The functions $A(t)$ and $B(t)$ are given by Fourier time series:

$$A_{mn}(t) = a_{mno} + \sum_{i=1}^{I} a_{mni} \cos(2\pi t/\tau) + b_{mni} \sin(2\pi t/\tau) \tag{9.10}$$

$$B_{mn}(t) = b_{mno} + \sum_{i=1}^{I} c_{mni} \cos(2\pi t/\tau) + d_{mni} \sin(2\pi t/\tau) \tag{9.11}$$

where t is time, the a_{mni}, b_{mni}, c_{mni}, and d_{mni} are arbitrary real constants, and τ the period. Any continuous periodically moving surface on the sphere may be represented by some selection of these real constants; in the cardiac application, reasonable approximations to the temporal behavior are obtained with $t \leqslant 3$. Figure 9.10 shows three stages from a moving-harmonic-surface representation of the heart in early systole. The atria, at the top, contract and pump blood into the ventricles below, after which there is a ventricular contraction.

9.3 GENERALIZED CYLINDER REPRESENTATIONS

The volume of many biological and manufactured objects is naturally described as the "swept volume" of a two-dimensional set moved along some three-space curve. Figure 9.11 shows a "translational sweep" wherein a solid is represented as the volume swept by a two-dimensional set when it is translated along a line. A "rotational sweep" is similarly defined by rotating the two-dimensional set around an axis. In "three-dimensional sweeps," volumes are swept. In a "general" sweep scheme, the two-dimensional set or volume is swept along an arbitrary space curve, and the set may vary parametrically along the curve [Binford 1971; Soroka and Bajcsy 1976; Soroka 1979a; 1979b; Shani 1980]. General sweeps are quite a popular representation in computer vision, where they go by the name *generalized cylinders* (sometimes "generalized cones").

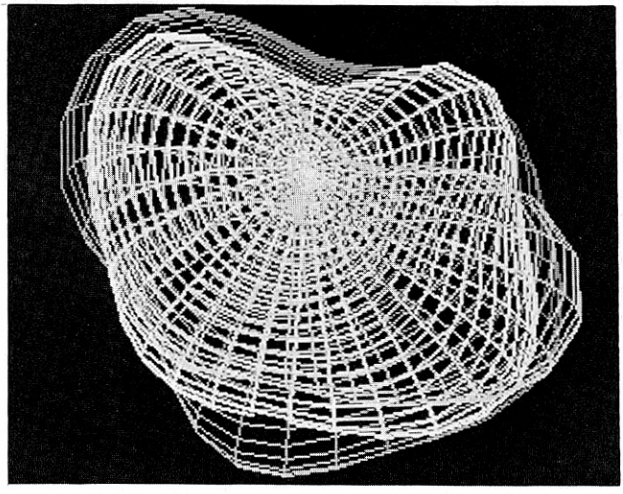


Fig. 9.10 Three stages from a moving harmonic surface (*see text and color insert*).

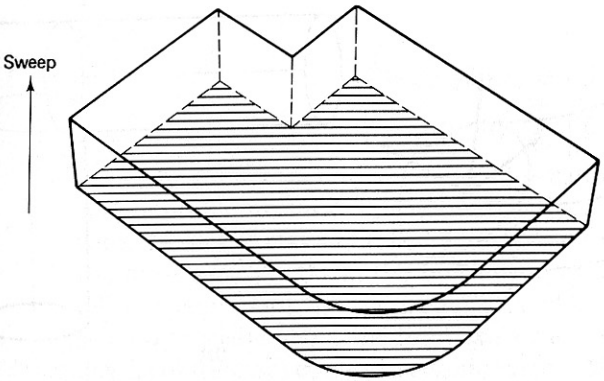


Fig. 9.11 A translational sweep.

A generalized cylinder (GC) is a solid whose axis is a 3-D space curve (Fig. 9.12a). At any point on the axis a closed cross section is defined. A usual restriction is that the axis be normal to the cross section. Usually it is easiest to think of an axis space curve and a cross section point set function, both parameterized by arc length along the axis curve. For any solid, there are infinitely many pairs of axis and cross section functions that can define it.

Generalized cylinders present certain technical subtleties in their definition. For instance, can it be determined whether any two cross sections intersect, as they would if the axis of a circular cylinder were sharply bent (Fig. 9.12b)? If the solid is defined as the volume swept by the cross section, there is no conceptual or computational problem. A problem might occur when computing the surface of such an object. If the surface is expressed in terms of the axis and cross-section functions (as below), the domain of objects must be limited so that the boundary formula indeed gives only points on the boundary.

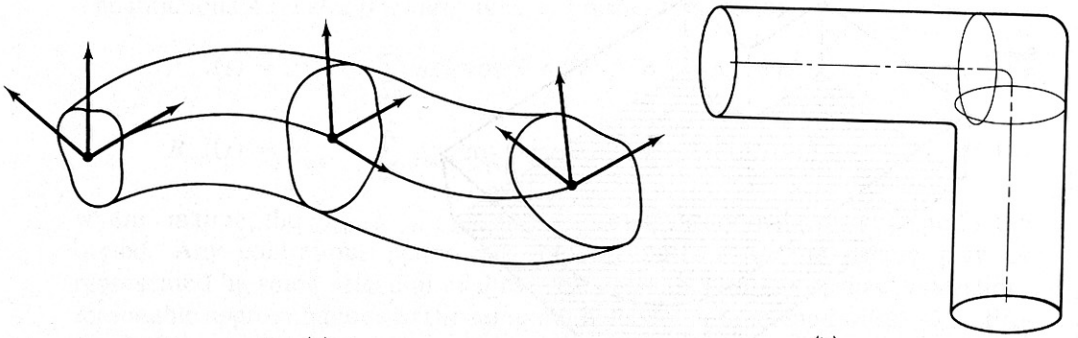
Generalized cylinders are intuitive and appealing. Let us grant that "pathological" cases are barred, so that relatively simple mathematics is adequate for representing them. There are still technical decisions to make about the representation. The axis curve presents no difficulties, but a usable representation for the cross-section set is often not so straightforward. The main problem is to choose a usable coordinate system in which to express the cross section.

