An Algorithm and Data Structure for 3D Object Synthesis Using Surface Patch Intersections

Wayne E. Carlson

The Ohio State University
Computer Graphics Research Group
Columbus, Ohio

ABSTRACT

There are several successful systems that provide algorithms that allow for the intersection of polygonal objects or other primitive shapes to create more complex objects. Our intent is to provide similar algorithms for intersecting surface patches. There have been contributions to this concept at the display algorithm level, that is, computing the intersection at the time the frame is generated. In an animation environment, however, it becomes important to incorporate the intersection in the data generation routines, in order that those parts of the intersected object that never contribute to an image are not processed by the display algorithm. This only increases the complexity of the object unnecessarily, and subsequently puts an additional burden on the display algorithms.

An algorithm is described which uses a modified Catmull recursive subdivision scheme to find the space curve which is the intersection of two bicubic patches. An associated data structure is discussed which incorporates this curve of intersection in the patch description in a way suitable for efficient display of the intersected object. Sample output of these intersections are shown which serve to illustrate the capabilities and limitations of the described procedures.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - Curve, surface, solid, and object

This research was supported in part by the National Science Foundation, grant number MCS 7923670.

General Terms: combinatorial geometry, recursive subdivision, surface patch

Additional Keywords and Phrases: quadtree

1. INTRODUCTION

One of the most important aspects of complex image synthesis is the generation and description of the data comprising an object to be rendered. Many early techniques were developed to deal with this task, which might be referred to as computational geometry [8] or computer-aided geometric design. The most interesting general purpose data generation techniques involve interactively specifying and successively modifying certain primitive structures in order to create an object for computer display. These primitive structures can be points, lines, polygons, or parametric surfaces.

Some of the most complex three dimensional objects have been created in an environment of this type by using capabilities which lie in the realm of combinatorial geometry. This involves describing a three dimensional object by the combination of simpler geometric shapes or primitives. That is, certain operators (for example, union and intersection) are applied to two objects, resulting in a third object which is defined by the original two and the operator. Several systems have been developed [3,13,1,4] which combine two polyhedral objects by intersecting their faces in order to define a third object. Levin [12] described a similar combinatorial technique for quadric surfaces.

This paper attempts to extend the concept of combinatorial geometry to parametric (in particular, bicubic) surface patches by developing an algorithm that finds the space curve which is the intersection of two patches, and uses this information to form a third object from two objects comprised of surface patches. The algorithm proceeds by using a Catmull patch subdivision scheme to recursively subdivide two patches until the intersection, if it exists, can be isolated.

2. CATMULL SUBDIVISION OF BICUBIC PATCHES

A bicubic parametric surface patch can be defined in matrix notation as follows:

$$V(u,v) = (u^3\ u^2\ u\ 1) * B * P * B^t * (v^3\ v^2\ v\ 1)^t$$

where P is a 4x4 matrix representing the sixteen points comprising the defining polygonal network for the patch. B is the 4x4 matrix comprising the coefficients of the basis functions. In particular, if we choose to use the Bernstein polynomials as our basis, these functions are

$$f(x) = (1 - x)^3 = 1 - 3x + 3x^2 - x^3$$

$$g(x) = 3x(1 - x)^2 = 3x - 6x^2 + 3x^3$$

$$h(x) = 3x^2(1 - x) = 3x^2 - 3x^3$$

$$k(x) = x^3$$

and the corresponding matrix is

$$B = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Using this formulation, Catmull[5] showed that such a patch could be subdivided into four subpatches by finding the midpoint of the patch and the midpoint of each of the four boundary curves. By observing that the midpoint of a cubic curve is the average of its two endpoints less a correction term, he created a correction matrix C for the patch which contains the endpoints of all the boundary curves, or the corner points of the patch, and the correction values. C can be defined as

$$C = S * B * P * B^t * S^t$$

where

$$S = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 3 & 1 & 0 & 1 \end{pmatrix}$$

is derived to obtain the correction terms for each curve. The center of the patch can be obtained by utilizing the values and the correction terms of the midpoints of the boundary curves.

