IEEE Computer Graphics & Applications*, vol. 2, no. 2, pp. 69–84, March 1982.

[Catmull 78] E. E. Catmull and J. Clark, "Recursively generated B–spline surfaces on arbitrary topological meshes", *Computer Aided Design*, vol. 10, no. 6, pp. 350–355, November 1978.

[Chiyokura 83] H. Chiyokura and F. Kimura, "Design of solids with free–form surfaces", *Proc. ACM SigGraph '83*, Detroit, MI, pp. 73–82, July 25–29, 1983.

[Carlson 82] W. E. Carlson, "An algorithm and data structure for 3D object synthesis using surface patch intersection", *Proc. ACM SigGraph '82.* Boston, MA, pp. 255–263, July 26–30, 1982.

[Doo 78a] D. W. H. Doo, "A subdivision algorithm for smoothing down irregular shaped polyhedrons", *Proc. Int'l. Conf. on Interactive Techniques in CAD*, Bologna, Italy, pp. 157–165, September 1978.

[Doo 78b] D. W. H. Doo and M. Sabin, "Behaviour of recursive division surfaces near extraordinary points", *Computer Aided Design*, vol. 10, no. 6, pp. 356–360, November 1978.

[Faux 79] I. D. Faux and M. J. Pratt, *Computational Geometry for Design and Manufacture.* Chichester, U.K.: Ellis Horwood, 1979.

[Fisher 78] W. B. Fisher, A. A. G. Requicha, N. M. Samuel and H. B. Voelcker, "Part and assembly description languages — II", Tech. Memo. No. 20b, Production Automation Project, Univ. of Rochester, June 1978.

[Fridshal 82] R. Fridshal, K. P. Cheng, D. Duncan and W. Zucker, "Numerical control part program verification system", *Proc. Conf. on CAD/CAM Technology in Mechanical Engineering*, Cambridge, MA, pp. 236–254, March 24–26, 1982.

[Harada 82] K. Harada and E. Nakamae, "Application of the Bezier curve to data interpolation", *Computer Aided Design*, vol. 14, no. 1, pp. 55-59, January 1982.

[Hoffmann 85] C. Hoffmann and J. Hopcroft, "Automatic surface generation in computer aided design", TR 85–661, Computer Science Dept., Cornell Univ., Ithaca, NY, January 1985.

[Jared 84] G. E. M. Jared and T. Varady, "Synthesis of volume modelling and sculptured surfaces in BUILD", *Proc. CAD '84*, Brighton, U.K., pp. 481–495, April 3–5, 1984.

[Lee 82] Y. T. Lee and A. A. G. Requicha, "Algorithms for computing the volume and other integral properties of solids: I — Known methods and open issues, pp. 635–641; II — A family of algorithms based on representation conversion and cellular approximation", pp. 642–650, *Comm. ACM*, vol. 25, no. 9, September 1982.

[Liou 76] M. Liou, "Spline fit made easy", *IEEE Trans. on Computers*, vol. C–25, no. 5, May 1976.

[Lozano–Perez 79] T. Lozano–Perez and M. A. Wesley, "An algorithm for planning collision–free paths amongst polyhedral obstacles", *Comm. ACM*, vol. 22, no. 10, pp. 560–570, October 1979.

[Koparkar 84] P. A. Koparkar and S. P. Mudur, "Computational techniques for processing parametric surfaces", *Computer Vision, Graphics, and Image Processing*, vol. 28, no. 3, pp. 303–322, December 1984.

[Matheron 75] G. Matheron, *Random sets and integral geometry*. New York: John Wiley & Sons, 1975.

[McLaughlin 83] H. W. McLaughlin, "Shape–preserving planar interpolation: An algorithm", *IEEE Computer Graphics & Applications*, vol. 3, no. 3, pp. 58–67, May 1983.

[Mendelson 75] B. Mendelson, *Introduction to topology*. Boston, MA: Allyn and Bacon, 3rd ed., 1975.

[Monge 1849] G. Monge, *Applications de l'analyse à la géométrie*. Paris: Bachelier, 5th ed., 1849.

[Morgan 81] A. P. Morgan, "An algorithm for solving the line–tube classification problem", Report GMR–3858, Mathematics Dept., General Motors Research Labs., Warren, MI, October 1981.

[Nadler 78] S. B. Nadler Jr., *Hyperspaces of sets*. New York: Marcel Dekker, 1978.

[Pressman 77] R. S. Pressman and J. E. Williams, *Numerical Control and Computer-Aided Manufacturing*. New York: John Wiley & Sons, 1977.

[Requicha 77a] A. A. G. Requicha, "Mathematical models of rigid solid objects", Tech. Memo. No. 28, Production Automation Project, Univ. of Rochester, November 1977.

[Requicha 77b] A. A. G. Requicha and H. B. Voelcker, "Constructive solid geometry", Tech. Memo. No. 25, Production Automation Project, Univ. of Rochester, November 1977.

[Requicha 78] A. A. G. Requicha and R. B. Tilove, "Mathematical foundation of constructive solid geometry: General topology of closed regular sets", Tech. Memo. No. 27a, Production Automation Project, Univ. of Rochester, June 1978.