9.3.1 Generalized Cylinder Coordinate Systems and Properties

Two mathematical functions defining axis and cross section for each point define a unique solid with the "sweeping" semantics described above. In a fixed Cartesian coordinate system x, y, z, the axis may be represented parametrically as a function of arc length s:

$$\mathbf{a}(s) = (x(s), y(s), z(s)) \tag{9.12}$$

It is convenient to have a local coordinate system defined with origin at each point of $\mathbf{a}(s)$. It is in this coordinate system that the cross section is defined. This system may change in orientation as the axis winds through space, or it may be most natural for it not to be tied to the local behavior of the axis. For instance, imagine tying a knot in a solid rubber bar of square cross section. The cross section



(a) (b)

Fig. 9.12 (a) A generalized cylinder and some cross-sectional coordinate systems. (b) A possibly "pathological" situation. Cross sections may be simply described as circles centered on the axis, but then their intersection makes volume calculations (for instance) less straightforward.

will stay approximately a square, and (this is the point) will remain approximately fixed in a coordinate system that twists and turns through space with the axis of the bar. On the other hand, imagine bolt threads. They can be described by a single cross section that stays fixed in a coordinate system that rotates as it moves along the straight axis of the bolt. There is no a priori reason to suppose that such a useful local coordinate system should twist along the GC axis.

A coordinate system that mirrors the local behavior of the GC axis space curve is the "Frenet frame," defined at each point on the GC axis. This frame provides much information about the GC-axis behavior. The GC axis point forms the origin, and the three orthogonal directions are given by the vectors $(\boldsymbol{\xi}, \boldsymbol{\nu}, \boldsymbol{\zeta})$, where

$\boldsymbol{\xi}$ = unit vector tangent axis

$\boldsymbol{\nu}$ = unit vector direction of center of curvature of axis normal curve

$\boldsymbol{\zeta}$ = unit vector direction of center of torsion of axis

Consider the curve to be produced by a point moving at constant speed through space; the distance the point travels is the parameter of the space curve [O'Neill 1966]. Since ξ is of constant length, its derivative measures the way the GC axis turns in space. Its derivative ξ' is orthogonal to ξ and the length of ξ' measures the curvature κ of the axis at that point. The unit vector in the direction of ξ' is ν. Where the curvature is not zero, a binormal vector ζ orthogonal to ξ and ν is defined. This binormal ζ is used to define the torsion τ of the curve. The vectors ξ, ν, ζ obey Frenet's formulae:

$$\begin{aligned} \boldsymbol{\xi}' &= \kappa\boldsymbol{\nu} \\ \boldsymbol{\nu}' &= -\kappa\boldsymbol{\xi} + \tau\boldsymbol{\zeta} \\ \boldsymbol{\zeta}' &= -\tau\boldsymbol{\nu} \end{aligned} \tag{9.13}$$

where

$$\kappa = \text{curvature} = -\boldsymbol{\nu}' \cdot \boldsymbol{\xi} = \boldsymbol{\nu} \cdot \boldsymbol{\xi}' \tag{9.14}$$

$$\tau = \text{torsion} = \boldsymbol{\nu}' \cdot \boldsymbol{\zeta} = -\boldsymbol{\nu} \cdot \boldsymbol{\zeta}' \tag{9.15}$$

The Frenet frame gives good information about the axis of the GC, but it has certain problems. First, it is not well defined when the curvature of the GC axis is zero. Second, it may not reflect known underlying physical principles that generate the cross sections (as in the bolt thread example). A solution, adopted in [Agin 1972, Shani 1980], is to introduce an additional parameter that allows the cross section to rotate about the local axis by an arbitrary amount. With this additional degree of freedom comes an additional problem: How are successive cross sections registered? Figure 9.13 shows two solutions in addition to the Frenet frame solution.

The cross sectional curve is usually defined to be in the $\boldsymbol{\nu}$–$\boldsymbol{\zeta}$ plane, normal to $\boldsymbol{\xi}$, the local GC axis direction. The cross section may be described as a point set in this plane, using inequalities expressed in the $\boldsymbol{\nu}$–$\boldsymbol{\zeta}$ coordinate system. The cross section boundary (outline curve) may be used instead, parameterized by another parameter r. Let this curve be given by

$$\text{cross section boundary} = (x(r, s),\ y(r, s))$$

The dependence on s reflects the fact that the cross section shape may vary along the GC axis. The expression above is in world coordinates, but should be moved to

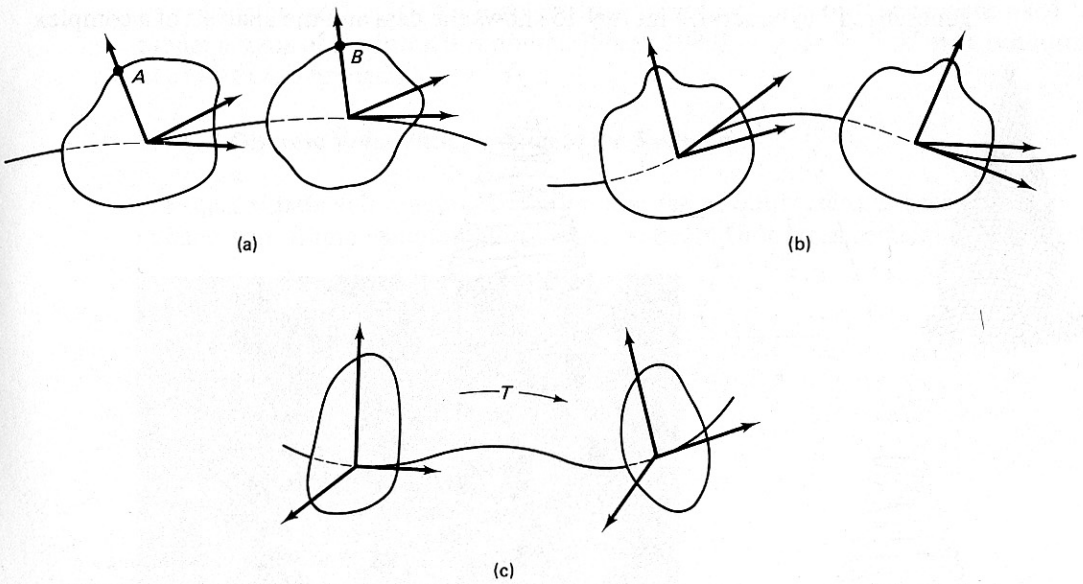


Fig. 9.13 (a) Local coordinates are the Frenet frame. Points A and B must correspond. (b) Local coordinates are determined by the cross sectional shape. (c) Local coordinates are determined by a heuristic transformation from world coordinates.

the local coordinates on the GC axis. A transformation of coordinates allows the GC boundary to be expressed (if the GC is well behaved) as

$$B(r, s) = \mathbf{a}(s) + x(r, s)\,\boldsymbol{\nu}(s) + y(r, s)\,\boldsymbol{\zeta}(s) \tag{9.16}$$

One of the advantages of the generalized cylinder representation is that it allows many parameters of the solid to be easily calculated.

