Trimmed-surface algorithms for the evaluation and interrogation of solid boundary representations

by Rida T. Farouki

Although trimmed surfaces play a fundamental role in the derivation and processing of solid boundary representations, they have received little attention to date. We propose a trimmed-surface formulation appropriate to the Boolean combination of primitives bounded by a family of elementary surface patches (e.g., planes, quadrics, ruled surfaces, surfaces of revolution) with dual parametric rational polynomial and implicit algebraic equations. Partial intersections between pairs of primitive surface patches are formulated precisely as algebraic curves in the parameter space of each patch. These curves are dissected into monotonic branches by the identification of a characteristic point set. The consolidation of all partial intersections yields a system of piecewise-algebraic loops which define a trimming boundary enclosing a parametric domain for the trimmed patch. With few exceptions, the trimmed-surface formulation is based on precisely defined mathematical procedures, in order to achieve maximum robustness. Some basic interrogation algorithms for solids bounded by trimmed-surface elements are also presented, including procedures for ray-tracing, point/solid classification, sectioning, and computation of surface area, volume, center of gravity, moments of inertia, and other mass properties.

1. Introduction

Considerable effort has recently been directed toward the development of *solid modeling* technology [1, 2] as a framework for formulating and solving complex geometric problems. The potential applications of these methods range from mechanical parts and VLSI circuits to geological formations, a diversity reflected in the current variety of representational and algorithmic approaches. These endeavors have been fueled by advances in computer hardware and software, new mathematical tools, graphical display techniques, and the resurrection of classical algebraic geometry. However, it is now widely conceded that fundamental algorithmic hurdles have impeded the development of systems with adequate efficiency, robustness, and versatility to fully realize their practical objectives. This is perhaps more true of systems restricted to the relatively modest surface forms (cf. Table 1, shown later) than those addressing complex or amorphous geometries, since the

precision and reliability demanded in the former case preclude the approximate methods which are generally suitable for the latter.

The premium placed on robustness has a clear practical motivation: The executive role proposed for solid modelers in synthesizing and integrating engineering design, analysis, manufacturing, assembly, and inspection applications imposes on such systems an onerous responsibility for individual or compounded errors, with costly consequences. Central to the development of a robust modeler encompassing the family of simpler surfaces commonly found in manufactured parts is a rigorous *trimmed-surface* formulation. A trimmed surface is a finite segment of an unbounded analytic surface, enclosed within complex (possibly nested) border curves lying on that surface. These borders are frequently curves of intersection with other curved surfaces and, in general, they possess no elementary parametric representations.

Few three-dimensional objects are representable by single-surface equations, and thus the need for trimmed surfaces arises immediately in the boundary description of solid models. Given the fundamental importance of trimmed-surface algorithms, it is remarkable that they are largely neglected in the geometric modeling literature—see, however, [3, 4]. Trimmed surfaces are intimately coupled to the thorny problem of surface-intersection computation, and published boundary-evaluation procedures which implicitly process them either bypass the issue of their formal representation or restrict their attention to simple domains (e.g., two-dimensional or polyhedral objects) for which closed-form solutions are available, e.g., [5]. While boundary-evaluation procedures have actually been implemented in some existing modelers, their reliability is apparently less than satisfactory [6].

In this paper we attempt to formulate a rigorous trimmed-surface definition. We avoid, as far as possible, heuristic methods and base the definition on well-defined mathematical procedures such as the arithmetic operations and polynomial root-solving. Our emphasis is on the definition and processing of individual trimmed-surface elements. Little consideration is given at present to issues such as computational efficiency, data structures for organizing the adjacency relationships between trimmed-surface elements, and the boundary evaluation procedure *per se*. These subjects are discussed elsewhere in the literature.

2. Trimmed surfaces and boundary representations

An abstract trimmed-surface definition, independent of the diverse *representations* of solids and their bounding surfaces occurring in solid modeling systems, forms a useful point of departure [7]. Adopting the usual notation of set theory, let A and B denote regular, compact volumes in $\mathbf{R}^3$ and let

$$A \cup B,\ A \cap B,\ A - B,\ B - A \tag{1}$$

be the volumes generated by the regularized [8] Boolean operations of union, intersection, and difference on them. If the volumes A and B under consideration represent homogeneous solids in a solid modeling system, they are unambiguously defined by their bounding surfaces $S(A)$ and $S(B)$. Our principal concern is thus to determine the boundaries

$$S(A \cup B),\ S(A \cap B),\ S(A - B),\ S(B - A) \tag{2}$$

of the Boolean combinations (1), given the input boundaries $S(A)$ and $S(B)$.

Any compact volume X may be regarded as partitioning $\mathbf{R}^3$ into two regions: a bounded open set $I(X)$, the *interior* of X, and an unbounded open set $E(X)$, the *exterior* or *complement* of X. The interior and exterior are separated by a surface $S(X)$, called the *boundary* of X. The *exterior trimmed surface* of A with respect to B, denoted $S(A > B)$, is defined as the open subset of $S(A)$ exterior to B:

$$S(A > B) = S(A) \cap E(B). \tag{3}$$

Thus $S(A > B)$ is empty if $A \subseteq B$, is equal to $S(A)$ if $B \subset A$ or A and B are disjoint, and is a subset of $S(A)$ otherwise. The *interior trimmed surface* of A with respect to B, denoted $S(A < B)$, is defined as the open subset of $S(A)$ interior to B:

$$S(A < B) = S(A) \cap I(B). \tag{4}$$

Thus $S(A < B)$ is empty if $B \subseteq A$ or A and B are disjoint, is equal to $S(A)$ if $A \subset B$, and is a subset of $S(A)$ otherwise.

The exterior and interior trimmed surfaces of B with respect to A, denoted $S(B > A)$ and $S(B < A)$ respectively, are defined in a similar manner:

$$S(B > A) = S(B) \cap E(A),\ S(B < A) = S(B) \cap I(A). \tag{5}$$

The *curve of intersection* of A and B, denoted $C(A, B)$, is defined as the set of points common to $S(A)$ and $S(B)$:

$$C(A, B) = S(A) \cap S(B). \tag{6}$$

Ordinarily, $C(A, B)$ consists of one or more closed, one-dimensional space curves. However, in exceptional cases where $S(A)$ and $S(B)$ touch at a point or over a common area, the curve $C(A, B)$ degenerates and may suffer isolated points or two-dimensional regions. In such cases a *regularization* procedure [9] may be invoked to render the intersection uniformly one-dimensional, and the trimmed-surface definitions must then be modified accordingly for the boundary formulae given below to remain valid.

Assuming regular intersections, the boundaries of the Boolean combinations (1) are given in terms of the interior and exterior trimmed surfaces and the intersection curve:

$$\begin{aligned}
S(A \cup B) &= S(A > B) \cup S(B > A) \cup C(A, B),\\
S(A \cap B) &= S(A < B) \cup S(B < A) \cup C(A, B),\\
S(A - B) &= S(A > B) \cup S(B < A) \cup C(A, B),\\
S(B - A) &= S(A < B) \cup S(B > A) \cup C(A, B).
\end{aligned} \tag{7}$$

The four possible combinations of interior and exterior trimmed surfaces from A and B, respectively, together with their mutual curve of intersection, thus comprise the boundaries of the Boolean combinations (1). The problems of computing and interrogating boundary representations for Boolean solid combinations therefore reduce to those of developing algorithms for deriving and processing trimmed-surface formulations.

In subsequent discussions it is convenient to define trimmed surfaces as closed two-dimensional sets, incorporating the curve-of-intersection boundary as part of the trimmed-surface definition. Thus, for example,

$$S(A \geq B) = S(A > B) \cup C(A, B) \tag{8}$$

denotes the exterior trimmed surface of A with respect to B *including* the regularized intersection boundary. Although intersection curves are duplicated when such trimmed surfaces are pieced together, this presents no special difficulties.

3. Polynomial root determination

The trimmed-surface methods presented below require procedures for determining the real roots of univariate polynomials and pairs of simultaneous bivariate polynomials on specified domains, usually the unit line [0, 1] and the unit square [0, 1] × [0, 1], respectively. The *multiplicities* of these roots must also be recognized. Polynomial roots are usually irrational numbers, and must therefore be represented with finite precision. Furthermore, since closed-form solutions are not available for degrees higher than 4, the numerical algorithms invoked to solve them introduce additional errors. A truly robust implementation of the trimmed-surface algorithms presented here will thus be realized only in the context of a full theory of imprecise geometric representation and computation. This matter is beyond our present scope, and we defer it to a subsequent study [10].

In principle, the numerical root-solving procedures discussed below are amenable to polynomials of arbitrary degree. For practical floating-point implementations, however, severe limitations are imposed by poor accuracy and efficiency in the evaluation of high-degree polynomials. The polynomial degrees arising in the intersection and trimming of a family of simpler surfaces of practical value (cf. Table 1, shown later) may nevertheless prove tractable.

The philosophy underlying our trimmed-surface formulation is to provide, as far as possible, a sound mathematical foundation for the definition and processing of trimmed-surface elements, without regard to computational efficiency. While ultimately we have recourse to numerical procedures in determining polynomial roots, we wish to preclude the *ab initio* heuristic flavor of many current surface-intersection computations, e.g., [11, 12], where algorithm deficiency may equal or exceed imprecise computation as a source of failure. The advantages of our delaying action in the introduction of numerical procedures are clear: Any advances in reliability or efficiency in the root algorithms or in floating-point precision in general are imported directly into the system.

3.1 Univariate polynomials

We require robust algorithms for determining the real roots t_i of a degree-n polynomial

$$P_n(t) = \sum_{k=0}^{n} a_k t^k = 0 \tag{9}$$

on a bounded interval $[t_a, t_b]$, to specified precision ε (i.e., the computed roots should lie in the intervals $t_i \pm \varepsilon$ about the true roots [13]). The interval $[t_a, t_b]$ may be conveniently transformed to the unit line [0, 1]. The procedure should find *all* real roots in the specified interval. Furthermore, there should be no loss of resolution at multiple or nearly multiple roots, where the problem becomes ill-conditioned [14].

The polynomial coefficients $\{a_k\}$ in (9) are rarely available as precise initial data, but result from floating-point computations themselves (e.g., as in the elimination procedures described below). Polynomial roots may be ill-conditioned with respect to perturbation of the coefficients even with widely separated roots, and therefore thorough error analyses [14] are required for truly robust implementation of our trimmed-surface formulations.

The above considerations preclude the simpler iteration methods based on guessing initial root approximations [15], unless there is additionally available a root *isolation* procedure which furnishes a set of nonoverlapping subintervals which contain each of the distinct real roots on the entire interval [16]. We outline two possible techniques.

3.1.1 Bernstein-Bezier subdivision

The interval $[t_a, t_b]$ is transformed to [0, 1] and $P_n(t)$ is expressed in the Bernstein basis:

$$P_n(t) = \sum_{k=0}^{n} \frac{n!}{(n-k)!k!} c_k (1-t)^{n-k} t^k \tag{10}$$

on that interval. The coefficients $\{c_k\}$ are obtained from the $\{a_k\}$ by collating and equating terms of equal power. In the form (10), the polynomial has the convex hull and variation-diminishing properties of a Bezier curve. These furnish an iterative subdivision procedure which should isolate and converge on each real root [17]. However, convergence problems arise at multiple roots, and the technique requires refinement to process such cases reliably.

3.1.2 Sturm sequences

A *Sturm sequence* for the polynomial $P_n(t)$ on the interval $[t_a, t_b]$,

$$f_0(t), f_1(t), \cdots, f_m(t), \tag{11}$$

may be generated in the following manner. Set $f_0(t) \equiv P_n(t)$ and $f_1(t) \equiv dP_n/dt$, and denote by $R[p(t), q(t)]$ the polynomial remainder on dividing $p(t)$ by $q(t)$. Then

$$f_i(t) = -R[f_{i-2}(t), f_{i-1}(t)], \qquad i = 2, \cdots, m, \tag{12}$$

where the process is continued until an $f_m(t)$ is generated ($m \leq n$) which has constant sign on $[t_a, t_b]$. The number N of real, distinct zeros of $P_n(t)$ on $[t_a, t_b]$ is then given by

$$N = V(t_a) - V(t_b), \tag{13}$$

where $V(t)$ denotes the number of sign changes of the Sturm sequence (11) evaluated at t [16, 18]. This theorem provides a means for isolating all real zeros by binary subdivision of the interval $[t_a, t_b]$ until the number of roots indicated for each subinterval is 0 or 1.

