FEATURE EXTRACTION FROM TESSELLATED AND SLICED DATA IN LAYERED MANUFACTURING

Kamesh Tata — Prototype Express
Georges Fadel — Clemson University

Abstract

When parts are built in layers, the cross sectional area of each layer has to be defined and filled with a pattern of vectors. This filling process is called hatching and the vectors define the hatch pattern. To accurately reproduce a three dimensional object, key features need to be identified. In particular, top and bottom surfaces, edges, holes and protrusions must be recognized to ensure the slice plane does carry the critical information required for the build. This paper describes a technique to extract relevant features from a tessellated model to generate a correct sliced representation.

Introduction

Layered based manufacturing is based on a mathematical slicing operation that generates slices from an original boundary representation model (tessellated or triangulated model). Different processes use different devices to draw the hatch vectors which represent the areas to be filled or solidified in a slice. For instance, Stereolithography and Selective Laser Sintering use a laser to draw the vectors, while other processes may use a pattern mask or a binding agent.

Many slicing engines exist, in particular, the slicing program of 3D systems has a proven algorithm,Grogan [1990] described an algorithm based on sorting, Chalasani [1992] proposed two algorithms for slicing 3D objects that also rely on a sorting process to identify areas, Vouzelaud and Bagchi [1992] proposed an adaptive technique, Kirschman and Jara-Almonte [1992] investigated parallel slicing, and Dolenc and Mäkelä (1993) suggested slicing procedures which considered the identification of some features, notably peaks and flat areas.

This paper considers the issue of features and their extraction from the tessellated data. It highlights the capabilities of such an algorithm to identify manufacturing features and lists the pitfalls and their possible remedys.

The STL File

The *de facto* industry STL model representation defines CAD solid models as a set of triangular facets (3D Systems, 1989). Facet models represent solid objects by spatial boundaries which are defined by a set of planar faces. This is a special case of the more general boundary representation which does not require object boundaries to be planar (Mortenson, 1985). In general, the term facet is used to denote any constrained polygonal planar region used to define a model boundary; however, in the SFF community, the term facet is typically understood to mean triangular facet (Rock, 1991). Unfortunately, these facets are stored independently, as if each facet was created and tossed into a bucket with no particular ordering and without information relating a given facet to any other facet in the bucket (Rock, 1991).

Hole or Projection

One of the time consuming tasks of any hatching routine is identifying holes and projections. Holes should not be hatched, while projections should. At the time of hatching, the information about the solid is no more available in a single piece. Most of it is divided and stored in a number of 2–D slices and some of it is lost. This implies that identifying a single hole or a projection might require studying several slices together. This is computationally expensive. Thus there is a need for an efficient hatching algorithm which identifies, in the quickest possible way, areas that need to be filled and areas that need not be filled.

Strategy for Hatching

In order to identify the areas that need to be hatched, we examined various contours and derived the following rule : If a contour is surrounded by n contours and n is an even number, the contour represents a protrusion and should be filled. On the other hand, if n is odd, it the contour represents a hole and should not be filled.
This rule, which is central to the hatching algorithm developed as part of this research, can also be explained in another way. Considering 3–D solid objects,

1. a solid is always bounded,
2. there cannot be a solid without thickness,
3. and there cannot be a hole without a solid.

These three simple and definitive statements ensure that when a solid is sliced the resulting contours are invariably closed and non–intersecting. Also, there will never be two holes adjacent to each other or one contained in the other without being separated by a definite distance. Furthermore, any unbounded contour (contours not surrounded by any other contour) can never be a hole.

Key Characteristics Identifier

A simple knowledge base, supported by a few rules, is created with an aim to recognize the following from a tesselated model:

1. base faces of features,
2. type of features (protrusion or depression), and
3. geometric shape and orientation of features.

The rules are mostly based on simple geometric and engineering principles, and derived by observation and logical reasoning. Some of the ideas discussed can be utilized for any solid model regardless of the representation scheme.