- In matching the GC to image data it is often necessary to search perpendicular to a cross section. This direction is given from $x(r, s)$, $y(r, s)$ by $((dy/ds)\boldsymbol{\nu}, -(dx/ds)\boldsymbol{\zeta})$.
- The area of a cross section may be calculated from Eq. (8.16).
- The volume of a GC is given by the integral of: the area as a function of the axis parameter multipled by the incremental path length of the GC axis, i.e.,

$$\text{volume} = \int_0^L \text{area}(s)\, ds$$

9.3.2 Extracting Generalized Cylinders

Early work in biological form analysis provides an example of the process of fitting a GC to real data and producing a description [Agin 1972]. One of the goals of this work was to infer the stick figure skeleton of biological forms for use in matching models also represented as skeletons. In Fig. 9.14 the process of inferring the axis from the original stripe three-dimensional data is shown; the process iterates toward a satisfactory fit, using only circular cross sections (a common constraint with "generalized" cylinders). Figure 9.15 shows the data and the analysis of a complex

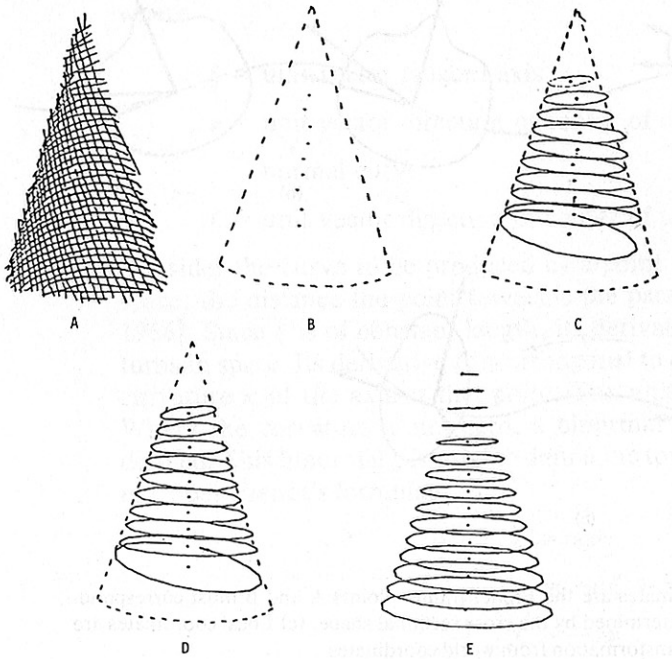


Fig. 9.14 Stages in extracting a generalized cylinder description for a circular cone. (a) Front view. (b) Initial axis estimate. (c) Preliminary center and axis estimate. (d) Cone with smoothed radius function. (e) Completed analysis.

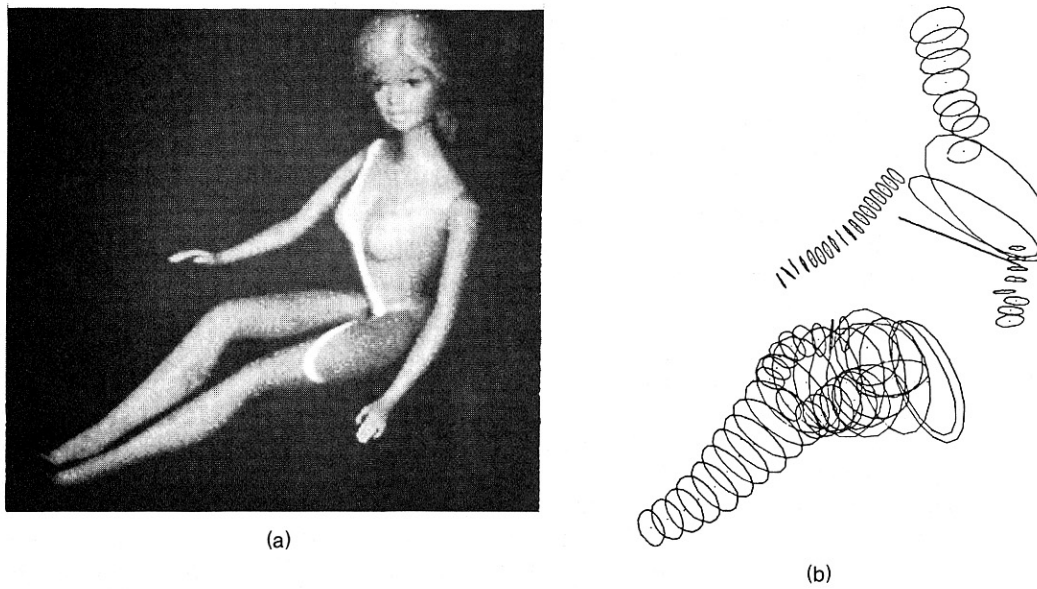


Fig. 9.15 (a) TV image of a doll. (b) Completed analysis of doll.

biological form. In real data, complexly interrelated GCs are hard to decompose into satisfactory subparts. Without that, the ability to form a satisfactory articulated skeleton is severely restricted.

In later work, GCs with spline-based axes and cross sections were used to model organs of the human abdomen [Shani 1980]. Figure 9.16 shows a rendition of a GC fit to a human kidney.

9.3.3 A Discrete Volumetric Version of the Skeleton

An approximate volume representation that can be quite useful is based on an articulated wire frame skeleton along which spheres (not cross sections) are placed.

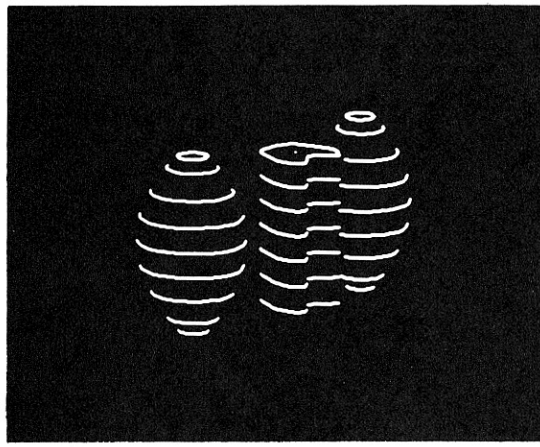


Fig. 9.16 Generalized cylinder representation of two kidneys and a spinal column. This coarse, nominal model is refined during examination of CAT data (see Fig. 9.6).