3.2 Simultaneous bivariate polynomials

Simultaneous bivariate polynomial equations of the form

$$\sum_{i=0}^{m} \sum_{j=0}^{n} a_{ij} u^i v^j = \sum_{k=0}^{p} \sum_{l=0}^{q} b_{kl} u^k v^l = 0 \tag{14}$$

arise frequently in the trimmed-surface procedures described below. We wish to determine the real roots $(u, v) \in [0, 1] \times [0, 1]$ of (14), which correspond to the intersections of two algebraic curves within the unit square. The bivariate root-solving procedure should meet the same criteria as in the univariate case. All relevant roots must be identified, and multiple-root cases should be accommodated without difficulty. To realize these requirements, we employ *elimination* techniques to transform the problem (14) into a univariate polynomial problem, and take advantage of the robust algorithms of Section 3.1.

The *resultant* of Equations (14) with respect to either variable is a univariate polynomial of degree $mq + np$ in the other variable whose roots are the discrete values for that variable at which the two curves (14) intersect. The resultant may be conveniently computed by expanding the Sylvester or Bezout determinant for the system (14) [19]. To each *simple* root of the univariate resultant there corresponds a *unique* value for the other variable; if the resultant has a root of multiplicity h, there are h (possibly coincident) corresponding values for the other variable. Usually only the real roots on the unit interval [0, 1] for each variable are of interest.

Elimination methods have been successfully implemented in symbolic computation (computer algebra) systems for some time, e.g., [20–24], but their application to computer-aided design problems is relatively recent [25–27]. In the former case, the coefficients of the resultant polynomial may be computed precisely as symbolic expressions or rational numbers, while in the latter they suffer errors due to the imprecise floating-point arithmetic incurred in the determinant expansion. An ill-conditioned resultant would then induce gross errors in the estimated roots. Although the numerical stability of elimination methods in this context requires further investigation, we have processed simultaneous bicubic equations to high accuracy in practical implementations [28].

The geometric problems encountered below which can be set in the form (14) include the following: constructing the bounding box around a surface patch; finding the (u, v) surface coordinates of a Cartesian point (x, y, z) lying on the parametric surface $\mathbf{r} = \mathbf{r}(u, v)$; determining the intersections of a three-dimensional ray with a surface patch; identifying a characteristic point set for an algebraic curve; and computing the intersections of two monotonic algebraic curve branches.

4. Surface formulations

We review below elementary features of a dual parametric and implicit surface patch representation, and propose a trimmed-surface formulation based on it. The discussion of algorithms for producing and processing such trimmed surfaces in specific Boolean solid combinations is deferred to a subsequent section.

4.1 Parametric equations

We define a surface patch as a mapping of the unit square $(u, v) \in [0, 1] \times [0, 1]$ into $\mathbf{R}^3$ by rational polynomial functions expressed in the Cartesian product form:

$$\mathbf{r}(u, v) = \frac{\sum_{j=0}^{m} \sum_{k=0}^{n} w_{jk} \mathbf{r}_{jk} u^j v^k}{\sum_{j=0}^{m} \sum_{k=0}^{n} w_{jk} u^j v^k}. \tag{15}$$

Such patches are inherently four-sided, but may be forced into three-sided configurations by introducing a parametric singularity in which one side degenerates to zero length. Patches arising in a solid modeling context are assumed by construction to be well formed, i.e., free of discontinuities such as cusps, ridges, and self-intersections.

The number of coefficients defining the surface (15) is $4(m+1)(n+1)$, the number of degrees of freedom being one less, since only the relative magnitudes of the projective coordinates w_{jk} are significant. Familiar elementary surfaces—planes, quadrics, surfaces of extrusion and revolution—and other simple *swept* surfaces possess parameterizations of the above form [29], although higher-order algebraic surfaces in general do not.

4.2 Implicit equations

All rational polynomial surfaces of the form (15) may be represented by implicit algebraic equations $f(x, y, z) = 0$ of degree $p \leq 2mn$, obtained by eliminating the two parametric variables (u, v) from the three equations $x = x(u, v)$, $y = y(u, v)$, $z = z(u, v)$, a procedure known as implicitization [26]. In general, a degree-p algebraic surface has the implicit equation

Table 1 Elementary surfaces.

Surface type	*Degree*
Plane	1
Quadric	2
Conical or cylindrical surface, degree-n profile curve	n
Surface of revolution, degree-n profile curve	$2n$

$$f(x, y, z) = \sum_{i=0}^{p} \sum_{j=0}^{p-i} \sum_{k=0}^{p-i-j} c_{ijk} x^i y^j z^k = 0, \tag{16}$$

defining an *unbounded* surface (which closes on itself or extends to infinity). The parametric form (15) represents only a finite segment on such a surface. All points satisfying (16) which lie outside the unit parameter square for (15) are called the *algebraic extension* of the patch. The number of distinct terms in (16) is $(p + 1)(p + 2)(p + 3)/6$, the number of degrees of freedom being one less since the equation may be divided through by any nonzero coefficient without material change.

The formal implicitization procedure based on elimination of the parametric variables from Equations (15) is, in general, too complex for practical software implementation [30]. For the family of low-degree surfaces shown in **Table 1**, implicit equations may be generated *ab initio* and must be suitably transformed whenever their parent solid is subjected to rigid-body motions. Although the trimmed-surface algorithms developed here are in principle applicable to arbitrary patches having the dual parametric and implicit equations (15) and (16), restricting consideration to these simpler surfaces constrains the polynomial degrees arising during trimmed-surface processing within realistic limits.

4.3 Solid primitives

A solid *primitive* P is defined to be the three-dimensional volume enclosed by a collection of surface patches of the form (15) with matched borders. Each border of every patch matches precisely a border of one other patch, or is degenerate (has zero length). The family of primitives possessing exact definitions in the above form is quite diverse, incorporating polyhedra and solids of revolution and extrusion [29] and their offsets [31] as well as more general solids. The basic interrogation function for such solid primitives, classifying a candidate point $\mathbf{p}$ as lying inside, on the boundary of, or outside the volume of P, is accomplished by determining the intersections of an arbitrary line through $\mathbf{p}$ with the patches of P. These intersections divide the line into inside/outside intervals, and the location of $\mathbf{p}$ relative to these intervals provides the required classification.

The intersections of a line and a finite patch may be determined in one of two ways: (a) from the parametric equations (15), using elimination methods to obtain a univariate polynomial of degree $2mn$ [25]—this gives the surface coordinates (u_i, v_i) of each intersection; or (b) from the implicit equation (16) [32], which yields a degree-p polynomial from which the Cartesian coordinates (x_i, y_i, z_i) of each intersection are obtained—an *inversion* procedure (cf. Section 4.5) must then be employed to ensure that these lie within the parameter domain of the patch. Testing the line against bounding boxes for the surface patches (see below) minimizes unnecessary computation.

4.4 Bounding boxes

Testing all patch pairs of two solids for intersection is inefficient, and Boolean combinations can be accelerated if bounding boxes are available for each surface patch, allowing most disjoint pairs to be bypassed immediately [33]. Suppose $\mathbf{r}(u, v)$ is of degree (m, n) in (u, v). Considering each coordinate component (x, y, z) individually, the spatial extent of a patch is determined by comparing *corner points*, *border extrema*, and *surface extrema*.

The corner points are simply

$$\mathbf{r}(0, 0), \mathbf{r}(1, 0), \mathbf{r}(0, 1), \mathbf{r}(1, 1), \tag{17}$$

and the border extrema occur at the roots on [0, 1] of

$$\mathbf{r}_u(u, 0) = 0, \mathbf{r}_u(u, 1) = 0, \mathbf{r}_v(0, v) = 0, \mathbf{r}_v(1, v) = 0, \tag{18}$$

which are univariate polynomials of degree $m - 1$ or $n - 1$. Finally, the surface extrema occur at the roots on $[0, 1] \times [0, 1]$ of the simultaneous equations

$$\mathbf{r}_u(u, v) = \mathbf{r}_v(u, v) = 0. \tag{19}$$

Eliminating either variable between these equations yields a univariate polynomial of degree $2mn - m - n + 1$. A bounding box for an entire primitive is defined by the extremum coordinates of the bounding boxes for its constituent patches. These boxes allow the Boolean combination procedure to be bypassed in most instances where the primitives are disjoint.

4.5 Inversion and imaging

In computing surface intersections and formulating trimmed-surface representations for patches with the dual implicit and parametric representation $\{\mathbf{r} = \mathbf{r}(u, v), f(x, y, z) = 0\}$, an *inversion* procedure is required: Given a Cartesian point $\mathbf{r}_0 = (x_0, y_0, z_0)$ known to lie on the surface, $f(x_0, y_0, z_0) = 0$, determine its parametric coordinates (u_0, v_0). This can be accomplished by elimination of u or v from any pair of the three equations $x_0 = x(u, v)$, $y_0 = y(u, v)$, $z_0 = z(u, v)$ which are not *both* independent of u or v—consistency is guaranteed by the condition $f(x_0, y_0, z_0) = 0$. For $\mathbf{r}(u, v)$ of degree (m, n) in (u, v), this incurs a univariate resultant polynomial of degree $2mn$. Once this has been solved, a corresponding value for the other variable is readily determined.

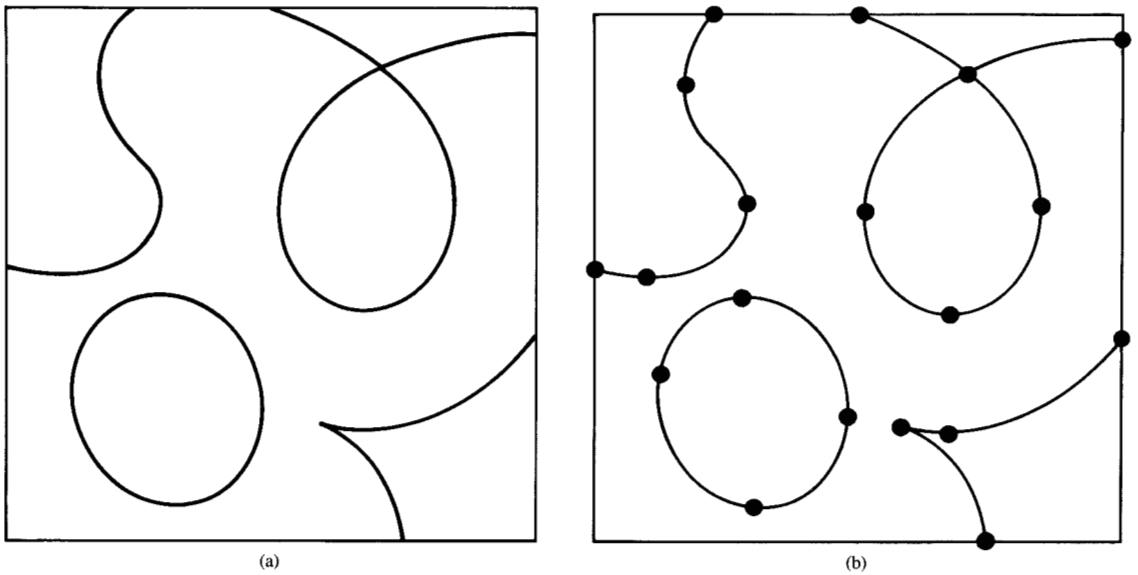


Figure 1

Schematic illustrations of (a) the complex structure of a high-order algebraic curve on the unit square; and (b) the curve dissected into monotonic branches by identifying a set of characteristic points: border points, turning points, and singular points.

An important application of the inversion procedure is in *imaging* points on the intersection of two surfaces between the parametric spaces of those surfaces. Suppose the point (u_0, v_0) of $\mathbf{r}(u, v)$ is known to lie on the intersection of the surfaces $\mathbf{r} = \mathbf{r}(s, t)$ and $\mathbf{r} = \mathbf{r}(u, v)$. We wish to determine the parametric coordinates (s_0, t_0) on $\mathbf{r}(s, t)$ corresponding to (u_0, v_0). This is achieved by projecting to three dimensions, $\mathbf{r}_0 = \mathbf{r}(u_0, v_0)$, and inverting the Cartesian point $\mathbf{r}_0$ with respect to $\mathbf{r} = \mathbf{r}(s, t)$.