3. PATCH INTERSECTION

After two objects defined as a collection of bicubic patches defined with the Bernstein basis have been interactively positioned and oriented in space, the algorithm proceeds on a patch by patch basis. Each patch of the first object is compared against every patch of the second object, and the intersection is calculated.

The algorithm has three main subalgorithms. First, the subdivision of the patches is performed. Second, the intersection is calculated, and finally the new object is described according to the combinatorial operator in effect.

3.1. SUBDIVISION ALGORITHM

Since there are more (and more complex) steps as the algorithm progresses, it is desirable to determine if a patch might possibly participate in the intersection as soon as possible, and eliminate it from consideration if not. At the highest level, the test relies on the fact that the Bernstein basis possesses the convex hull property; that is, all points on the surface of the patch lie within the convex hull defined by the sixteen points of the defining polygonal network. Thus, it suffices to show that the convex hulls of the two patches under consideration are linearly separable to determine if the patches intersect. As it was discovered, the separability test for two convex hulls was more expensive than performing the more complex tests on the patch later in the algorithm, so a faster test was desired.

After determining that a simple min-max enclosing box test which used the defining polygon was inconclusive for too many cases where the patches were in fact separable, the following test was implemented. An arbitrary planar equation related to the defining polygonal network was calculated. In particular, three of the corner vertices of the network were used to obtain the planar equation. The maximum distance of each of the other network points from the plane, both above and below, were determined. This segmented the space into three parts, with the patch guaranteed to be totally within the middle segment, since the convex hull is guaranteed to be totally included in this segment. (See Figure 1) If any point in the network of the other patch was found to lie within this center segment, no determination of separability could be made. In most cases that were tried, this test drastically cut the number of patches that needed to be considered further. In many cases, there were over fifty percent fewer patches than were considered with the basic min-max enclosing box test, and this segmentation test was only slightly more expensive to perform.

The subdivision algorithm is a recursive algorithm. If two patches failed the separability test described above, then one is divided into four subpatches. The actual bicubic patch is divided according to the Catmull scheme presented in section 2. Then a new defining polygonal network for each new subpatch is calculated. This is done relatively simply by using the inverses of some of the matrices that were used in the subdivision.

If C is the matrix containing the subpatch values and correction information for the four corners, and S and B are the matrices defined in the section 2, then the matrix SP of subpolygon coordinates is given by

$$SP = B^{-1} * S^{-1} * C * (S^{t})^{-1} * (B^{t})^{-1}$$

The separability test is then made for the subpatch using this new subpolygon network and the second of the two patches. The subdivision is recursively performed on a subpatch until either its subpatches pass the separability test, or the faces of the defining polygonal networks for these subpatches are coplanar to within a given epsilon. The planarity is measured by determining the distance from each point in the defining polygonal network to the plane containing three of the corner points, and testing to see if this distance is less than epsilon. When this happens, it can be shown that the patch will approximate very closely the quadrilateral containing the four corners of the network. Hence, this quadrilateral is entered into the data structure to be considered in the intersection part of the algorithm. The second patch is then recursively subdivided, being tested against the first patch.

3.2. INTERSECTION ALGORITHM

When the subdivision algorithm completes, we have two sets of (planar) polygons, one set from each patch. The intersection algorithm is an order n-squared algorithm that compares each polygon of one set against every polygon of the second set. Once again, an essential criterion is that processing will be aborted as soon as possible if no intersection between the two polygons exists.

The following sequence of tests will guarantee that this criterion is met to an acceptable degree. First, a (min-max) enclosing box test is performed on the two polygons under consideration. If the test is successful, the two polygons are linearly separable and need not be considered any further. If it fails, then the first polygon is compared edgewise against the second. The planar equation of the second polygon is evaluated at both endpoints of an edge. If both evaluate to nonzero results of the same sign, they lie on the same side of the plane containing the quadrilateral, and hence can't intersect. If not, the edge is compared against the bounding box of the quadrilateral, and we only continue if they can't be shown to be separable.

Next, the point of intersection between the edge and the plane containing the face is calculated. Two tests are performed to determine if this point lies within the quadrilateral or not. The largest coefficient of the planar equation will determine which 2D plane will result in the 2D projection of the face having the largest area. Both the quadrilateral and the point of intersection are projected onto this 2D plane.