This representation has some of the flavor of an approximate sweep representation. An example of the use of such a representation and a figure are given in Section 7.3.4. This representation was originally conceived for graphics applications (the spheres look the same from any viewpoint) [Badler and Bajcsy 1978]. Collision detection is easy, and three-dimensional objects can be decomposed into spheres automatically [O'Rourke and Badler 1979]. From the spheres, the skeleton may be derived, and so may the surface of the solid. This representation is especially apt for many computer vision applications involving nonrigid bodies if strict surface and volumetric accuracy is not necessary [Badler and O'Rourke 1979].

9.4 VOLUMETRIC REPRESENTATIONS

Most world objects are solids, although usually only their surfaces are visible. A representation of the objects in terms of more primitive solids is often useful and can have pleasant properties of terseness, validity, and sometimes ease of computation. The representations given here are presented in order of increasing generality; constructive solid geometry includes cell decomposition, which in turn includes spatial occupancy arrays.

Algorithms for processing volume-based representations are often of a different flavor than surface-based algorithms. We give some examples in Section 9.4.4. Objects represented volumetrically can be depicted on raster graphics devices by a "ray-casting" approach in which a line of sight is constructed through the viewing plane for a set of raster points. The surface of the solid at its intersection with the line of sight determines the value of the display at the raster point. Ray casting can produce hidden-line and shaded displays; graphics is only one of its applications (Section 9.4.4).

9.4.1 Spatial Occupancy

Figure 9.17 shows that three-dimensional spatial occupancy representations are the three-dimensional equivalent of the two-dimensional spatial occupancy representations of Chapter 8. Volumes are represented as a three-dimensional array of cells which may be marked as filled with matter or not. Spatial occupancy arrays can require much storage if resolution is high, since space requirements increase as the cube of linear resolution. In low-resolution work with irregular objects, such as arise in computer-aided tomography, spatial occupancy arrays are very common. It is sometimes useful to convert an exact representation into an approximate spatial occupancy representation. Slices or sections through objects may be easily produced. The spatial occupancy array may be run-length encoded (in one dimension), or coded as blocks of different sizes; such schemes are actually cell-decomposition schemes (Section 9.4.2).

With the declining cost of computer memory, explicit spatial occupancy arrays may become increasingly common. The improvement of hardware facilities for parallel computation will encourage the development of parallel algorithms to compute properties of solids from these representations.

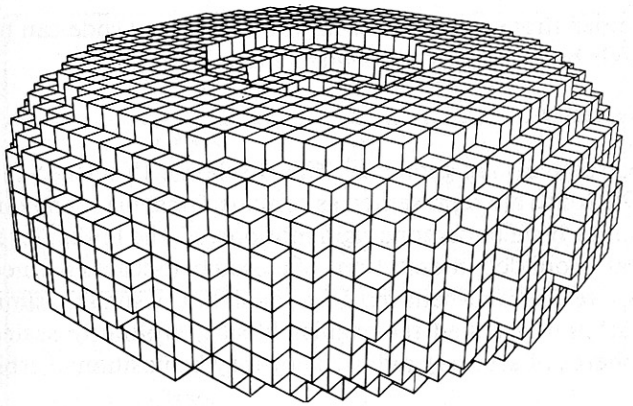


Fig. 9.17 A solid (the shape of a human red blood cell) approximated by a volume occupancy array.

9.4.2 Cell Decomposition

In cell decomposition, cells are more complex in shape but still "quasi-disjoint" (do not share volumes), so the only combining operation is "glue" (Fig. 9.18). Cells are usually restricted to have no holes (they are "simply connected"). Cell decompositions are not particularly concise; their construction (especially for curved cells) is best left to programs. It seems difficult to convert other representations exactly into cell decompositions. Two useful cell decompositions are the "oct-tree" [Jackins and Tanimoto 1980] and the kd-tree [Bentley 1975]. They both can be produced by recursive subdivision of volume; these schemes are the three-dimensional analogs of pyramid data structures for two dimensional binary images.

The quasi-disjointness of cell-decomposition and spatial-occupancy primitives may be helpful in some algorithms. Mass properties (Section 9.4.4) may be computed on the components and summed. It is possible to tell whether a solid is connected and whether it has voids. Inhomogeneous objects (such as human anatomy inside the thorax) can be represented easily with cell decomposition and spa-

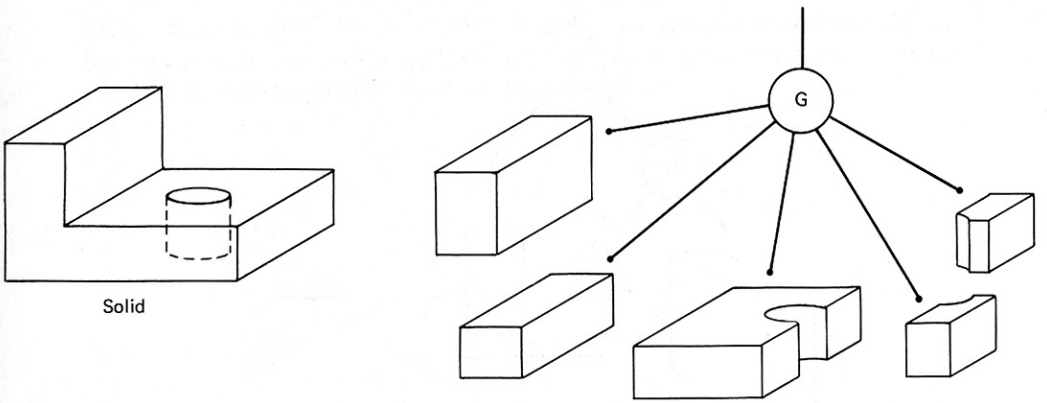


Fig. 9.18 A volume and its cell decomposition.

tial occupancy. The CT number (transparency to x-rays) or a material code can be kept in a cell instead of a single bit indication of "solid or space."