4.6 Parametric surface sections

The intersection of an unbounded algebraic surface (16) with a parametric surface patch (15) is given precisely by an algebraic curve of the form

$$F(u, v) = \sum_{r=0}^{pm} \sum_{s=0}^{pn} a_{rs} u^r v^s = 0, \tag{20}$$

obtained by substituting Equation (15) directly into (16) [28]. The curve (20) on $(u, v) \in [0, 1] \times [0, 1]$ defines the intersection of the patch (15) with the *entire* unbounded algebraic surface (16). This is termed a *complete* intersection. If, however, we are interested in the intersection of two finite parametric patches, Equation (16) (representing the implicitization of one, then only a segment of the algebraic surface) participates in the intersection. The portions of $F(u, v) = 0$ contributing to the intersection in this case lie within an ambiguous subset of the unit parameter square and must be identified carefully. With trimmed-surface patches, the difficulties are even further compounded. We return to the problem of such *partial* intersections in Section 5.

4.7 Characteristic points

In general, high-order algebraic curves of the form (20) possess complex topological structures [34–36]. The properties of such curves have been studied in depth by the methods of classical algebraic geometry, e.g., [37–39]. For our present purposes, an essential requirement of a surface intersection algorithm is to recognize each individual loop or segment of the intersection and to render the complex structure of the curve tractable. The identification of a set of *characteristic points* [28] for the curve (20) accomplishes these goals by guaranteeing at least one point on each portion or *feature* of the curve, and dissecting the curve into a set of smooth, monotonic branches (**Figure 1**). For the complete surface section (20), the characteristic points fall into three categories:

The *border points* occur at the roots on [0, 1] of

$$F(u, 0) = 0,\ F(u, 1) = 0,\ F(0, v) = 0,\ F(1, v) = 0, \tag{21}$$

where the curve (20) enters or leaves the unit square. Equations (21) are univariate polynomials of degree m or n.

The *turning points* occur at the roots of

$$F(u, v) = F_v(u, v) = 0, \; F(u, v) = F_u(u, v) = 0, \tag{22}$$

where the curve tangent is parallel to the axes $u = 0$ and $v = 0$, respectively. Solving Equations (22) by elimination incurs univariate polynomials of degree $2mn - m$ and $2mn - n$, respectively. Finally, the *singular points* occur at the roots of

$$F(u, v) = F_u(u, v) = F_v(u, v) = 0, \tag{23}$$

where the curve tangent $\pm(F_v, -F_u)$ is not uniquely defined. A singular point is said to have *multiplicity h* if all partial derivatives of F to order $h - 1$ are zero and at least one partial derivative of order h is nonzero. Singular points are identified without further computation by comparing the roots of the two equations (22).

In processing *partial* intersections between pairs of finite parametric patches or trimmed-surface patches, a further characteristic point category must be introduced, the *termination points*, to accommodate intersection tracks which end abruptly within the unit parameter square (cf. Section 5).

4.8 Link multiplicities

Associated with each characteristic point i is a *link multiplicity* m_i, indicating the number of monotonic branches entering or leaving that point. The link multiplicity depends on the type of the characteristic point. For border points and termination points it is 1, and for turning points it is 2. For a singular point of multiplicity h the link multiplicity is usually $2t$, where t ($\leq h$) is the number of real, distinct tangent directions (λ, μ) occurring as roots of

$$\sum_{k=0}^{h} \frac{h!}{(h-k)!k!} \lambda^k \mu^{h-k} \frac{\partial^h F}{\partial u^k \partial v^{h-k}} = 0. \tag{24}$$

For a double point ($h = 2$), for example, (24) becomes $\lambda^2 F_{uu} + 2\lambda\mu F_{uv} + \mu^2 F_{vv} = 0$, and there are three possibilities: (i) roots real and distinct, $t = 2$ and $m_i = 4$, a self-intersection; (ii) roots real and coincident, $t = 1$ and $m_i = 2$, a cusp; and (iii) complex conjugate roots, $t = 0$ and $m_i = 0$, an isolated point of the curve.

In exceptional situations where one of the tangents at a singular point is also a *singular tangent* of the curve, the link multiplicity rule $m_i = 2t$ may fail, however [28]. It is also possible for a characteristic point to fall into more than one category; e.g., a turning point or singular point may lie on the unit square boundary. A more detailed investigation is then required to ascertain the appropriate link multiplicity.

4.9 Monotonic branches

A monotonic branch C of an algebraic curve $F(u, v) = 0$ is a smooth, directed segment of the curve along which a unique tangent direction $\pm(F_v, -F_u) \neq (0, 0)$ is defined, which varies by not more than 90°. Such a branch has definite starting and ending points (u_s, v_s) and (u_e, v_e). The tangent direction may be parallel to the coordinate axes $u = 0$ or $v = 0$ at these endpoints, but not at intermediate points. Note that the definition of a monotonic branch refers to a definite coordinate system and is therefore not independent of orientation.

The specification of an implicit equation and consistent endpoints,

$$\{F(u, v) = 0\text{: } (u_s, v_s) \rightarrow (u_e, v_e)\}, \tag{25}$$

uniquely identifies a monotonic branch C, except under the exceptional circumstance where several monotonic branches connect two singular points. If the singular points are *ordinary* (i.e., all their tangents are distinct), start and end tangent directions may be appended to the specification to resolve the ambiguity.

The monotonic branch is entirely enclosed by the bounding box

$$[u_{min}, u_{max}] \times [v_{min}, v_{max}], \tag{26}$$

where $u_{min} = \min(u_s, u_e)$, $u_{max} = \max(u_s, u_e)$ and $v_{min} = \min(v_s, v_e)$, $v_{max} = \max(v_s, v_e)$. The *algebraic extension* of a monotonic branch C given by (25) is defined to be the set of all points satisfying $F(u, v) = 0$ which do not belong to C. Note that elements of the algebraic extension of C may lie inside its bounding box (26).

4.10 Algebraic curve tracing

The characteristic points (21), (22), and (23) are derived directly from the algebraic curve definition (20) and are known to dissect the curve into a set of monotonic branches of the form (25). However, the *identity* of these branches remains to be established; i.e., we must determine which pairs of characteristic points are indeed connected by a monotonic branch. To accomplish this, we require a *curve-tracing* procedure, i.e., a means of moving along a monotonic branch, starting at a given characteristic point, to see which other characteristic point it leads to.

Ideally, we would like to derive a parameterization of the form

$$\{u = u(t), \; v = v(t)\}, \qquad 0 \leq t \leq 1, \tag{27}$$

for the monotonic branches (25) in terms of elementary rational functions $u(t)$ and $v(t)$ of a parameter t—this would automatically solve the curve-tracing problem. A *sufficient* condition for such a parameterization to exist (i.e., for the curve to be *rational*) is that its *genus* be zero, which implies an exceptional configuration of singular points [39]. Since this condition is not realized for algebraic curves in general, we must resort to numerical procedures for curve tracing (**Figure 2**).

Between two adjacent points (u, v) and $(u + du, v + dv)$ on the curve $F(u, v) = 0$, the change in F is given by $dF = F_u du + F_v dv = 0$. Hence we obtain the expressions

$$\frac{dv}{du} = -F_u/F_v, \qquad \frac{du}{dv} = -F_v/F_u, \tag{28}$$

along $F(u, v) = 0$. Higher-order derivatives along $F(u, v) = 0$ may be generated by repeated application of the total derivative operators

$$\frac{d}{du} = \frac{\partial}{\partial u} - (F_u/F_v)\frac{\partial}{\partial v}, \qquad \frac{d}{dv} = -(F_v/F_u)\frac{\partial}{\partial u} + \frac{\partial}{\partial v} \tag{29}$$

to (28); the resulting expressions are rather cumbersome [28].

The total derivatives define a Taylor series giving v in terms of u, or vice versa, in the neighborhood of any nonsingular point of the curve. The power series can also be derived from an algebraic recursion formula [40]. Such local power series may be employed to trace a monotonic branch by small steps in u or v. Caution must be exercised in choosing step size and controlling cumulative error, especially in the vicinity of near-singular points where there is danger of migration between branches [28]. Algebraic curve-tracing is a crucial procedure in our trimmed-surface formulation, requiring a careful, tolerance-based implementation. Alternate methods for traversing an algebraic curve segment in a deterministic manner deserve further investigation.

4.11 Intersection of a line and a monotonic branch

In addition to identifying the monotonic branches of an algebraic curve, the curve-tracing procedure plays an important role in computing the intersections of a straight line and a monotonic branch C (25). Of special interest are the intersections of the coordinate lines $u = u_0$ and $v = v_0$ with C. We illustrate with the case $u = u_0$.

If $u_0 \notin [u_s, u_e]$, there are no intersections. If $u_0 \in [u_s, u_e]$, determine the roots of the univariate polynomial $F(u_0, v) = 0$ on $[v_s, v_e]$ (there must be at least one). If there is *only* one root, this gives the desired intersection. If there are *several* roots, all but one lie on the algebraic extension of C. The true intersection with the branch under consideration must then be selected by tracing the curve branch from u_s or u_e toward u_0 (**Figure 3**). If (u_0, v_0) is the intersection point thus determined, we may *split* the monotonic branch (25) at u_0 or v_0 into two portions $\{F(u, v) = 0: (u_s, v_s) \rightarrow (u_0, v_0)\}$ and $\{F(u, v) = 0: (u_0, v_0) \rightarrow (u_e, v_e)\}$. A portion of a monotonic branch is necessarily a monotonic branch itself.

4.12 Intersection of two monotonic branches

Given two monotonic branches C_1 and C_2 of the form

$$\{F_1(u, v) = 0: (u_{1s}, v_{1s}) \rightarrow (u_{1e}, v_{1e})\},$$
$$\{F_2(u, v) = 0: (u_{2s}, v_{2s}) \rightarrow (u_{2e}, v_{2e})\}, \tag{30}$$

we wish to determine their intersection points, if any. We begin by determining the bounding boxes (26) for each branch. If the boxes do not share common area, there are no intersections of the branches. Otherwise, an overlap box $[u_a, u_b] \times [v_a, v_b]$ is defined, and the branch intersections must lie inside it.

Eliminating, say, u from the two equations $F_1(u, v) = 0$ and $F_2(u, v) = 0$, we find the real roots v_i on $[v_a, v_b]$ of the

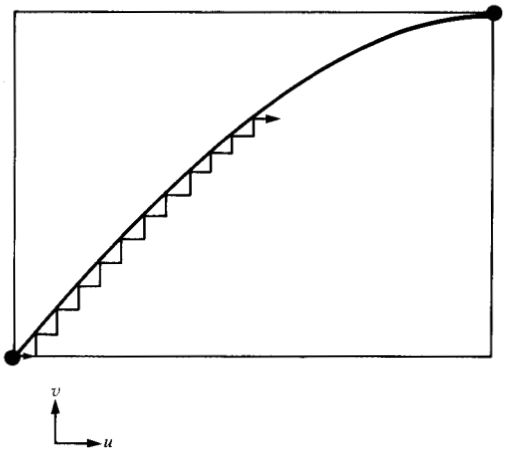


Figure 2

A monotonic algebraic curve branch, enclosed by its bounding box, illustrating the tracing of the curve by small steps in u using local power-series expansions.

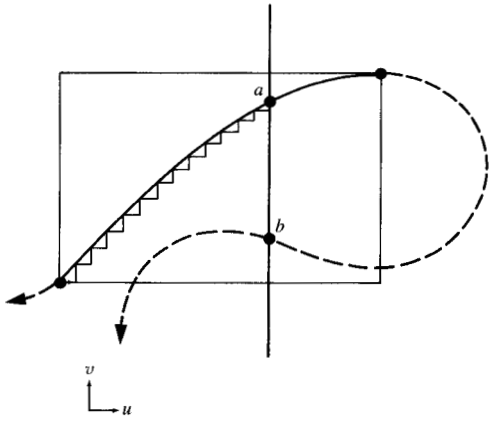


Figure 3

Computing the intersection of a straight line and a monotonic branch. Point a is a true intersection with the branch, while b is an intersection with its algebraic extension. The true intersection is identified by tracing along the branch from one of its endpoints.