[Requicha 80] A. A. G. Requicha, "Representations for rigid solids: Theory, methods, and systems", *ACM Computing Surveys*, vol. 12, no. 4, pp. 437–464, December 1980.

[Requicha 82] A. A. G. Requicha and H. B. Voelcker, "Solid modelling: A historical summary and contemporary assessment", *IEEE Computer Graphics & Applications*, vol. 2, no. 2, pp. 9–24, March 1982.

[Requicha 83] A. A. G. Requicha and H. B. Voelcker, "Solid modelling: Current status and research directions", *IEEE Computer Graphics & Applications*, vol. 3, no. 7, pp. 25–37, October 1983.

[Requicha 85] A. A. G. Requicha and H. B. Voelcker, "Boolean operations in solid modelling: Boundary evaluation and merging algorithms", *Proc. IEEE*, vol. 3, no. 1, pp. 30–44, January 1985.

[Ricci 73] A. Ricci, "A constructive geometry for computer graphics", *Computer Journal*, vol. 16, no. 2, pp. 157–160, May 1973.

[Rockwood 83] A. P. Rockwood, "Introducing sculptured surfaces into a geometric modeler", in M. S. Pickett and J. W. Boyse, Eds., *Solid Modeling by Computers.* New York: Plenum Press, pp. 237–258, 1984.

[Rossignac 85] J. R. Rossignac and A. A. G. Requicha, "Depth–buffering display techniques for Constructive Solid Geometry", Tech. Memo 64, (in draft), Production Automation Project, University of Rochester, 1985.

[Roth 82] S. D. Roth, "Ray casting for modeling solids", *Computer Graphics & Image Processing*, vol. 18, no. 2, pp. 109–144, February 1982.

[Sabin 77] M. Sabin, "The use of piecewise forms for the numerical representation of shape", Report 60, Computer and Automation Institute, Hungarian Academy of Science, 1977.

[Sabin 80] M. Sabin, "Contouring — A review of methods for scattered data", in K. W. Brodlie, Ed., *Mathematical Methods in Computer Graphics and Design.* New York: Academic Press, 1980.

[Salmon 1882] G. Salmon, *A Treatise on the Analytic Geometry of Three Dimensions.* Dublin: Hodges, Figgis and Co., 4th ed., 1882.

[Samuel 76] N. M. Samuel, A. A. G. Requicha and S. A. Elkind, "Methodology and results of an industrial part survey", Tech. Memo. No. 21, Production Automation Project, Univ. of Rochester, July 1976.

[Sarraga 83] R. F. Sarraga and W. C. Waters, "Free–form surfaces in GMSOLID: Goals and issues", in M. S. Pickett and J. W. Boyse, Eds., *Solid Modeling by Computers.* New York: Plenum Press, pp. 187–209, 1984.

[Serra 82] J. Serra, *Image analysis and mathematical morphology.* New York: Academic Press, 1982.

[Shafer 83] S. Shafer and T. Kanade, "The theory of straight homogeneous generalized cylinders", Report CMU–CS–83–105, Computer Science Dept., Carnegie–Mellon Univ., January 1983.

[Shaffner 81] S. C. Shaffner, "Calculation of B–spline surfaces using digital filters", *ACM Computer Graphics*, vol. 15, no. 4, pp. 437–457, December 1981.

[Shirma 83] Y. Shirma, N. Okino, and Y. Kakazu, "Research on 3–D geometric modelling by sweep primitives", *Proc. CAD '82*, Brighton, U.K., pp. 671–680, March 30–April, 1982.

[Struik 50] D. J. Struik, *Lectures on classical differential geometry*. Reading, MA: Addison Wesley Press, 1950.

[Tiller 83] W. Tiller, "Rational B–splines for curve and surface representation", *IEEE Computer Graphics & Applications*, vol. 3, no. 6, pp. 61–69, September 1983.

[Tiller 84] W. Tiller and E. G. Hanson, "Offsets of two–dimensional profiles", *IEEE Computer Graphics & Applications*, vol. 4, no. 9, pp. 36–46, September 1984.

[Tilove 80] R. B. Tilove, "Set membership classification: A unified approach to geometric intersection problems", *IEEE Trans. on Computers*, vol. C–29, no. 10, pp. 874–883, October 1980.

[Timmer 77] H. G. Timmer, "A solution to the surface intersection problem", Report MDC J7789, McDonnell Douglas Corporation, November 1977.

[Van Wijk 84] J. J. Van Wijk, "Ray tracing objects defined by sweeping a sphere", *Proc. Eurographics '84*, Copenhagen, September 12–14, 1984.

[Varady 83] T. Varady, "Surface–surface intersections for double–quadric parametric patches in a solid modeller", Computer and Automation Institute, Hungarian Academy of Science, 1983.

[Veenman 82] P. Veenman, "The design of sculptured surfaces using recursive subdivision techniques", *Proc. Conf. CAD/CAM Technology in Mechanical Engineering*, Cambridge, MA, pp. 54–63, March 24–26, 1982.

[Willmore 58] T. J. Willmore, *Differential geometry*. Oxford: Oxford University Press, 1958.