9.4.3 Constructive Solid Geometry

Figure 9.19 shows one constructive solid geometry (CSG) scheme [Voelcker and Requicha 1977; Boyse 1979]. Solids are represented as compositions, via set operations, of other solids which may have undergone rigid motions. At the lowest level are primitive solids, which are bounded intersections of closed half-spaces defined by some $F(x, y, z) \geqslant 0$, where F is well-behaved (e.g., analytic). Usually, primitives are entities such as arbitrarily scaled rectangular blocks, arbitrarily scaled cylinders and cones, and spheres of arbitrary radius. They may be positioned arbitrarily in space.

Figure 9.20 shows a parameterized representation [Marr and Nishihara 1978; Nishihara 1979] based on shapes (here cylinders) that might be extracted from an image.

A CSG representation is an expression involving primitive solid and set operators for combination and motion.

<CSG*Rep*> ::= <primitive solid> |
MOVE <CSG Rep> BY <Motion Params> |
<CSG Rep> <Combine Op> <CSG Rep>

The combining operators are best taken to be *regularized* versions of set union, intersection, and difference (the complement is a possible operator, but it allows unbounded solids from bounded primitives).

Regularity is a fundamental property of any set of points that models a solid. In a given space, a set X is regular if $X = kiX$, where k and i denote the *closure* and *interior* operators. Intuitively, a regular set has no isolated or dangling boundary points. The regularization r of a set X is defined by $rX = kiX$. Regularization informally amounts to taking what is inside a set and covering that with a tight skin. Regular sets are not closed under conventional set operations, but *regularized*

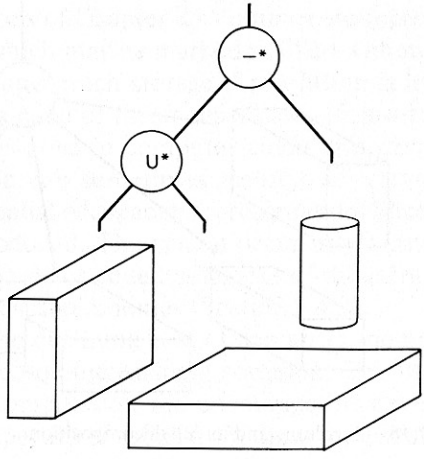


Fig. 9.19 Constructive solid geometry for the volume of Fig. 9.18.

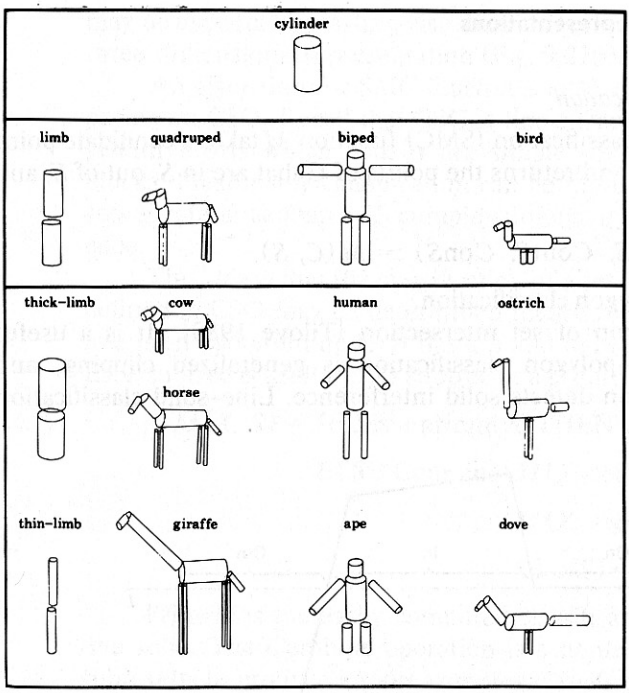


Fig. 9.20 A parameterized constructive representation for animal shapes.

operators do preserve regularity. Regularized operators are defined by

$$X \ <\text{OP}>^* \ Y = r(X \ <\text{OP}> \ Y)$$

Regularity and regularized set operators provide a natural formalization of the dimension-preserving property exhibited by many geometric algorithms, thus obviating the need to enumerate many annoying "special cases." Figure 9.21 illustrates conventional versus regularized intersection of two sets that are regular in the plane.

If the primitives are unbounded, checking for boundedness of an object can be difficult. If they are bounded, any CSG representation is a valid volume representation. CSG can be inefficient for some geometric applications, such as a line drawing display. (Converting the CSG representation to a boundary representation is the one way to proceed; see Section 9.4.4.)

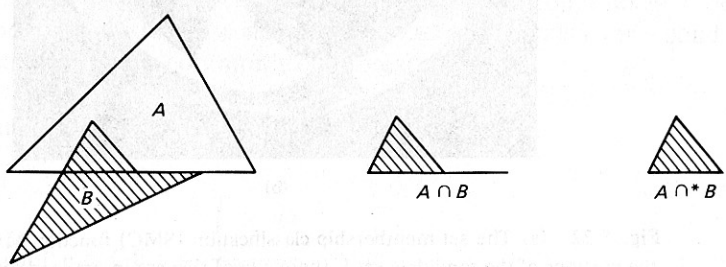


Fig. 9.21 Conventional ($\cap$) and regularized ($\cap^*$) polygon intersection.

9.4.4 Algorithms for Solid Representations

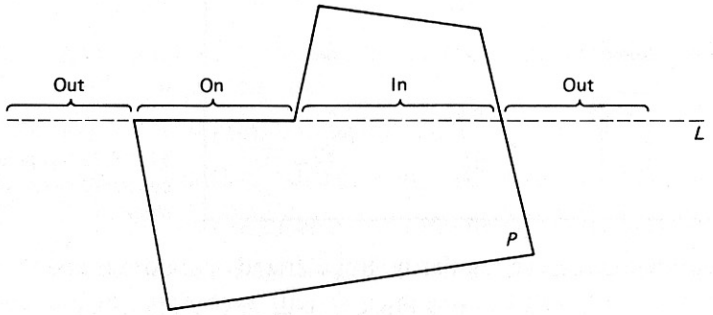
Set Membership Classification

The set membership classification (SMC) function M takes a candidate point set C and a reference set S, and returns the points of C that are in S, out of S, and on the boundary of S.