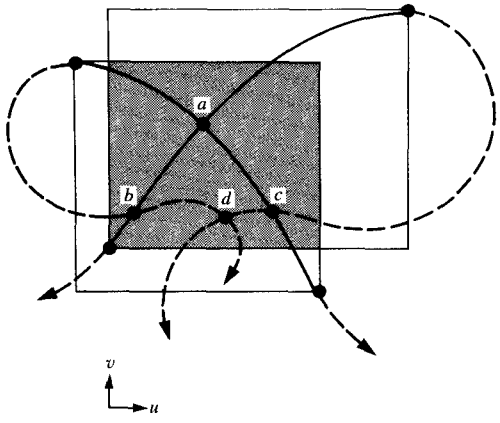


Figure 4

Computing the intersection of two monotonic branches. The overlap of the bounding boxes is the shaded area. Point a is a true intersection of the branches, b and c are intersections of one branch with the algebraic extension of the other, and d is an intersection of the algebraic extensions of both branches. True intersections are differentiated by tracing from an endpoint of *each* branch to the intersection point.

resultant. Corresponding values u_i for u are then determined and (u_i, v_i) is retained as a candidate intersection if u_i lies on $[u_a, u_b]$. The candidate intersections (u_i, v_i) fall into three categories: true intersections of the two branches, intersections of one branch with the algebraic extension of the other, and intersections of the two algebraic extensions (**Figure 4**). To verify a candidate as a true intersection of the branches, we must be able to trace *each* branch from one of its endpoints to the candidate point.

The above method serves for determining *proper* intersections of monotonic branches from distinct curves, i.e., the discrete points where the branches cross. Occasionally, however, the two branches (30) may derive from the same parent curve and their equations $F_1(u, v) = 0$ and $F_2(u, v) = 0$ are identical within a constant nonzero factor. The elimination procedure then yields an identity, and the intersection either is empty or degenerates to the continuous coincident portion of the branches. (Note that noncoincident branches from the same curve may at most share an endpoint, since singular points of an algebraic curve are always endpoints of its monotonic branches.)

4.13 Piecewise-algebraic loops

A piecewise-algebraic loop L is defined to be an ordered sequence of N (≥ 2) monotonic algebraic curve branches C_i connecting a set of N nodes (u_i, v_i):

$$\{F_i(u, v) = 0\colon (u_i, v_i) \rightarrow (u_{i+1}, v_{i+1})\}, \qquad i = 1 \rightarrow N, \qquad (31)$$

where $(u_{N+1}, v_{N+1}) \equiv (u_1, v_1)$. No restrictions are imposed on the definition (31) other than that the nodes be all distinct and that intersections between distinct branches are prohibited (self-intersections of branches are precluded by definition).

Piecewise-algebraic loops possess a definite *orientation*, clockwise or anticlockwise, as determined by the ordering of the nodes (u_i, v_i). The direction of the constituent branches of a loop is consistent with this orientation. Anticlockwise loops are denoted *positive* and clockwise loops *negative* (**Figure 5**). The piecewise-algebraic loop (31) defines a simply connected region in the (u, v) plane. By the monotonicity of the branches, the extent of a loop in the u and v directions is

$$u_{\min} = \min_i \{u_i\},\ v_{\min} = \min_i \{v_i\};$$

$$u_{\max} = \max_i \{u_i\},\ v_{\max} = \max_i \{v_i\}. \qquad (32)$$

4.14 Point classification and nesting of loops

A point-classification procedure for piecewise-algebraic loops may be formulated by considering their intersections with straight lines in the (u, v) plane. First, if the candidate point (u_0, v_0) lies outside the bounding box (32) for the entire loop, it must lie outside the loop itself. Otherwise, we classify (u_0, v_0) as lying inside, on the perimeter of, or outside the area of L, by determining the intersections of the coordinate lines $u = u_0$ or $v = v_0$ with each of the constituent branches of L (cf. Section 4.11). The bounding boxes (26) minimize unnecessary computation. The intersections divide the lines into inside/outside intervals; the location of the candidate point relative to these intervals gives the desired classification.

Occasionally, a procedure for *generating* arbitrary points within a piecewise-algebraic loop is also required. A simple (though possibly inefficient) method successively produces points with random distribution inside the bounding box (32) for the loop, subjecting each to the point/loop classification procedure and rejecting those which lie outside the loop.

A loop L_a is said to be nested within another loop L_b if the constituent branches of L_a lie entirely inside or on the perimeter of the area enclosed by L_b. To verify this we classify all the nodes of L_a with respect to L_b and test for proper intersections of each branch of L_a with every branch of L_b. If none of the nodes lie outside L_b and no proper intersections are found, the verification is complete. Similarly, two loops L_a and L_b are said to be disjoint if the constituent branches of L_a lie entirely outside or on the perimeter of the area enclosed by L_b, and vice versa. If all the nodes of each loop lie outside or on the perimeter of the other loop, and there are no proper intersections of the branches of the two loops, they are verified to be disjoint. Note that according to these definitions, loops which share common elements (nodes, branches, or portions of branches) may still be regarded as nested or disjoint.

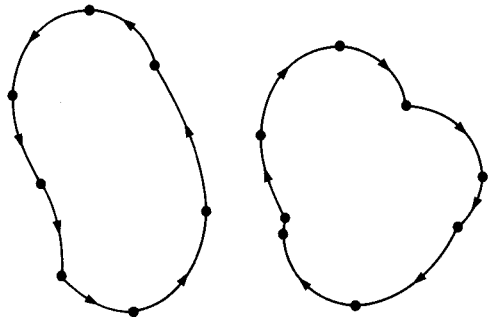


Figure 5

Schematic illustrations of piecewise-algebraic loops having positive and negative orientation.

4.15 Trimmed-surface formulation

A trimmed-surface patch is specified by the dual parametric and implicit equations (15) and (16) together with a *trimming boundary*, defined as a tree structure of nonintersecting, nested piecewise-algebraic loops (cf. **Figure 6**). There may be one or several loops at the top level of the tree structure. The branches of the tree issuing from a given loop point to other loops nested within it. Loops on the same nesting level are always disjoint. By convention, the orientation of the outermost or top-level loops is taken as anticlockwise in the (u, v) plane, and reverses with each successive nesting level. The area of the trimmed-surface patch then lies to the *left* in traversing any loop.

The area of the (u, v) plane contained within the piecewise-algebraic loop trimming boundaries is called the *parametric domain* of the trimmed patch, denoted Ω. The simplest trimmed surface is the entire patch (15) with the trivial trimming boundary

$$\{v = 0\text{: } (0, 0) \rightarrow (1, 0),\ u = 1\text{: } (1, 0) \rightarrow (1, 1),$$

$$v = 1\text{: } (1, 1) \rightarrow (0, 1),\ u = 0\text{: } (0, 1) \rightarrow (0, 0)\}. \qquad (33)$$

We refer to such a surface as a *complete* patch. General trimmed-surface elements have more complex trimming boundaries; often these include portions of the unit square boundary (33).

4.16 Area integrals over trimmed surfaces

In computing the mass properties of solids bounded by trimmed surfaces, we require a procedure to integrate a given function $\psi(u, v)$ over the parametric domain Ω of each patch:

$$I = \int_{\Omega} \psi(u, v) du dv. \qquad (34)$$

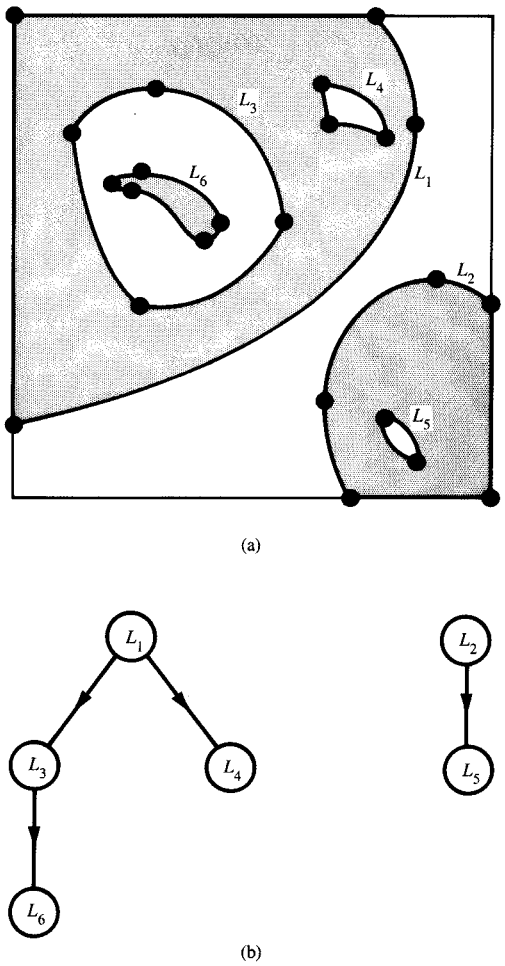


Figure 6

(a) Schematic example of a system of nested piecewise-algebraic loops defining a trimming boundary for a trimmed patch. The parametric domain Ω of the trimmed surface is the shaded area enclosed within the trimming boundary. (b) The tree describing the nesting behavior of the piecewise-algebraic loops which form the trimming boundary.

Numerical cubature formulae for estimating the integral (34) on elementary domains (e.g., triangles) over which $\psi(u, v)$ varies smoothly are well known [41, 42]. These methods have rapid convergence and may be formulated in an adaptive manner to optimize accuracy versus compute time, e.g., [43]. Our principal concern is thus to formulate tesselation algorithms for decomposing a complicated parametric domain Ω, defined by a given trimming boundary, into well-formed triangles or other elements, to which a suitable cubature rule may be applied.

We take Ω in (34) to be the area enclosed by a single piecewise-algebraic loop L. For domains defined by nested loops, we integrate over each loop individually and obtain

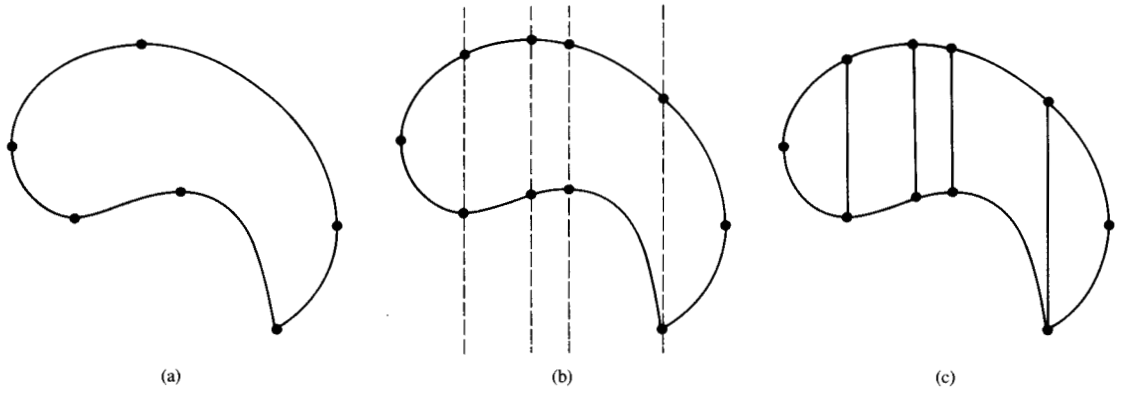


Figure 7

Cylindrical decomposition of a piecewise-algebraic loop: (a) the loop L under consideration; (b) splitting the branches of L through the nodes, parallel to the v-axis; and (c) the constituent cylindrical regions of L.

the integral (34) by adding the contributions of positive loops and subtracting those of negative loops. This technique is algorithmically simpler but perhaps less efficient than tesselating multiply connected domains. Several approaches are possible. For example, the curve-tracing procedure (Section 4.10) may be employed to polygonize L by selecting ordered sequences of discrete points along each branch; finite-element meshing algorithms for 2D polygonal boundaries may then be invoked [44]. The quadtree method [45] could also be employed to approximate the area of L by the successive subdivision of square elements.

Another approach is to perform a *cylindrical decomposition* of L [46] into strips parallel to, say, the v-axis. For each node (u_i, v_i) of the loop (31), we split each branch C_j at its intersections with the line $u = u_i$. The constituent branches of L may then be arranged in *stacks* defining cylindrical regions with vertical sidewalls and monotonic branches as upper and lower bounds (**Figure 7**). These cylindrical regions, together with the curve-tracing procedure, are relatively simple to process.