The Three Types of Base Faces

The top and bottom faces of any feature, including the object, may be termed as base faces. Recognition of base faces is essential to improve the tolerances of layered models. Since a polygon is comprised of only two basic entities, point and line, only their combinations will result in any topologically significant characteristic. Consequently, any solid represented by planar polygons can have only three types of base faces:

1. Horizontal surface: A horizontal surface is represented by a single or multiple polygons parallel to the slice plane. Figure 1 shows horizontal faces composed of triangular facets.
2. Pointed edge: When at least two non–co planar polygons share an edge, the edge may be termed as a pointed edge. This is illustrated in Figure 2, where a block with v–grooves is shown.

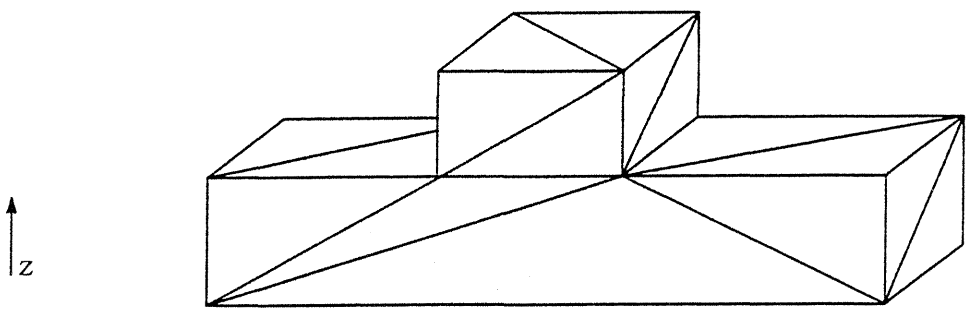


Figure 1. Multiple Facets Forming Horizontal and Vertical Surfaces

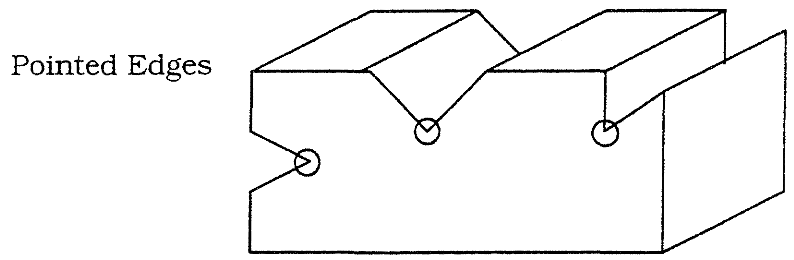


Figure 2. Block with Pointed Edges

3. Pointed End: When at least two non–co planar polygons share a vertex, the vertex may be termed as a pointed end. Tip of a cone and corner of a block, shown in Figure 3, are examples of pointed ends.

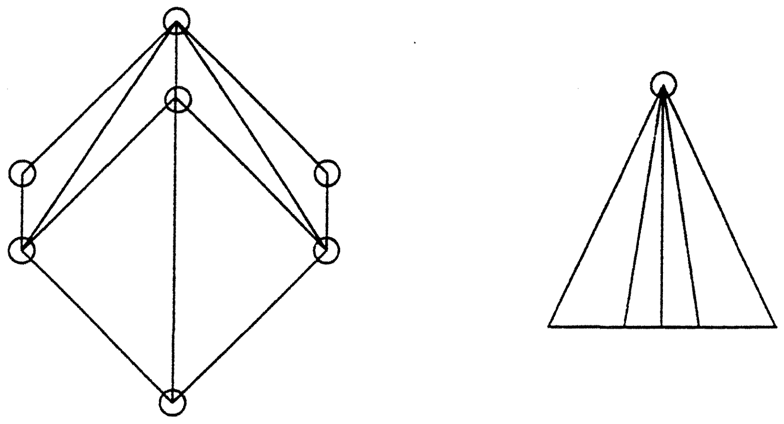


Figure 3. Examples of Pointed Ends

For the purpose of layered manufacturing, the effort must be to identify the existence and then to determine the height (along the slice axis and from a reference point) of each one of the above so that the slice plane can be forced to assume that height. Identification of these three types from an STL model format is basically a searching operation.

Identifying Horizontal Surfaces

Horizontal surfaces can be identified by checking unit normals or ordinate values of facets. If the unit normal of a facet describes 90^0 (or 270^0) with the slice plane, there lies a horizontal surface with that facet contained in that surface. Alternatively, if the z coordinate values of all the three vertices of a facet are equal, then that facet is part of a horizontal surface.