A two dimensional min-max test is used on the projected quadrilateral and the projected point. If the point lies outside this 2D box, it cannot be inside the three dimensional quadrilateral. If it is inside the enclosing box, a vector emanating from the projected point, and extending to infinity in some direction is considered. If it intersects the projected edges of the face an odd number of times, the point must be within the quadrilateral. (See Figure 2) These tests are done for each edge, and then the roles of the two quadrilaterals are reversed. Upon completion of this part of the algorithm, we have a collection of line segments in our data structure which are the intersections of the various quadrilaterals. Now they are organized, by using information about their relative orientation, into an ordered set that can faithfully be used as a good approximation to the intersection of the two patches.

3.3. DATA STRUCTURE

The subdivision process described above can logically be represented by a tree of degree four, or quadtree [9]. In this structure, the root of the tree represents an entire patch. The four children are the subpatches into which the patch is subdivided, and the leaf nodes are those subpatches that either are found not to participate in the intersection, or have satisfied the planarity termination condition.

Each node of the tree is stored as a six field record. Included are pointers to the parent node, the four children, and a leaf pointer. If the node is itself the root of a subtree, the leaf pointer has the value NULL. Otherwise, it points to a record containing the following information relevant to the subdivision and subsequent intersection: a pointer to the defining polygonal network for the subpatch of the tree limb; the orientation of this subpatch relative to the other object; a pointer to a record containing information relevant to the actual intersection; links to records for subpatches 'outside' and subpatches 'inside' the other object.

As the subdivision process progresses, it can determine if a subpatch absolutely does not contain the curve of intersection. If this is determined, the relative orientation of the subpatch can also be easily calculated. In this case, the orientation is entered into the proper field of the record just described, and the intersection pointer is NULL. If, on the other hand, it cannot be determined without a doubt that the patch doesn't contain the curve of intersection, a link is established to the intersection information record, and the orientation field must wait until later to be filled in.

If the link field to the intersection information record is not NULL, then the intersection algorithm considers the defined subpatch in its comparison to determine the space curve approximation. This record contains the following fields: a count of the number of points of intersection calculated and which are relevant to this particular subpatch (it is 0 if there are none); pointers to the coordinates of the intersection points; a pointer to the edge from which this point came; an orientation flag which indicates if the edge was 'entering' or 'leaving' the other subpatch (this is used to correctly recreate the new object in the combinatorial algorithm); the slopes of the subpatches at this point of intersection.

A quadtree was chosen as the internal data structure because it is relatively compact, it retains the information relative to the topological structures of the original patches and of the subpatches, and it lends itself quite well to the combinatorial operators available to the user. The implementation represents an entire object as an n-branch tree, where n is the number of patches in the description, and each branch links a quadtree, which is related to the corresponding patch.

Figure 3 shows a patch subdivided until those subpatches whose polygonal approximations possibly participate in determining the curve of intersection were determined. Figure 4 represents the associated quadtree. A black leaf node represents a non NULL node, with relevant information provided by the appropriate algorithms.

3.4. COMBINATORIAL OPERATORS

The purpose of the combinatorial operators is to provide a description to the system as to how two patch defined objects are to be combined in the creation of a third object. When the intersection subalgorithm completes, the tree is built and contains all the information necessary in order to apply these operators. That is, the approximation to the curve of intersection has been found and all subpatches have information regarding their new definition. They also have recorded the relative orientation with respect to the other object. This orientation is necessary to classify [14] the subpatches. For the purpose of exposition, suppose that we want to combine two objects A and B as shown in Figures 5 and 6. Figure 7 shows the two objects in the desired positions. The resulting shape is shown for each of the available operators using these two objects.

UNION(O1,O2) -- this operator results in a third object O3 which contains the entire 'outside' surfaces of objects O1 and O2. The two objects can be considered "welded" at the space curve(s) of intersection, and if they are solid objects, any surface area not relevant to the display (e.g., if part of O2 is inside of O1) need not be included in the description of O3. See Figure 8.

INTERSECT(O1,O2) -- this creates O3 from the space that is common to both objects, or the 'inside' surfaces of the two objects. See Figure 9.