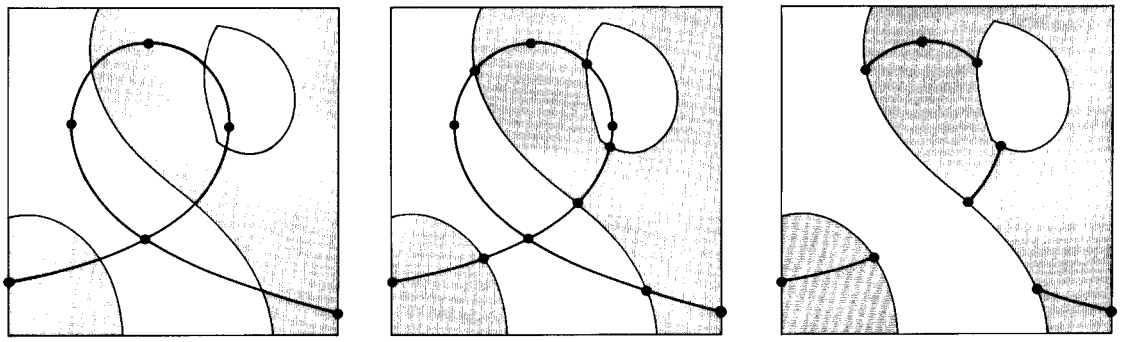
In principle, the area integral (34) may be transformed into a line integral around each circuit of the trimming boundary defining Ω by invoking the Stokes theorem [47]. In the present context, however, this approach has two drawbacks which render it inferior to direct numerical cubature—the lack of precise parametric representations (27) for the constituent branches of the trimming boundary, and the inability to determine the integrand of the line integral exactly in most cases.

4.17 Curves on trimmed surfaces

Given a parametric domain Ω contained within a trimming boundary of nested piecewise-algebraic loops, and a general algebraic curve $G(u, v) = 0$, we wish to ascertain which portions of this curve lie within Ω and to represent them as an evaluated system of monotonic branches. This algorithm forms the basis for computing section cuts through solids bounded by trimmed-surface elements (cf. Section 6.3).

The algorithm proceeds as follows: First, we determine the characteristic points for the *complete* curve $G(u, v) = 0$, and identify the set of monotonic branches they define (cf. Sections 4.6–4.10). We then compute the intersections of each monotonic branch of $G(u, v) = 0$ with each monotonic branch of the trimming boundary (Section 4.12), and split the branches of $G(u, v) = 0$ at these intersections. This process generates a new set of monotonic branches defining $G(u, v) = 0$ with the property that each branch lies entirely inside or outside the parametric domain Ω defined by the trimming boundary. Those branches lying outside Ω must be identified and discarded. This is accomplished by partially tracing each branch to a point intermediate between its endpoints and classifying that point with respect to Ω (cf. Section 4.14 and 6.1 below); if it lies outside the parametric domain, the branch must be discarded (**Figure 8**).

We have thus computed the representation of an algebraic curve $G(u, v) = 0$ defined on a trimmed-surface patch with parametric domain Ω. The representation consists of a collection of monotonic branches connecting a set of characteristic points. The characteristic point set comprises a subset of the border, turning, and singular points for the complete curve $G(u, v) = 0$ (Section 4.7), augmented by a new type of border point—the intersections of the complete curve $G(u, v) = 0$ with the trimming boundary for Ω. These new border points also have a link multiplicity of 1. Under exceptional circumstances, however, they may coincide with



Evaluating an algebraic curve $G(u, v) = 0$ on a trimmed surface with parametric domain Ω: (a) monotonic branches of the complete curve superposed on Ω (the shaded area); (b) splitting the branches at the intersections with the trimming boundary; and (c) branches outside Ω rejected by classifying intermediate points with respect to the parametric domain.

turning points or singular points, and the link multiplicity must be adjusted accordingly.

4.18 Solid boundary files

The representation of a solid bounded by trimmed-surface patches of the form defined above is called a *boundary file* when at least one of the patch trimming boundaries is not the trivial case (33). The complex nature of the elements comprising a boundary file necessitates a sophisticated data structure for representing the adjacency and incidence relationships of the faces, edges, and vertices of the solid. The solid interrogation algorithms (cf. Section 6) also require such organized representations for their reliable and efficient operation.

Although boundary-file data structures are not the immediate concern of the present study—detailed discussions are available elsewhere, e.g., [48]—we note that the boundary-file data structure and trimmed-surface formulation impose mutual constraints upon each other (see Section 5.1) which deserve further attention. The trimmed-surface formulation and algorithms described in this section lead to the present definition of a boundary file, but do not describe how it is actually computed in specific Boolean combination scenarios. That is the subject of the following section, where the motivation for the present formulation becomes apparent.

5. Boolean combination

We now outline procedures for deriving the boundary files arising from Boolean solid combinations. The emphasis here is on the derivation of representations for the individual trimmed-surface elements produced by such combinations, rather than the data structures expressing their topological relationships, issues of computational efficiency, or a detailed enumeration of the various stages of the boundary evaluation procedure *per se* (the latter subject is discussed, independent of any specific trimmed-surface formulation, in [5]). We consider three solid combination processes of increasing computational complexity: the combination of two primitives, the combination of a primitive and a boundary file, and the combination of two boundary files.

In the first case, only intersections between complete patches arise. This simplifies the boundary-file computation and serves to illustrate its salient features. However, algorithms which perform primitive combination only cannot be employed in further modifying the solid, and are therefore of limited practical utility.

In the second case, intersections between a complete patch and a trimmed patch occur. The solid is successively modified by unioning, differencing, or intersecting it with a new primitive at each stage of a linear constructive solid geometry (CSG) tree **[Figure 9(a)]**. The first stage corresponds to case one. This linear process avoids the difficult combination of two boundary files. Although certain solids defined by general CSG trees (which require boundary-file combinations) cannot be created by this incremental method, many commonly encountered mechanical components are accommodated.

In the final case one encounters complex intersections between trimmed-surface pairs. Boundary-file combination or *merging* capabilities are required to evaluate solids defined by a general binary CSG tree **[Figure 9(b)]**, and

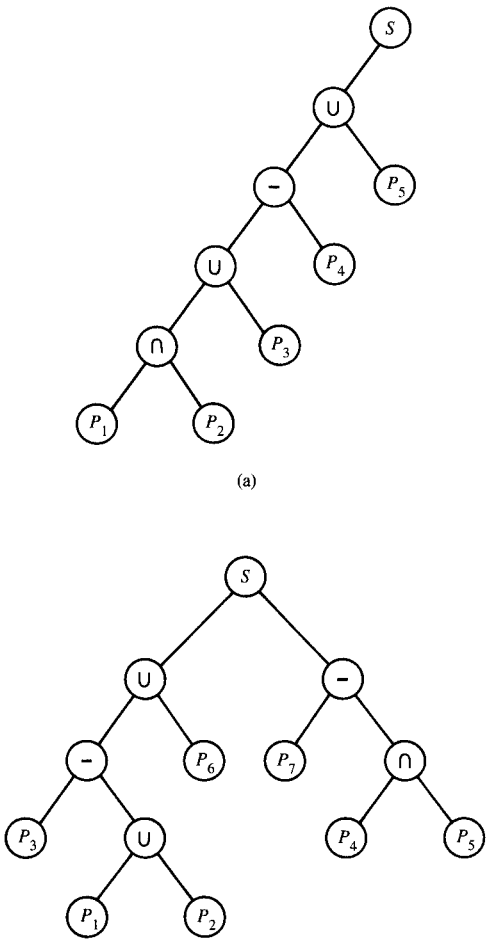


Figure 9

(a) Schematic illustration of a solid S defined by a linear CSG tree acting on a set of primitives P_1, P_2, etc. The first Boolean operation combines two primitives, but all subsequent ones combine a primitive with a boundary file (the combination of two boundary files is avoided). (b) A solid S defined by a general binary CSG tree. Early Boolean operations combine two primitives or a primitive and a boundary file, but subsequent stages involve the combination of two boundary files.

provide the greatest flexibility in creating and modifying solids. However, the algorithms required are substantially more complex.

5.1 Combination of primitives

Let P_1 and P_2 denote two solid primitives as defined in Section 4.3, bounded by N_1 and N_2 patches, respectively. The first step in the boundary evaluation of Boolean combinations of these primitives is to compute the partial intersections of each patch i of P_1 with each patch j of P_2. Of the N_1N_2 possible intersections, a substantial fraction are expected to be null. Most of these are bypassed by bounding-box interference tests (Section 4.4).

Let patch i of P_1 be specified by the dual equations $\{\mathbf{r} = \mathbf{r}(s, t), f(x, y, z) = 0\}$ and patch j of P_2 by $\{\mathbf{r} = \mathbf{r}(u, v), g(x, y, z) = 0\}$. Substituting $\mathbf{r}(s, t)$ into $g(x, y, z) = 0$ and $\mathbf{r}(u, v)$ into $f(x, y, z) = 0$, we obtain the algebraic curve equations $G(s, t) = 0$ and $F(u, v) = 0$ in the unit parameter squares of patch i of P_1 and patch j of P_2, respectively. The partial intersection is a subset of these curves. We identify the correct subsets and evaluate them as collections of monotonic branches as follows.

Consider the curve $G(s, t) = 0$ on patch i of P_1 (symmetric arguments are employed in processing $F(u, v) = 0$ on patch j of P_2). First, we identify the characteristic points for the *complete* curve $G(s, t) = 0$—the border, turning, and singular points—and compute their link multiplicities. We then determine the monotonic branches delineated by these characteristic points by the curve-tracing procedure, ensuring that all link multiplicities are satisfied (cf. Sections 4.6–4.10).

A set of *termination points* is then identified. These occur where the border curves of patch j of P_2 cross the surface of patch i of P_1 within the unit parameter square; i.e., they are the image points on $\mathbf{r}(s, t)$ of the border points of $F(u, v) = 0$. Only those lying in the unit parameter square for $\mathbf{r}(s, t)$ are retained. These termination points lie on the monotonic branches of $G(s, t) = 0$ (**Figure 10**).

We split each monotonic branch at the termination points which lie on it. This generates a new set of monotonic branches defining $G(s, t) = 0$, with the following property: Each branch belongs entirely to the intersection of the finite patches $\mathbf{r}(s, t)$ and $\mathbf{r}(u, v)$, or to the intersection of $\mathbf{r}(s, t)$ with the algebraic extension of $\mathbf{r}(u, v)$ (cf. Section 4.2). Only the monotonic branches of the first category should be retained for the partial intersection.

These are identified as follows. We partially trace each branch to an intermediate point (s_i, t_i) between its endpoints, and compute its image (u_i, v_i) on $\mathbf{r}(u, v)$. The branch is retained if $(u_i, v_i) \in [0, 1] \times [0, 1]$; otherwise it is discarded. In the final representation of the partial intersection as a collection of monotonic branches, termination points may be regarded as having a link multiplicity of 1.

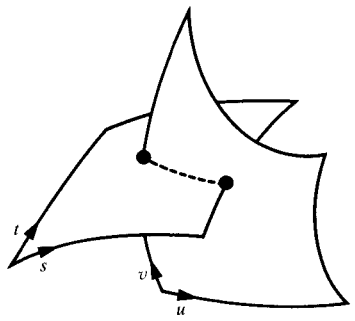
The procedure described above is similar to that employed to evaluate the portions of a general algebraic curve within the parametric domain Ω of a trimmed-surface patch (Section 4.17). The main difference is that in the present situation there is no unambiguously defined parametric domain, and monotonic branches must be retained or discarded by reference to the other patch participating in the intersection.

The above process may be repeated to evaluate the partial intersection subset of $F(u, v) = 0$ in the parameter space of patch j of P_2. Note that in the partial intersection of two

patches there is a one-to-one correspondence between border points and termination points in the respective parameter spaces of the patches. Such a correspondence also holds for singular points, but *not* for turning points. A turning point of $G(s, t) = 0$ is not uniquely identified with a turning point of $F(u, v) = 0$, and even the *number* of turning points for the partial intersection representations in the two parameter spaces may differ.