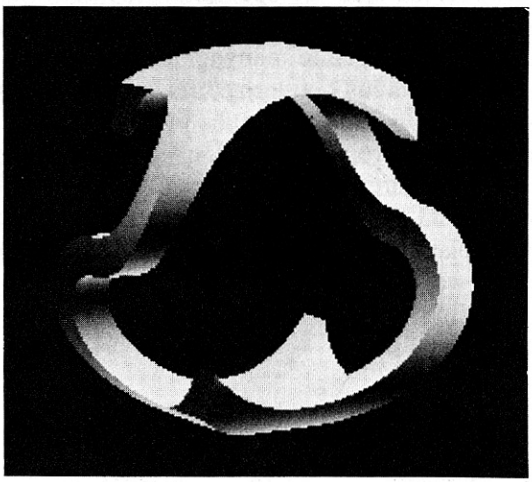
$$(C\text{in}S,\ C\text{out}S,\ C\text{on}S) := M(C, S)$$

Figure 9.22a shows line–polygon classification.

SMC is a generalization of set intersection [Tilove 1980]. It is a useful geometric utility; polygon–polygon classification is generalized clipping, and volume–volume classification detects solid interference. Line–solid classification



(a)



(b)

Fig. 9.22 (a) The set membership classification (SMC) function M(L, P) finds the portions of the candidate set L (here a line) that are in, on, and out of a reference set (here a polygon) P. (b) Image produced by ray casting, a special case of SMC.

may be used for ray casting visualization techniques to generate images of a known three-dimensional representation (Fig. 9.22b).

An algorithm for SMC illustrates a "divide and conquer" approach to computing on CSG. Recall that CSG is like a tree of set operations, whose leaves are primitive sets which usually are simple solids such as cylinders, spheres, and blocks. Presumably classification can be more easily computed with these simple sets as reference than with complex unions, intersections, and differences as reference.

The idea is that the classification of a set C with respect to a complex object S defined in CSG may be determined recursively. Any internal node S in the CSG tree is an operation node. It has left and right arguments and an operation OpofS. Each subtree is itself a CSG subtree or a primitive.

$$M(X, S) = \text{IF } S\text{is a primitive THEN prim}-M(X, S)$$
$$\text{ELSE Combine}(M(X, \text{ left}-\text{subtree}(S)),$$
$$M(X, \text{ right}-\text{subtree}(S)),$$
$$\text{OPof}S);$$

Prim-M is the easily computed classification with respect to a simple primitive solid. The Combine operation is a nontrivial calculation that combines the subresults to produce a more complex classification. It is illustrated in two dimensions for line classification in Fig. 9.23. Having classified the line L against the polygon $P1$ and $P2$, the classifications can be combined to produce the classification for $P1 \cap P2$. Precise rules for combine may be written for (regularized) union, intersection, and set difference. An important point is that when a point is in the "on" set of S_1 and in the "on" set of S_2, the result of the combination depends on extra information. In Fig. 9.23, segments X and Y both result from this ON–ON case of combine, but segment X is OUT of the boundary of the intersection and Y is IN the intersection. The ambiguity must be resolved by keeping "neighborhood information" (local geometry) attached to point sets, and combining the neighborhoods along with the classifications. The technical problems surrounding combine can be solved, and SMC is basic in several solid geometric modeling systems [Boyse 1979; Voelcker et al. 1978; Brown et al. 1978].

Mass Properties

The analog of many two-dimensional geometric properties is to be found in "mass properties," which are defined by volume integrals over a solid. The four types of mass properties commonly of interest are:

$$\text{Volume: } V = \int_S du$$

$$\text{Centroid: e.g. } GC_x = \frac{\int_S x \, du}{V}$$

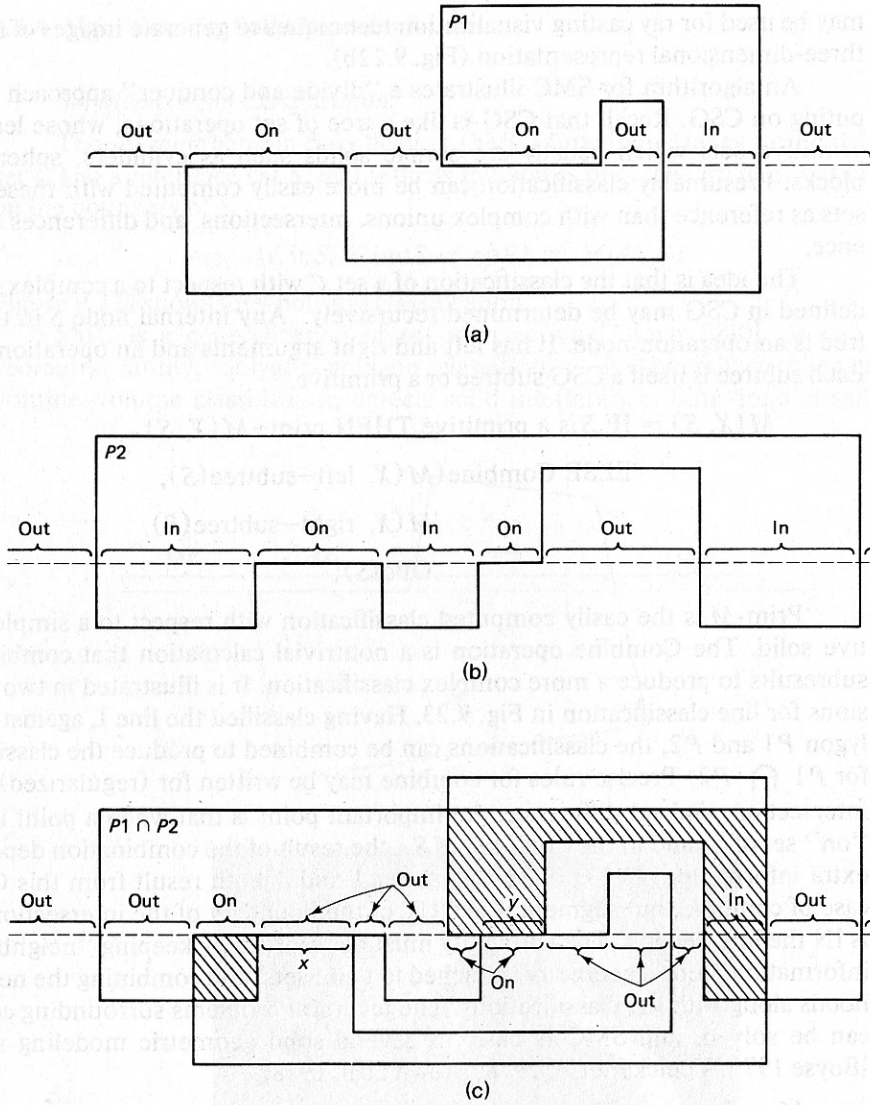


Fig. 9.23 Combining line-polygon classifications (a) and (b) must produce the classification (c).