CUT(O1,O2,S) -- this operator creates O3 by "cutting" O2 with the surface of O1. S is an option to determine which part of O2 will be retained as O3. If S = '+', that part of O2 outside of O1 is used. If S = '-', the complementary surfaces will be used. (See Figures 10 - 13.)

It can be seen that the operation INTERSECT and UNION can be derived directly from combinations of the set difference, or CUT operations. They were used as operators explicitly because they are more commonly used.

Since the tree structure used to store the subdivision retains an indication of the topological relationship, it is very easy to determine the resulting surfaces of the above operators. It suffices to traverse the tree along the links that go to the orientation records defined by the operator.

4. SUMMARY

The concepts of patch subdivision have been used several times before for computer display of surfaces [5,2,15,10]. The idea of using planarity as a termination condition for the subdivision process was used in a display algorithm by Lane and Carpenter and one by Clark [10,6]. Lane and Riesenfeld [11] used planar approximations to the subpatches as a means to determine the intersection of Bezier and b-spline patches (see also[7] for a thorough treatment of patch intersection issues). The primary difference between the ideas in this paper and those in the above references is that we do the intersection at object generation time, and incorporate the intersection information in the description of the resulting object. This reduces the time spent in rendering the object at each frame. If this intersection information isn't retained as part of the object description, it is necessary to recalculate the intersection each time a frame is generated. In an animation environment, where many frames using a given object need to be created for an animation sequence, the overhead can be quite significant.

The concepts included in this paper are currently incorporated in an integrated data generation system in use at the Computer Graphics Research Group. This system currently runs on a VAX 11/780. The objects generated by the subdivision process are being converted to a polygonal description before being displayed in solid shaded format using a polygon tiler. They are also displayed in vector format on a Megatek calligraphic display. Research is continuing to provide a patch display algorithm that can utilize the object description directly to render the object, rather than combining a patch display with a polygonal display algorithm. Moreover, the subdivision algorithm is being used to provide an efficient method for providing a compact polyhedral approximation to a patch defined object. Since only areas of excessive curvature need to be described as a collection of very small polygons, those areas of little curvature need not have more complexity than is necessary.

The slope values of the subpatches at the points of intersection are recorded in the leaf nodes as described in Section 6. We are interested in utilizing these values together with the curve of intersection to calculate "fillets", or smooth transitions where the patches intersect. We are particularly interested in modeling the human body as a collection of surface patches, and these fillets would provide the smoothness at the joints (eg, elbow, wrist, knee) that is desired for an accurate representation.

5. ACKNOWLEDGMENTS

The author would like to acknowledge the advice and motivation of Dr. Frank Crow. Prof. Charles Csuri provided the necessary support, and other members of the Computer Graphics Research Group added assistance, comments, and criticisms. Julian Gomez deserves special recognition for the effort he put forth to provide the electronic documentation software.

References

1. Baumgart, B.G., "Geometric Modeling for Computer Vision," AIM-249, STAN-CS-74-463, Stanford U. Computer Sci. Dept. (October 1974).

2. Blinn, James F., "Computer Display of Curved Surfaces," PhD Thesis , University of Utah (December 1978).

3. Braid, Ian C., "The Synthesis of Solids Bounded by Many Faces," Comm. ACM Vol. 18(4) pp. 209-216 (April 1975).

4. Brown, C. M., "PADL-2: A Technical Summary," IEEE Computer Graphics and Applications Vol. 2(2) pp. 69-84 (March 1982).

5. Catmull, Edwin E., "A Subdivision Algorithm for Computer Display of Curved Surfaces," UTEC-CSc-74-133, Salt Lake City, Utah (December 1974). University of Utah Dept of Comp Science.

6. Clark, James H., "A Fast Scan-Line Algorithm for Rendering Parametric Surfaces," Computer Graphics (SIGGRAPH 79 supplement) Vol. 13(3)(August 1979).

7. Cohen, Elaine, Lyche, Tom, and Riesenfeld, Richard F., "Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics," Computer Graphics and Image Processing Vol. 14(2) pp. 87-111 (October 1980).