A well-structured boundary-file data structure should allow a unique identification between portions of the intersection curve, as represented in the parameter space of each patch. The lack of correspondence between turning points implies that such an identification does not hold for the monotonic branches as determined above. However, there is a simple remedy for this dilemma (**Figure 11**):

We image the turning points in one parametric space into the other parametric space and vice versa, splitting the monotonic branches of the partial intersection representation in each case. This additional splitting of monotonic branches guarantees a unique correspondence of branches for the partial intersection representation in the two parameter spaces. The splitting introduces redundancy in the definition of the partial intersection. This does not necessitate any alteration in the trimmed-surface formulation or interrogation algorithms, but may induce some degradation of computational efficiency.



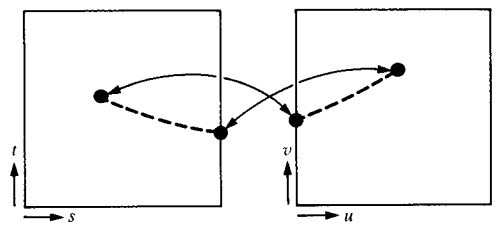


Figure 10

Partial intersection of two finite patches $\mathbf{r}(s, t)$ and $\mathbf{r}(u, v)$. Border points of the intersection curve representation on $\mathbf{r}(s, t)$ are imaged to termination points of the representation on $\mathbf{r}(u, v)$, and vice versa.

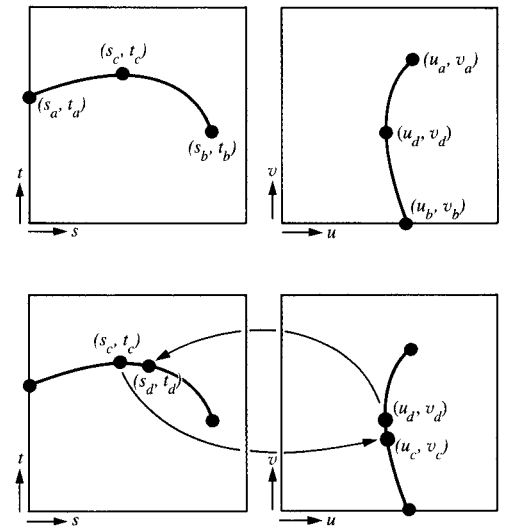


Figure 11

Imaging of turning points to establish a unique correspondence of branches in the partial intersection of two patches. The border point (s_a, t_a), and termination point (u_a, v_a) are mutual images, as are the termination point (s_b, t_b) and the border point (u_b, v_b). However, the turning points (s_c, t_c) and (u_d, v_d) are *not* mutual images. By splitting $G(s, t) = 0$ at the image (s_d, t_d) of (u_d, v_d) on $\mathbf{r}(s, t)$, and $F(u, v) = 0$ at the image (u_c, v_c) of (s_c, t_c) on $\mathbf{r}(u, v)$, the branches of $G(s, t) = 0$ and $F(u, v) = 0$ become unique images of each other.

Once the partial intersections of each patch $i = 1, 2, \cdots, N_1$ of P_1 with each patch $j = 1, 2, \cdots, N_2$ of P_2 have been computed, we consider each patch of the two primitives in turn. If no partial intersections have been found, the trimming boundary is the unit square (33). If partial intersections do occur, we chain their constituent monotonic branches together by identifying branches with common endpoints. Such chains either form closed piecewise-algebraic loops within the unit square, or open tracks which terminate at border points on the unit-square boundary.

By ordering the border points around the perimeter of the unit-square boundary (33), it is possible to construct a set of piecewise-algebraic loops from the open tracks and portions of the unit-square boundary which cover the entire area of the unit square (**Figure 12**). These are called *principal loops*—all other loops in the unit square generated by partial intersections must be nested within a principal loop. The nesting relationships of these loops are determined by the procedures described in Section 4.14. If no border points occur, there is only a single principal loop, the unit-square boundary (33).

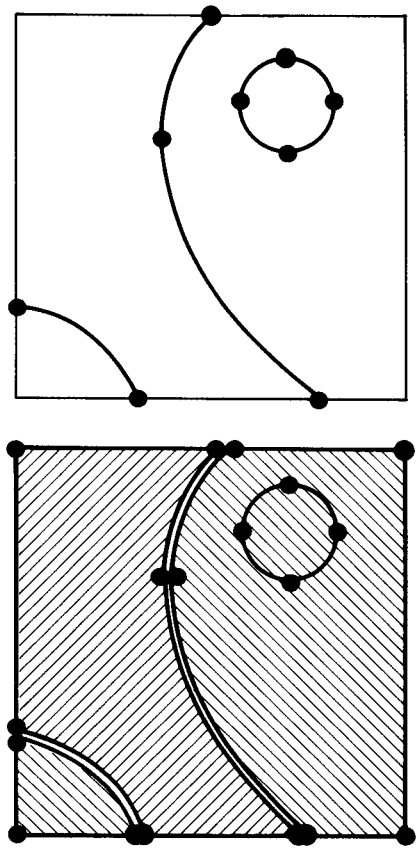


Figure 12

Consolidated partial intersections in the parameter space of a patch, forming closed piecewise-algebraic loops or open tracks which terminate in border points on the unit-square boundary (upper), and principal loops formed from the open tracks and portions of the unit square (lower). Loops of different polarity are crosshatched in different directions.

Consider a single patch $\mathbf{r}(s, t)$ on primitive P_1. Each principal loop is assigned a polarity, positive or negative, according to whether we wish to retain the interior or exterior trimmed surface of P_1 with respect to P_2 (cf. Section 2). The polarity is determined by obtaining a point (s_0, t_0) inside the principal loop, but *outside* all loops nested within it, projecting it to three dimensions, $\mathbf{r}_0 = \mathbf{r}(s_0, t_0)$, and classifying the Cartesian point $\mathbf{r}_0$ with respect to primitive P_2 (cf. Section 4.3). The polarity is positive when $\mathbf{r}_0$ is outside P_2 and we want the exterior trimmed surface, or $\mathbf{r}_0$ is inside P_2 and we want the interior trimmed surface; otherwise it is negative. Note that adjacent principal loops must have opposite polarity.

The tree structure of piecewise-algebraic loops may now be assembled. Each principal loop of positive polarity is an initial node of the tree. For principal loops of negative polarity, each contained loop at the first nesting level (if any) is also an initial node of the tree (the principal loop of negative polarity itself is discarded). Since the nesting structure for each principal loop has been already determined, the remaining tree structure is readily available.

We impose positive (anticlockwise) orientation on the loops at the initial nodes of the tree, and reverse the orientation with each successive nesting level. This completes the trimmed-surface definition, which consists of the implicit and parametric equations (15) and (16), the specifications (31) of a collection of piecewise-algebraic loops, and the tree structure (Figure 6) describing their nesting relationships.

5.2 Combination of a primitive and a boundary file

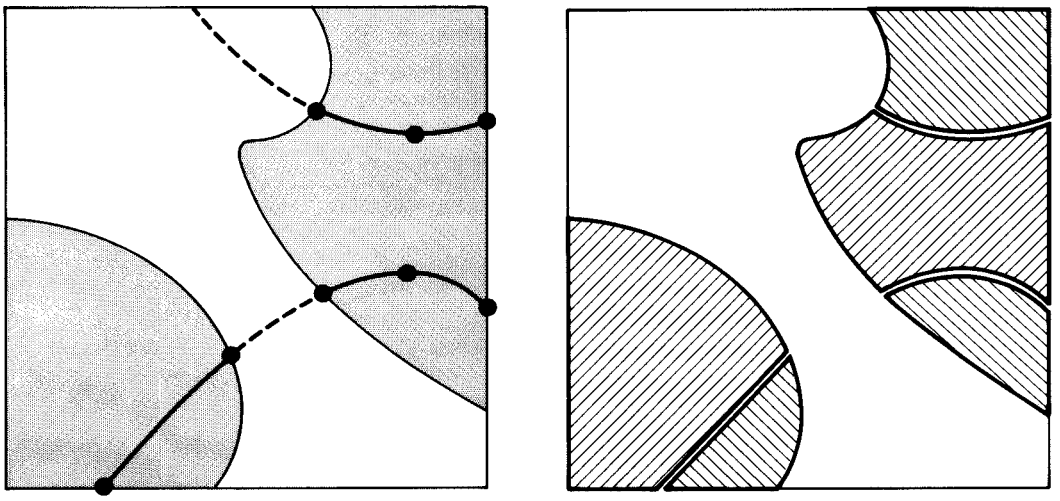
Let P denote a solid primitive bounded by N_p complete patches, and B a solid boundary file comprising N_b trimmed patches. In the boundary evaluation of a Boolean combination of P and B, many of the arguments for combining two primitives (see above) apply, and we discuss here only the significant modifications required for the procedure.

Consider the partial intersection of complete patch i of P, $\{\mathbf{r} = \mathbf{r}(s, t), f(x, y, z) = 0\}$, and trimmed patch j of B, $\{\mathbf{r} = \mathbf{r}(u, v), g(x, y, z) = 0\}$ with parametric domain Ω, defined by a trimming boundary of nested piecewise-algebraic loops (Section 4.15). We wish to determine the subsets of the algebraic curves $G(s, t) = 0$ on the complete patch and $F(u, v) = 0$ on the trimmed patch which describe the partial intersection in the parameter space of each patch.

We evaluate as collections of monotonic branches the complete curve $G(s, t) = 0$ on the unit parameter square (Sections 4.6–4.10), and the curve $F(u, v) = 0$ on the parametric domain Ω (Section 4.17). In the latter case, the border points are the intersections of the complete curve $F(u, v) = 0$ with the trimming boundary for Ω. Termination points are then introduced for $G(s, t) = 0$ as the images of the border points of $F(u, v) = 0$ on the parametric domain Ω, and for $F(u, v) = 0$ as the images of the border points of $G(s, t) = 0$ on the unit square. The monotonic branches are split at these termination points in each case.

After splitting, each monotonic branch of $G(s, t) = 0$ either belongs entirely to the intersection of the complete patch $\mathbf{r}(s, t)$ and the trimmed patch $\mathbf{r}(u, v)$, or to the intersection of the complete patch $\mathbf{r}(s, t)$ with the extensions of $\mathbf{r}(u, v)$ beyond the parametric domain Ω. We partially trace each branch to a point (s_i, t_i) intermediate between its endpoints, and find its image (u_i, v_i). The branch is retained only if $(u_i, v_i) \in \Omega$.

Similarly, each monotonic branch of $F(u, v) = 0$ either belongs entirely to the intersection of the complete patch



(a) (b)

Figure 13

(a) Consolidated partial intersections on the parametric domain Ω (shaded area) of a trimmed patch. The dashed curves are intersections with the complement of Ω. (b) Principal loops formed from the partial intersections and portions of the trimming boundary for Ω. Loops of different polarity are crosshatched in different directions.

$\mathbf{r}(s, t)$ and the trimmed patch $\mathbf{r}(u, v)$, or to the intersection of the trimmed patch $\mathbf{r}(u, v)$ with the algebraic extension of the complete patch $\mathbf{r}(s, t)$. Partially tracing each branch to a point (u_i, v_i) intermediate between its endpoints, we retain the branch only if its image point satisfies $(s_i, t_i) \in [0, 1] \times [0, 1]$. As before, we can perform imaging of turning points between the parameter spaces of the complete and trimmed patches, and splitting of branches at the image points, if a unique identification between branches in the two parameter spaces is desired.

The next step is to form the principal loops and assign their polarities. For the complete patch, this is done exactly as described above in the combination of two primitives. For the trimmed patch, the principal loops are formed from open tracks of monotonic branches, generated by partial intersections, plus portions of the trimming boundary (the branches of the trimming boundary must be split at the border points where the open tracks meet it). Taken together, the principal loops cover the entire parametric domain Ω (**Figure 13**).

The polarities are assigned by taking a parametric point inside each principal loop (but outside all loops nested within it), projecting to three dimensions, and classifying the resulting Cartesian point with respect to the boundary file B (cf. Section 6.2) or primitive P, as appropriate. The nesting behavior of the piecewise-algebraic loops within each principal loop is determined as described in Section 4.14, and the tree structure for the trimmed surface is then derived from the principal loops as described in Section 5.1. When each piecewise-algebraic loop has been assigned its canonical orientation, the trimmed-surface definition is complete.

5.3 Combination of boundary files

Finally, we consider the boundary evaluation of the Boolean combination of two solid boundary files, B_1 and B_2, comprising N_1 and N_2 trimmed-surface patches, respectively. Again, we address only the issues arising in this case which are significantly different from the procedures described above.