Moment of Inertia: e.g. $I_{xx} = m \int_s (y^2 + z^2)\, du$ (9.17)

Product of Inertia: e.g. $P_{xy} = m \int_s xy\, du$

where m is a density measure, du the volume differential, and integrals are taken over the volume.

Measures such as these are not necessarily easy to compute from a given representation. The calculation of mass properties of solids from various representations is discussed in [Lee and Requicha 1980]. The approaches suggested by the representations are shown in Fig. 9.24.

One method is based on decomposing the solid into quasi-disjoint cells. An integral property of the cell decomposition is just the sum of the property for each of the cells. Hence if computing the property for the cells is easy, the calculation is easy for the whole volume. One is invited to decompose the body into simple cells, such as columns or cubes, as shown in Fig. 9.25. The resulting calculations, performed to reasonable error bounds on fairly complex volumes, take unacceptably long for the pure spatial occupancy enumeration, but are acceptable for the column and block decompositions. (The column decomposition corresponds to a ray casting approach.) The block decomposition method can be programmed using octtrees or kd-trees in a manner reminiscent of the Warnock hidden-line algorithm [Warnock 1969], in which the blocks are found automatically, and their size diminishes as increased resolution is needed in the solid. In calculating from a constructive solid geometry representation, the same divide-and-conquer strategy that is useful for SMC may be applied. Again, it recursively solves subproblems induced by the set operators (Fig. 9.26). The strategy is less appealing here since the number of subproblems can grow exponentially in the worst case.

In boundary representations, one can perhaps directly integrate over the boundary in a three-dimensional version of the polygon area calculation given in Chapter 8. This method is often impossible for curved surfaces, which, however, may be approximated by planar faces. An alternative is to use the divergence

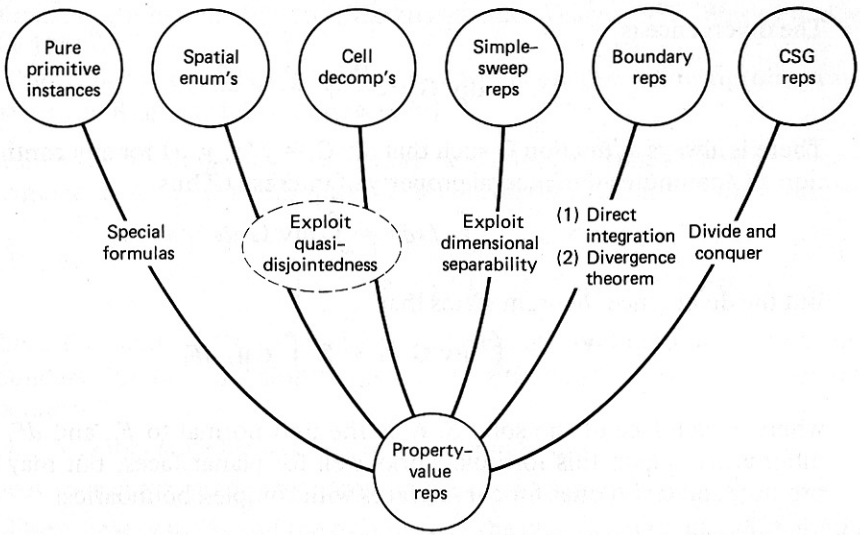


Fig. 9.24 "Natural" approaches to computing mass properties from several representations.

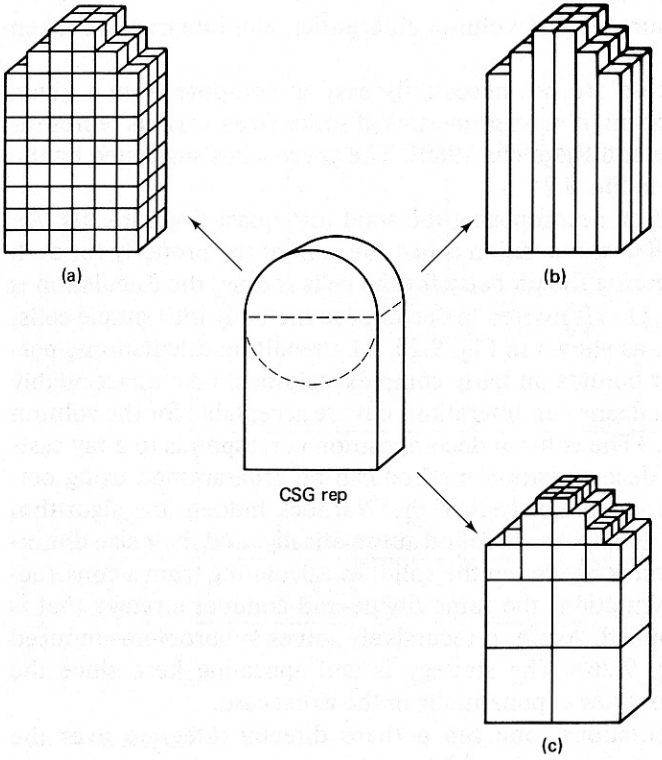


Fig. 9.25 Cell decompositions for mass properties.

theorem (Gauss's theorem). The *divergence* is a scalar quantity defined at any point in a vector field by writing the vector function as

$$G(x, y, z) = P(x, y, z)\mathbf{i} + Q(x, y, z)\mathbf{j} + R(x, y, z)\mathbf{k}. \tag{9.18}$$

The divergence is

$$\text{div } \mathbf{G} = \frac{P}{x} + \frac{Q}{y} + \frac{R}{z} \tag{9.19}$$

There is always a function $\mathbf{G}$ such that div $\mathbf{G} = f(x, y, z)$ for any continuous function f (f computes the integral property of interest.) Thus

$$\int_S f \, dv = \int_S \text{div } \mathbf{G} \, dv \tag{9.20}$$

But the divergence theorem states that

$$\int_S \text{div } \mathbf{G} \, dv = \sum_i \int_{F_i} \mathbf{G}\mathbf{n}_i \, dF_i \tag{9.21}$$

where F_i is a face of the solid S, $\mathbf{n}_i$ is the unit normal to F_i, and dF_i the surface differential. Again this formula works well for planar faces, but may require approximation techniques for curved faces with complex boundaries.