8. Forrest, A. R., "Computational Geometry - Achievements and Problems," in Computer Aided Geometric Design, ed. Richard F. Riesenfeld,Academic Press, New York (1974).

9. Hunter, G.M., Efficient Computation and Data Structures for Graphics, Princeton U., Dept. of EE and CSc (1978). Ph.D. Dissertation.

10. Lane, Jeffrey M. and Carpenter, Loren C., "Scan Line Methods for Displaying Parametrically Defined Surfaces," Comm. ACM Vol. 23(1) pp. 23-34 (January 1980).

11. Lane, Jeffrey M. and Riesenfeld, Richard F., "A Theoretical Development for the Computer Generation of Piecewise Polynomial Surfaces," IEEE Trans. Pattern Analysis and Machine Intelligence Vol. PAMI-2(1) pp. 35-46 (January 1980).

12. Levin, Joshua Z., "A Parametric Algorithm for Drawing Pictures of Solid Objects Composed of Quadric Surfaces," Comm. ACM Vol. 19(10) pp. 555-563 (October 1976).

13. Parent, Richard E., "A System for Sculpting 3-D Data," Computer Graphics Vol. 11(2) pp. 138-147 Proc. Siggraph '77, (Summer 1977).

14. Tilove, R. B., "Set Membership Classification: A Unified Approach to Geometric Intersection Problems," IEEE Trans. Computers Vol. C-29(10) pp. 874-883 (Oct. 1980).

15. Whitted, J. Turner, "A Scan Line Algorithm for Computer Display of Curved Surfaces," Computer Graphics (SIGGRAPH 78 supplement) Vol. 13(3)(August 1978).

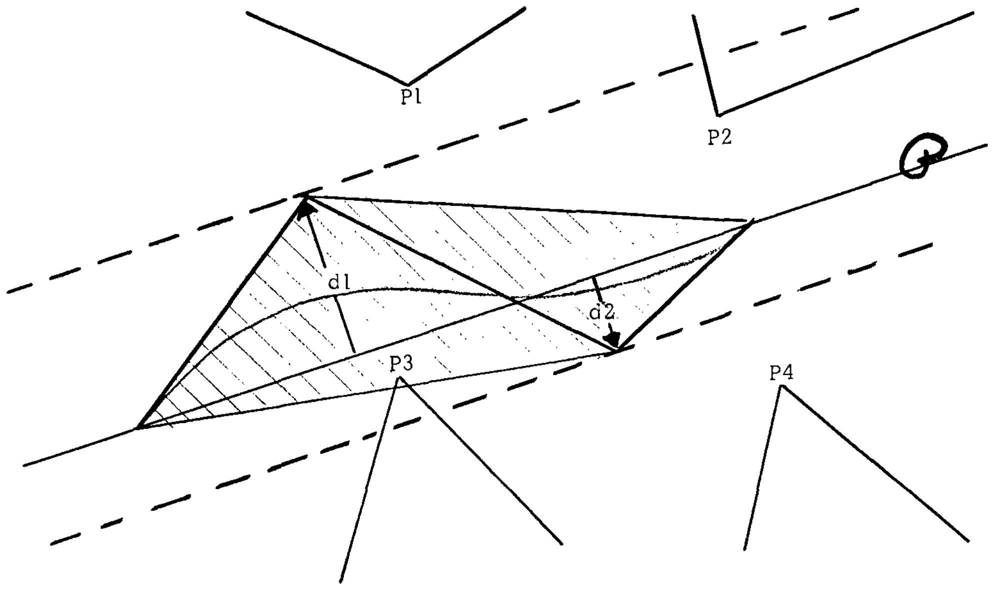


Figure 1

The shaded area represents (a 2D cross section of) the convex hull of the defining polygon. P is the plane derived from three of the four corner points. d1 is the maximum distance in the + direction, and d2 is the maximum distance in the - direction. The defining polygons containing the points P1 and P4 are separable, but no determination can be made with this test for those containing points P2 and P3.