Consider the partial intersection of patch i of B_1, defined by $\{\mathbf{r} = \mathbf{r}(s, t), f(x, y, z) = 0\}$ and parametric domain Ω_1, and patch j of B_2, defined by $\{\mathbf{r} = \mathbf{r}(u, v), g(x, y, z) = 0\}$ and parametric domain Ω_2. We must identify the partial intersection subsets of the algebraic curves $G(s, t) = 0$ and $F(u, v) = 0$ and evaluate them as collections of monotonic branches.

We consider the curve $G(s, t)$ on patch i of B_1 (symmetric arguments are employed in processing $F(u, v) = 0$ on patch j of B_2). We evaluate $G(s, t) = 0$ with respect to the parametric domain Ω_1 (cf. Section 4.17) of the trimmed patch $\mathbf{r}(s, t)$.

Termination points are then introduced on the resulting monotonic branches, as the images of the border points of $F(u, v) = 0$ evaluated on Ω_2. The monotonic branches are split at these termination points.

After splitting, each monotonic branch either belongs entirely to the intersection of the trimmed patches $\mathbf{r}(s, t)$ and $\mathbf{r}(u, v)$, or to the intersection of the trimmed patch $\mathbf{r}(s, t)$ with the extensions of $\mathbf{r}(u, v)$ beyond the parametric domain Ω_2. Partially tracing each branch to a point (s_i, t_i) intermediate between its endpoints, we retain the branch only if its image point satisfies $(u_i, v_i) \in \Omega_2$.

The above process is repeated to evaluate the partial intersection subset of $F(u, v) = 0$ on patch j of B_2. Imaging of turning points between the parametric spaces of the patches may be performed to establish a unique correspondence of branches. The identification of the principal loops and the construction of the tree describing the nesting behavior of the piecewise-algebraic loops in the trimming boundary proceed in a manner analogous to that described previously.

6. Interrogation algorithms

A boundary-file representation for a solid model generated by Boolean operations is of limited value if *interrogation algorithms*, answering elementary queries concerning the model, cannot be formulated or are too complex to implement. In this section we outline some basic interrogation procedures for boundary files derived from the trimmed-surface formulation presented above. More general interrogation functions may often be synthesized from these basic procedures.

The reader should not infer that we advocate boundary evaluation as a recommended intermediate step in performing these interrogations. Although empirical evidence is sparse, it is probable that if the initial solid specification is a CSG tree, for example, ray-tracing and point classification may be performed more efficiently and reliably directly from that representation. However, a boundary formulation of these algorithms is still important for cases where no other representation is available. Additionally, for the case of volumetric or *mass properties* integrations, boundary methods offer potentially the most accurate and systematic approach [49].

6.1 Ray-tracing of boundary files

The intersections of a ray with a solid boundary file are determined by finding the intersections with each of its constituent trimmed-surface patches. Testing the ray against the bounding boxes for the *complete* patches associated with each trimmed-surface patch (cf. Section 4.4) minimizes unnecessary computation. (The problem of computing true bounding boxes for *trimmed* patches is quite complex, and the time saved by having such boxes might not offset the cost of computing them.)

To determine the intersection of a ray with a trimmed-surface patch, we first determine the intersections (u_i, v_i) with the complete patch (cf. Section 4.3). Each of these must then be classified as lying inside or outside the parametric domain Ω of the trimmed patch. To accomplish this, we classify the points with respect to each of the piecewise-algebraic loops of the trimming boundary (cf. Section 4.14). If a candidate intersection point does not lie inside any of the loops, it is outside the parametric domain Ω. If it lies inside one or more of the loops, these loops must be nested, and we determine the orientation of the *innermost* loop. If this is clockwise, the candidate intersection is outside Ω; if anticlockwise, it lies inside Ω.

An obvious application of the ray-tracing technique for solid boundary files is the generation of high-resolution shaded raster images. It should be noted, however, that if the boundary file results from the evaluation of a Boolean CSG tree acting on a set of primitives, it is almost certainly more efficient to ray-trace the primitives directly, and then perform the Boolean operations on the in/out intervals along the ray generated by each primitive.

6.2 Point classification for boundary files

A candidate point $\mathbf{p}$ may be classified as lying inside, on the boundary of, or outside the volume V defined by a boundary file B by considering the intersections of a straight line through $\mathbf{p}$ with the trimmed-surface patches of B. The procedure is analogous to that for solid primitives (Section 4.3), except that the intersections of the line with *trimmed* rather than complete patches must be determined (see the preceding section). Note again that point/solid classification may be performed more efficiently directly from the Boolean CSG tree in most cases.

6.3 Sectioning of boundary files

Automated manufacturing applications driven directly from solid boundary representations are still in their infancy, and even in the most immediate application domain of solid modeling—detailed design and drafting—there is still a heavy reliance on two-dimensional blueprints as the final production specification. In this context, the ability to compute, annotate, and dimension planar section cuts through a solid boundary file is of crucial importance. More complex sectioning surfaces also frequently arise [28], for example in aerodynamic applications, and we consider below the general problem of sectioning a boundary file by an unbounded surface defined by a low-degree implicit equation $f(x, y, z) = 0$ (cf. Table 1).

Substituting the parametric form (15) for each trimmed patch into the implicit equation of the sectioning surface, we obtain an algebraic curve equation of the form (20) for the section through that patch. The segments of this curve which lie in the parametric domain Ω of the trimmed patch are then identified and broken up into monotonic branches by

the technique described in Section 4.17. It is then usually desirable to approximate these by three-dimensional parametric arcs. This may be accomplished by splining through an ordered set of discrete points sampled along the branch and projected to three dimensions. Alternately, the inherent smoothness of monotonic branches suggests developing special algorithms for interpolating the endpoints, tangent directions, and other differential properties of a branch, projected to three dimensions, by single parametric polynomial arcs.

A systematic boundary-file data structure, describing the adjacency relationships of the trimmed-surface elements, provides a means for ordering the parametric space arcs resulting from the section computation (this subject is beyond the scope of the present study). An important related issue is the treatment of arc end conditions where the section curve traverses the boundary between adjacent trimmed patches. If surface tangent continuity obtains across the boundary, the three-dimensional approximation procedure for monotonic branches should ensure tangent continuity for the section curve also. This is unlikely to be true if the trimming boundary branch under consideration results from an intersection of primitives; however, it is true if the branch is part of the unit square (33) and the original primitive is tangent-continuous across patch boundaries.

6.4 Mass properties for boundary files

Perhaps the most important applications of boundary files are in the estimation of volumetric integrals (mass properties) of complex solids. While simpler algorithms are readily available based on polyhedral approximation [50, 51], Monte Carlo methods, cellular decomposition, etc. [49], the surface-integral form for boundary files is potentially the most accurate, since it is based on an essentially precise representation of the solid.

Let a boundary file B enclosing a solid volume V be composed of N trimmed-surface patches with parametric equations $\mathbf{r}_i(u, v)$, and let Ω_i denote the parametric domain contained within the trimming boundary of patch i. A surface integral over B has the general form

$$\sum_{i=1}^{N} \int_{\Omega_i} \psi_i(u, v) du dv, \tag{35}$$

where $\psi_i(u, v)$ is a function of the parametric coordinates on the domain Ω_i of patch i. Several important solid properties are easily expressed in the surface integral form (35) by invoking the Gauss divergence theorem

$$\int_V \nabla \cdot \mathbf{F}\, dxdydz = \int_B \mathbf{F} \cdot d\mathbf{A}, \tag{36}$$

$\mathbf{F}$ being an arbitrary vector field defined in V, and $d\mathbf{A}$ a vector surface-area element of the boundary B.

On the trimmed-surface patch $\mathbf{r} = \mathbf{r}_i(u, v)$, the vector area element is

$$d\mathbf{A} = \mathbf{S}_i(u, v) du dv, \quad \text{where} \quad \mathbf{S}_i(u, v) = \frac{\partial \mathbf{r}_i}{\partial u} \times \frac{\partial \mathbf{r}_i}{\partial v}. \tag{37}$$

We assume that the surfaces under consideration have nonsingular parameterizations, i.e., $| \mathbf{S}_i(u, v) | \neq 0$ over the parametric domain Ω_i of each patch. To evaluate the surface integrals over each trimmed patch in (35), the numerical procedures described in Section 4.16 must be invoked. In principle, the adaptive nature of these schemes allows any prescribed precision to be realized in the integration.

6.4.1 Surface area

The surface area of the solid is obtained by integrating the scalar magnitude of $d\mathbf{A}$ over each trimmed patch. Hence, we integrate the functions

$$\psi_i(u, v) = |\mathbf{S}_i(u, v)| \tag{38}$$

over the parametric domain of each patch in Equation (35).

6.4.2 Volume

By setting $\mathbf{F} = \mathbf{r}$ in the Gauss theorem (36), the volume V of the solid may be expressed as a surface integral of the functions

$$\psi_i(u, v) = \tfrac{1}{3}\, \mathbf{r}_i(u, v) \cdot \mathbf{S}_i(u, v) \tag{39}$$

over the parametric domain of each trimmed patch in Equation (35). For uniform density ρ, the mass of the solid is $M = \rho V$.

6.4.3 Center of gravity

The center of gravity $\mathbf{r}_c = (x_c, y_c, z_c)$ of a homogeneous solid is defined by

$$V\mathbf{r}_c = \int_V \mathbf{r}\, dxdydz. \tag{40}$$

The x, y, z components of the right-hand side of (40) may be expressed as surface integrals by choosing the vector fields

$$\mathbf{F} = \tfrac{1}{4} x\mathbf{r}, \tfrac{1}{4} y\mathbf{r}, \tfrac{1}{4} z\mathbf{r}, \tag{41}$$

respectively, for $\mathbf{F}$ in the Gauss theorem (36). For x_c this yields the integrands

$$\psi_i(u, v) = \tfrac{1}{4}\, x_i(u, v)\mathbf{r}_i(u, v) \cdot \mathbf{S}_i(u, v) \tag{42}$$

on each trimmed patch, where $x_i(u, v)$ is the x-component of $\mathbf{r}_i(u, v)$. Symmetric expressions obtain when computing y_c and z_c.

6.4.4 Inertia tensor

For uniform density ρ the components of the solid inertia tensor [52] are defined by

$$I_{xx} = \int_V (y^2 + z^2)\rho dxdydz,$$

$$I \quad = -\int xy\rho dxdydz, \text{ etc.} \tag{43}$$

Since the inertia tensor is symmetric, only six of the nine components (43) are independent. The vector fields

$$\mathbf{F} = \tfrac{1}{5}\rho(y^2 + z^2)\mathbf{r}, \quad -\tfrac{1}{5}\rho xy\mathbf{r}, \quad \text{etc.}, \tag{44}$$

may be employed to transform the inertia components (43) into surface integral expressions, using the Gauss theorem (36). The integrands for each trimmed patch then have the form

$$\psi_i(u, v) = \tfrac{1}{5}\rho[y_i^2(u, v) + z_i^2(u, v)]\mathbf{r}_i(u, v) \cdot \mathbf{S}_i(u, v),$$

$$\psi_i(u, v) = -\tfrac{1}{5}\rho x_i(u, v)y_i(u, v)\mathbf{r}_i(u, v) \cdot \mathbf{S}_i(u, v), \text{ etc.} \tag{45}$$

in (35). Once $I_{xx}, I_{xy}, \cdots$ are known, the inertia components may be referred to the center of gravity (40) and the principal axes and principal moments are then easily determined.

6.4.5 Other volume integrals

In general, the integral of an arbitrary scalar field $\Phi(x, y, z)$ over the volume of a solid

$$I = \int_V \Phi(x, y, z)\,dxdydz \tag{46}$$

may be transformed to the form (35) appropriate to boundary representations if $\Phi(x, y, z)$ is an integrable function of the coordinates. A vector field $\mathbf{F} = (F_x, F_y, F_z)$ satisfying

$$\frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z} = \Phi(x, y, z) \tag{47}$$

must be found. Solutions to (47) are indeterminate to the extent of the curl of an arbitrary vector field; i.e., if $\mathbf{F}$ is a solution, then

$$\mathbf{F}' = \mathbf{F} + \nabla \times \mathbf{G} \tag{48}$$

is also, $\mathbf{G}$ being an arbitrary vector field. A simple approach is to assume the three terms on the left of (47) to be equal, and thus take the components of $\mathbf{F}$ to be the indefinite integrals

$$F_x = \tfrac{1}{3}\int \Phi dx, \; F_y = \tfrac{1}{3}\int \Phi dy, \; F_z = \tfrac{1}{3}\int \Phi dz. \tag{49}$$

The integrands in the surface integral (35) then become

$$\psi_i(u, v) = \mathbf{F}[x_i(u, v), y_i(u, v), z_i(u, v)] \cdot \mathbf{S}_i(u, v), \tag{50}$$

where $\mathbf{F}(x, y, z)$ is expressed *on the surface* of patch i.