Boundary Evaluation

The calculation of a face-based surface (boundary) representation from a

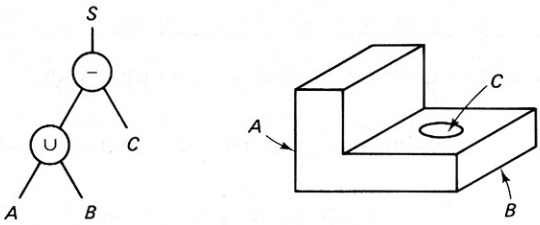
• Divide and conquer

Reduction formula

$$\int_{A \cup B} = \int_A + \int_B - \int_{A \cap B}$$

$$\int_{A-B} = \int_A - \int_{A \cap B}$$

Example



$$I_S = I_A + I_B - I_{A \cap B} - \underbrace{I_{A \cap C}}_{\emptyset} - I_{B \cap C} + \underbrace{I_{A \cap B \cap C}}_{\emptyset}$$

Fig. 9.26 Recursive problem decomposition for mass property calculation.

CSG representation is called *boundary evaluation.* It is an example of *representation conversion.* Both the CSG and boundary are usually unambiguous representations of a volume; a CSG expression (a solid) has just one boundary, but a boundary (representing a solid) usually has many CSG expressions. Since a solid may be put together from primitives in many ways, the mapping back from boundary to CSG is not usually attempted (but see [Markovsky and Wesley 1980, Wesley and Markovsky 1981]).

One style of boundary evaluation is based on the following observations [Voelcker and Requicha 1980; Boyse 1979].

- Boundaries of composite objects may be computed from certain set-theoretic formulae. For (regularized) intersection of two objects S and T, the formula is

$$b(S \cap^* T) = (bS \cap^* iT) \cup^* (iS \cap^* bT) \cup^* (bS \cap^* bT \cap^* ki(S \cap^* T)) \tag{9.22}$$

  where $\cap^*$ and $\cup^*$ are regularized intersection and union: b, i, and k are the boundary, interior, and closure operators. (Recall that ki is r, the regularization operator).
- Faces of composite objects can arise only from faces of primitives.
- Faces are either bounded by edges or are self-closing (as is the sphere).

These observations and the existence of the classification operation motivate the grand strategy that follows (ignoring several important details and concentrating on the core of the algorithm.)

1. Find all possible ("tentative") edges for each face of each primitive in the composite.
2. Classify each tentative edge with respect to the composite solid.
3. The ON portions of those edges must be enough to define the boundary.

Given the grand strategy, several algorithms of varying sophistication are possible, depending on what edges should be classified (how to generate tentative edges), in what order they should be classified, and how classification is done. The following algorithm is very simple (but very inefficient); useful algorithms are rather more complex.

Algorithm 9.1: CSG to Boundary Conversion (top-level control loop)

Input: Solid defined by CSG expression of regularized set operations applied to primitive solids.

Output: "Bfaces" in the object boundary. Bfaces are represented by their bounding edges. They may have little relation to the "intuitive faces" of the boundary; they may overlap each other, and a Bface may be disconnected (specify more than one region). Edges may appear many times. The Bface-oriented boundary may be processed to remove repetition and merge Bfaces into more intuitively appealing boundary faces.

BEGIN

Form a list PFaces of all ("intuitive") faces of primitive solids involved in the CSG expression, and an initially empty list BFaces to hold the output faces.

For every PFace $F1$ in PFaces:

Create a B-Face called ThisBFace, initially with no edges in it.

For every PFace $F2$ after $F1$ in the PFaces list (this generates all distinct pairs of PFaces just once):

Intersect $F1$ and $F2$ to get TEdges, a set of edges tentatively on the boundary of the solid. If $F1$ and $F2$ do not intersect or intersect only in a point, TEdges is empty. If they intersect in a line, TEdges is the single resulting edge. If they intersect in a two-dimensional region, TEdges contains the bounding edges of the intersection region.

Classify every TEdge in TEdges with respect to the whole solid (the CSG expression). Put TEdges that are ON the solid boundary into ThisBFace.

If ThisBFace is not empty, put it into BFaces.

End Inner Loop

End Outer Loop

END

Algorithms such as this involve many technical issues, such as merging coplanar faces, stitching edges together into faces, regularization of faces, removing multiple versions of edges. Boundary evaluation is inherently rather complex, and depends on such things as the definition and representation of faces as well as the geometric utilities taken as basic [Voelcker and Requicha 1981]. Boundary evaluation is an example of exact conversion between significantly different representations. Such conversions are useful, since no single representation seems convenient for all geometric calculations.

9.5 UNDERSTANDING LINE DRAWINGS

"Engineering" line drawings have been (and to a great extent are still) the main medium of communication between human beings about quantitative aspects of three-dimensional objects. The line drawings of this section are only those which are meant to represent a simple domain of polyhedral or simply curved objects. Interpretation of "naturalistic" drawings (such as a sketchmap [Mackworth 1977]) is another matter altogether.

Line drawings (even in a restricted domain) are often ambiguous; interpreting them sometimes takes knowledge of everyday physics, and can require training. Such informed interpretation means that even drawings that are strictly nonsense can be understood and interpreted as they were meant. Missing lines in drawings of polyhedra are often so easy to supply as to pass unnoticed, or be "automatically supplied" by our model-driven perception.

Generalizing the line drawing to three dimensions as a list of lines or points is not enough to make an unambiguous representation, as is shown by Fig. 9.27,

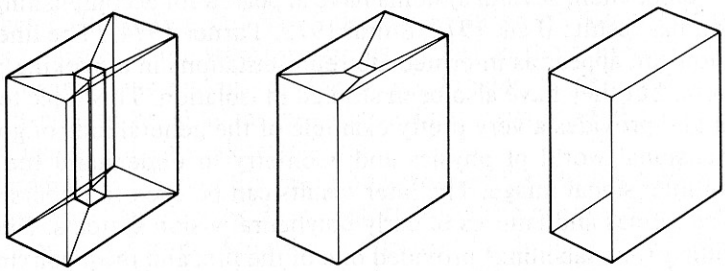


Fig. 9.27 An ambiguous (wireframe) representations of a solid with two of three possible interpretations.