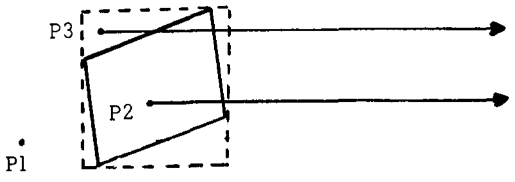


Figure 2

The solid box is the quadrilateral under consideration after being projected onto the plane that will result in the largest area. The points P1, P2, and P3 have also been projected onto the same plane. The dashed lines are the min-max enclosing box, and this test obviously eliminates P1 from further consideration. Point P2 is inside the box, since the semiinfinite ray crosses the projected edge an odd number of times, while the ray associated with P3 crosses an even number of times, eliminating it from consideration.

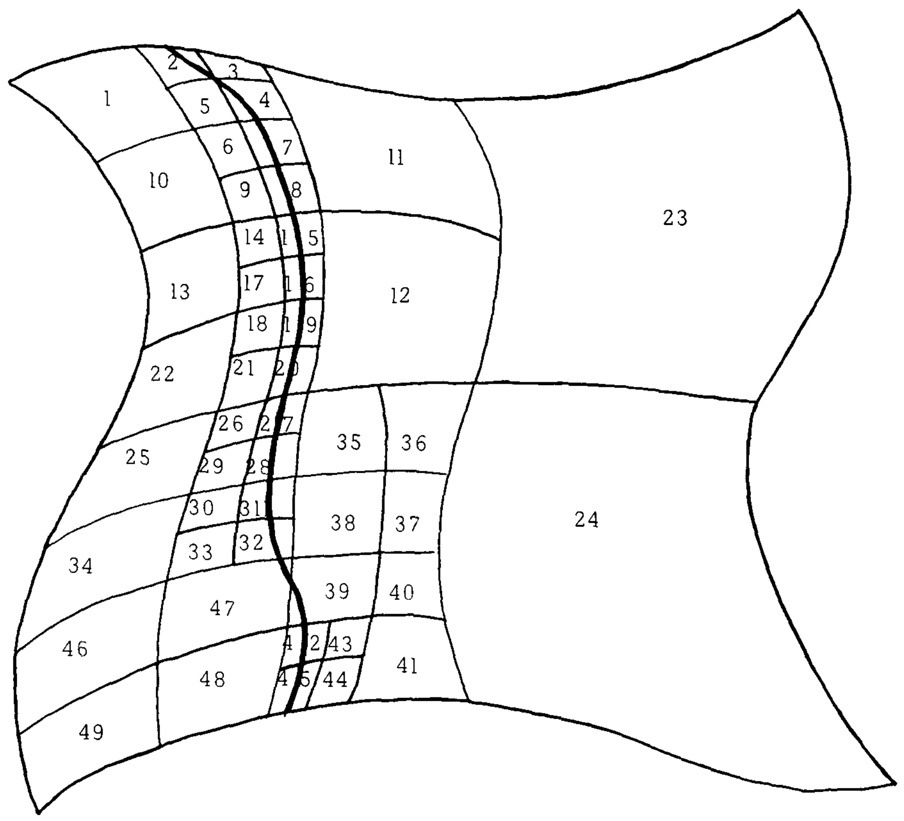


Figure 3

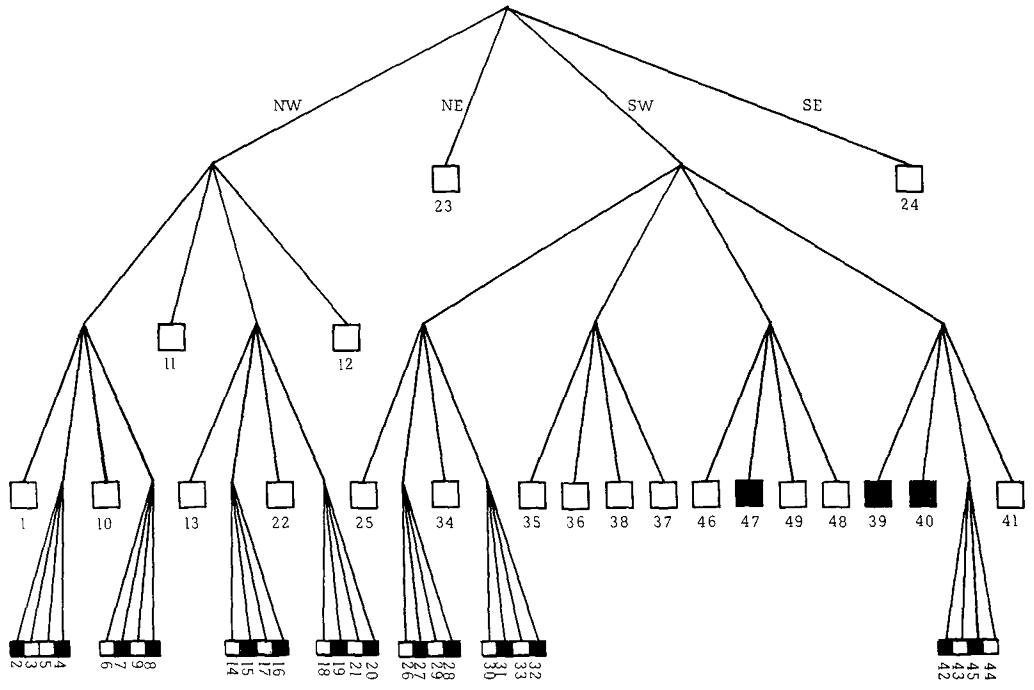
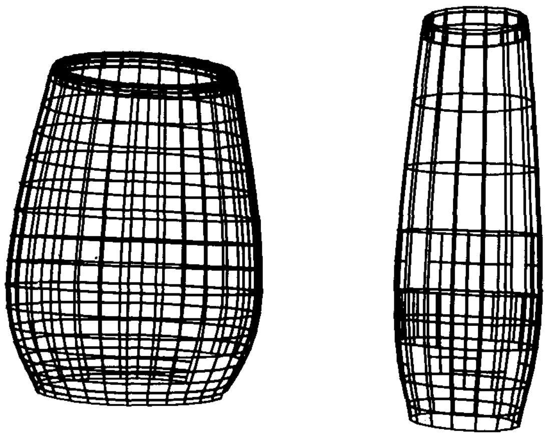


Figure 4


Figures 5 and 6
O1 and O2

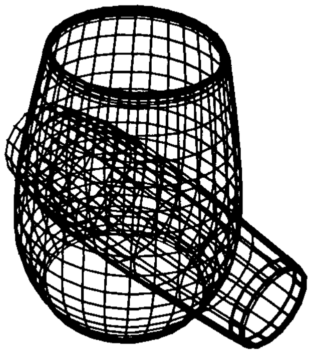


Figure 7

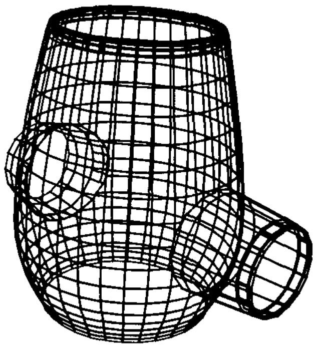


Figure 8
O3 = UNION(O1,O2);

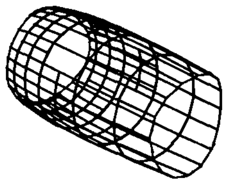


Figure 11
O3 = CUT(O1,O2,'-');

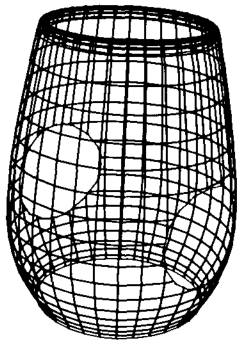


Figure 12
O3 = CUT(O2,O1,'+')

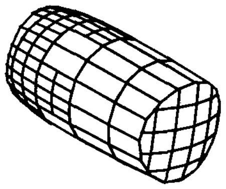


Figure 9
O3 = INTERSECTION(O1,O2);

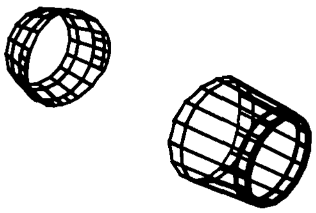


Figure 10
O3 = CUT(O1,O2,'+');