7. Summary and conclusions

We have outlined algorithms for representing and processing trimmed-surface elements, i.e., finite portions of unbounded analytic surfaces with complex border curves. A unified approach was adopted in which the intimately coupled problems of surface intersection computation and trimmed-surface representation have been addressed in a consistent manner; all surfaces belonged to a single canonical form and were thus amenable to processing by the same algorithms. The trimmed-surface formalism was based, as far as possible, on deterministic mathematical procedures such as the elementary arithmetic operations and univariate polynomial root-solving.

Errors in geometric computations may be regarded as originating from two sources: heuristics in the mathematical or algorithmic formulation of the problem, and the vagaries of practical floating-point software implementation of the proposed solution. In setting the surface intersection and trimming problem on a sounder mathematical footing, we have attempted to minimize the former source. Any advances in the accuracy and reliability of computer arithmetic or root-solving algorithms [10] can thus be imported into the methods described here with immediate benefit.

Our algorithms are amenable, in principle, to surfaces of arbitrary degree. In practice, however, the complexity of surfaces which can be realistically addressed will be constrained by the degree of the univariate polynomials arising in the processing of those surfaces. As an extreme example, the intersection of two bicubic patches yields an algebraic curve of the form (20) of degree 54 in each of u and v. Determining the characteristic points of such a curve would then incur an intractable univariate resultant polynomial of degree 5778. However, we believe that a family of simpler surfaces of interest (cf. Table 1) can be systematically and reliably processed.

We have vindicated the practical value of the trimmed-surface methods presented here by providing some well-defined interrogation procedures for solids bounded by such surface elements (cf. Section 6). In particular, the surface integral formulation provides a reliable and systematic means of estimating mass properties for complex solids.

In concluding, we highlight areas for further research consolidating or complementing the present study. Perhaps the weakest link in the surface-intersection computation is the algebraic curve-tracing procedure (cf. Section 4.10). Since this plays a central role in identifying and processing monotonic branches, a more deterministic method is desirable (a tolerance-based variant of the present technique might suffice). Important areas that we have given only cursory treatment or none at all are the Boolean combination algorithms that generate the trimmed-surface elements of a boundary file, and data structures for representing their adjacency relationships in easily accessible form. Finally, we remark that while some of the algorithms presented here have been validated by actual software implementation [28], the remainder require further practical exploration and testing.

References

1. A. A. G. Requicha and H. B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment," *IEEE Comput. Graph. & Appl.* **2**, No. 2, 9–24 (1982).

2. A. A. G. Requicha and H. B. Voelcker, "Solid Modeling: Current Status and Research Directions," *IEEE Comput. Graph. & Appl.* **3,** No. 7, 25–37 (1983).
3. S. W. Thomas, "Modelling Volumes Bounded by B-Spline Surfaces," Ph.D. thesis, University of Utah, Salt Lake City, 1984.
4. M. S. Casale, "Free-Form Solid Modeling with Trimmed Surface Patches," *IEEE Comput. Graph. & Appl.* **7,** No. 1, 33–43 (1987).
5. A. A. G. Requicha and H. B. Voelcker, "Boolean Operations in Solid Modeling: Boundary Evaluation and Merging Algorithms," *Proc. IEEE* **73,** No. 1, 30–44 (1985).
6. R. F. Sarraga and W. C. Waters, "Free-Form Surfaces in GMSOLID: Goals and Issues," *Solid Modeling by Computers—From Theory to Applications*, M. S. Pickett and J. W. Boyse, Eds., Plenum Press, New York, 1984.
7. A. A. G. Requicha and H. B. Voelcker, "Constructive Solid Geometry," *Technical Memo 25*, Production Automation Project, University of Rochester, Rochester, NY, 1977.
8. R. B. Tilove and A. A. G. Requicha, "Closure of Boolean Operations on Geometric Entities," *Comput. Aided Design* **12,** No. 5, 219–220 (1980).
9. A. A. G. Requicha and R. B. Tilove, "Mathematical Foundations of Constructive Solid Geometry: General Topology of Closed Regular Sets," *Technical Memo 27a*, Production Automation Project, University of Rochester, Rochester, NY, 1978.
10. R. T. Farouki and V. T. Rajan, "Imprecise Geometric Computation," presented at the *Surfaces in Computer-Aided Geometric Design Conference*, Oberwolfach, West Germany, 1987.
11. H. G. Timmer, "A Solution to the Surface Intersection Problem," *Report No. MDC J7789*, McDonnell-Douglas Corporation, Long Beach, CA, 1977.
12. E. G. Houghton, R. F. Emnett, J. D. Factor, and L. Sabharwal, "Implementation of a Divide-and-Conquer Method for Intersection of Parametric Surfaces," *Comput. Aided Geom. Design* **2,** 173–183 (1985).
13. G. E. Collins, "Infallible Calculation of Polynomial Zeros to Specified Precision," *Mathematical Software III*, J. R. Rice, Ed., Academic Press, Inc., New York, 1977.
14. J. H. Wilkinson, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1963.
15. G. Dahlquist and A. Bjorck [trans. N. Anderson], *Numerical Methods*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1974.
16. J. V. Uspensky, *Theory of Equations*, McGraw-Hill Book Co., Inc., New York, 1948.
17. J. M. Lane and R. F. Riesenfeld, "Bounds on a Polynomial," *BIT* **21,** No. 1, 112–117 (1981).
18. A. S. Householder, *The Numerical Treatment of a Single Nonlinear Equation*, McGraw-Hill Book Co., Inc., New York, 1970.
19. G. Salmon, *Lessons Introductory to the Modern Higher Algebra* (reprint), Chelsea Publishing Co., New York, 1885.
20. L. H. Williams, "Algebra of Polynomials in Several Variables for a Digital Computer," *J. ACM* **9,** 29–40 (1962).
21. J. Moses, "Solution of Systems of Polynomial Equations by Elimination," *Commun. ACM* **9,** No. 8, 634–637 (1966).
22. G. E. Collins, "Subresultants and Reduced Polynomial Remainder Sequences," *J. ACM* **14,** No. 1, 128–142 (1967).
23. S. Y. Ku and R. J. Adler, "Computing Polynomial Resultants: Bezout's Determinant vs. Collins' Reduced P.R.S. Algorithm," *Commun. ACM* **12,** No. 1, 23–30 (1969).
24. G. E. Collins, "The Calculation of Multivariate Polynomial Resultants," *J. ACM* **18,** No. 4, 515–522 (1971).
25. J. T. Kajiya, "Ray Tracing Parametric Patches," *ACM Comput. Graph.* (Proc. SIGGRAPH '82) **16,** No. 3, 245–254 (1982).
26. T. W. Sederberg, D. C. Anderson, and R. N. Goldman, "Implicit Representation of Parametric Curves and Surfaces," *Comput. Vision, Graph. & Image Proc.* **28,** 72–84 (1984).
27. Y. de Montaudouin and W. Tiller, "The Cayley Method in Computer Aided Geometric Design," *Comput. Aided Geom. Design* **1,** 309–326 (1984).
28. R. T. Farouki, "The Characterization of Parametric Surface Sections," *Comput. Vision, Graph. & Image Proc.* **33,** 209–236 (1986).
29. R. T. Farouki and J. K. Hinds, "A Hierarchy of Geometric Forms," *IEEE Comput. Graph. & Appl.* **5,** No. 5, 51–78 (1985).
30. T. W. Sederberg, "Implicit and Parametric Curves and Surfaces for Computer Aided Geometric Design," Ph.D. thesis, Purdue University, West Lafayette, IN, 1983.
31. R. T. Farouki, "Exact Offset Procedures for Simple Solids," *Comput. Aided Geom. Design* **2,** 257–279 (1985).
32. P. Hanrahan, "Ray Tracing Algebraic Surfaces," *ACM Comput. Graph.* (Proc. SIGGRAPH '83) **17,** No. 3, 83–90 (1983).
33. B. Dimsdale, "Bicubic Patch Bounds," *Comput. & Math. with Appl.* **3,** 95–104 (1977).
34. D. S. Arnon, "Topologically Reliable Display of Algebraic Curves," *ACM Comput. Graph.* (Proc. SIGGRAPH '83) **17,** No. 3, 219–227 (1983).
35. P. Frost, *An Elementary Treatise on Curve Tracing* (reprint), Chelsea Publishing Co., New York, 1872.
36. I. Petrowsky, "On the Topology of Real Plane Algebraic Curves," *Ann. Math.* **39,** No. 1, 189–209 (1938).
37. G. Salmon, *A Treatise on the Higher Plane Curves* (reprint), Chelsea Publishing Co., New York, 1879.
38. J. G. Semple and G. T. Kneebone, *Algebraic Curves*, Oxford University Press, Oxford, England, 1959.
39. R. J. Walker, *Algebraic Curves* (reprint), Springer-Verlag, New York, 1978.
40. Y. de Montaudouin, W. Tiller, and H. Vold, "Applications of Power Series in Computational Geometry," *Comput. Aided Design* **18,** No. 10, 514–524 (1986).
41. H. Engels, *Numerical Quadrature and Cubature*, Academic Press, Inc., New York, 1980.
42. A. H. Stroud, *Approximate Calculation of Multiple Integrals*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1971.
43. R. E. Barnhill and F. F. Little, "Adaptive Triangular Cubature," *Rocky Mount. J. Math.* **14,** No. 1, 53–75 (1984).
44. M. S. Shephard, "Automatic and Adaptive Mesh Generation," *IEEE Trans. Magnetics* **MAG-21,** No. 6, 2484–2489 (1985).
45. M. A. Yerry and M. S. Shephard, "Finite Element Mesh Generation Based on a Modified-Quadtree Approach," *IEEE Comput. Graph. & Appl.* **3,** No. 1, 36–46 (1983).
46. D. S. Arnon, G. E. Collins, and S. McCallum, "Cylindrical Algebraic Decomposition I: The Basic Algorithm," and "II: An Adjacency Algorithm for the Plane," *SIAM J. Computing* **13,** No. 4, 865–889 (1984).
47. H. G. Timmer and J. M. Stern, "Computation of Global Geometric Properties of Solid Objects," *Comput. Aided Design* **12,** No. 6, 301–304 (1980).
48. K. J. Weiler, "Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments," *IEEE Comput. Graph. & Appl.* **5,** No. 1, 21–40 (1985).
49. 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," and "II. A Family of Algorithms Based on Representation Conversion and Cellular Approximation," *Commun. ACM* **25,** No. 9, 635–650 (1982).
50. A. M. Messner and G. Q. Taylor, "Algorithm 550: Solid Polyhedron Measures [Z]," *ACM Trans. Math. Software* **6,** No. 1, 121–130 (1980).
51. S. Lien and J. T. Kajiya, "A Symbolic Method for Calculating the Integral Properties of Arbitrary Nonconvex Polyhedra," *IEEE Comput. Graph. & Appl.* **4,** No. 10, 35–41 (1984).
52. L. D. Landau and E. M. Lifshitz, *Mechanics*, Pergamon Press, Oxford, England, 1960.

Received August 25, 1986; accepted for publication December 22, 1986

Rida T. Farouki *IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Farouki graduated from Oxford University, England, in 1978 with First Class Honours in engineering science and was awarded the Maurice Lubbock Memorial Prize. In 1983 he received a Ph.D. in astronomy and space sciences from Cornell University, Ithaca, New York. His doctoral research was concerned with the dynamics of galaxy collisions and mergers. From 1983 to 1986 he was engaged in the design and implementation of geometric modeling algorithms at the General Electric Research and Development Center. His current research interests focus on the computational and representational aspects of boundary evaluation in solid modeling.