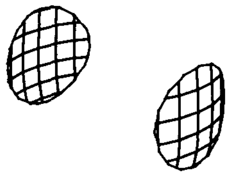
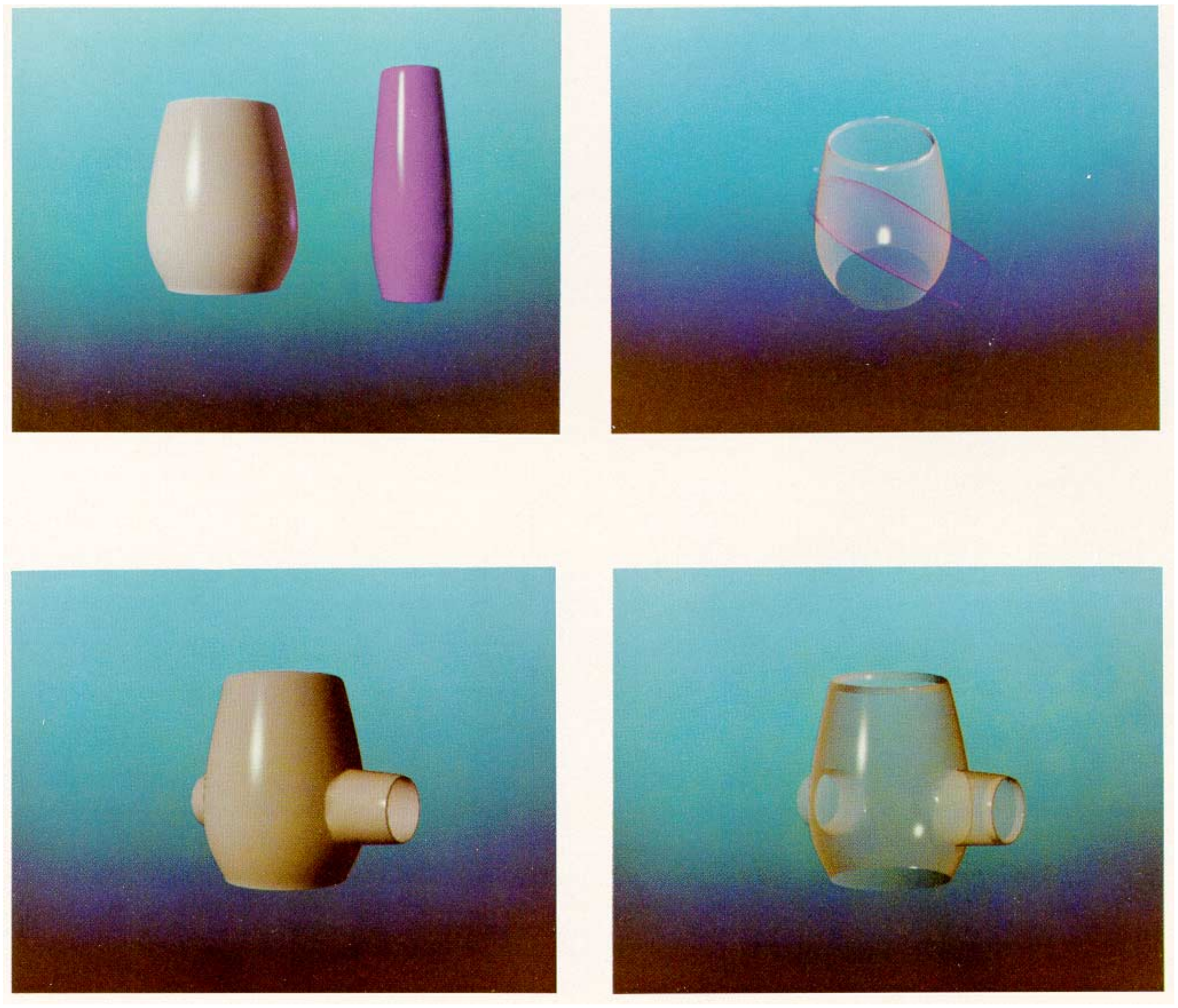


Figure 13
O3 = CUT(O2,O1,'-');



The raster images above show the original tubes, their desired orientation, and the object resulting from the UNION operator. Transparency is used to clarify the objects.