Vertical walls can be identified essentially using the same approach used to identify horizontal surfaces. If the unit normal of a facet makes 0^0 (or 180^0) with the slice plane, there exists a vertical wall with that facet contained in that wall.

Identifying Pointed Edges

According to the definition of a pointed edge, any edge shared by two non–co planar facets is a pointed edge. With this definition, there are twelve pointed edges, four vertical and eight horizontal, in a cube shown in Figure 4. It does not serve any purpose in the context of layered manufacturing to identify these edges separately, since the vertical and horizontal surfaces of which these edges are part of, can be easily identified using methods suggested earlier. Likewise, it is worthless to identify the edges of a tetrahedron shown in Figure 4 because none of the edges of the inclined surfaces are parallel to the slice plane. As we discussed previously, our primary goal is to identify key characteristics which are parallel to the slice plane so that we can retain them in the sliced model .

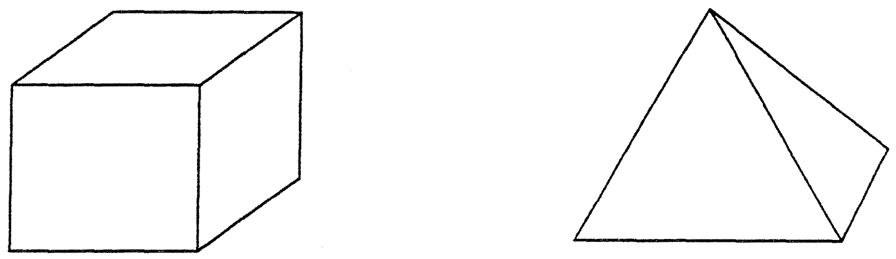


Figure 4. Unwanted Pointed Edges

It is, therefore, necessary to narrow the definition of a pointed edge, in the context of layered manufacturing. First, none of the polygons sharing an edge should be horizontal or vertical. This is due to that horizontal and vertical faces are already identified and with that all their edges. Further, the common edge must be parallel to the slice plane. (Only then a separate effort is required for its retention in the sliced model.)

A pointed edge can now be defined as: Any non–horizontal and non–vertical facet with two vertices having the same z coordinate value will form a pointed edge provided the second facet that shares these vertices (there will be only one such facet) is not in the same plane as the first facet and is non–horizontal and non–vertical. Also, the angle between those two facets need to be defined by the user. This is to avoid mistakenly identifying the edges of facets approximating curved surfaces as pointed edges. If we consider the faceted model of a sphere, according to the above definition, potentially every edge becomes a pointed edge. This can be avoided by defining the angle between the two facets sharing an edge to be less than a certain value, say, 120^0, before the edge can be considered as a pointed edge.

This new and narrow definition of a pointed edge helps reduce the time required to search for such edges in a faceted model. However, identifying two vertices of a facet with the same z coordinate value, the very first step in identifying a pointed edge, demands virtually every vertex of all non–horizontal and non–vertical facets in the model to be searched. This again leads to high execution times.

Identifying Pointed Ends

Finally, pointed ends can be identified from an STL file by looking for a single vertex shared by several non-co planar facets. For example, the apex of a cone is shared by a number of non-co planar facets which form the conical surface. This calls for checking every vertex in the model for the number of facets that share the vertex and their unit normals.

Base Faces - Change in the Number of Contours

Clearly, identifying pointed ends and edges is a time intensive search operation. Isolation of a vertex that forms a pointed end or a facet side that forms a pointed edge demands a large scale search often covering every facet of the model. This problem can be largely overcome by adopting a new strategy. This strategy is based on the fact that whenever there is a change in the number of contours in successive slices, there lies a base face of a feature. The exact location of the base face can be anywhere between the two slice planes or contained in the slice plane which has the greater number of contours. Once a change in the number of contours is observed, the exact location of the base face can be found by considering facet groups in that region and applying the methods suggested above. However, this strategy requires slicing the model and counting the number of closed contours in each slice.

Slice Based Feature Recognition - Protrusions or Depressions

Protrusions are any projections on the object, such as pads, bosses, and mounds, while depressions are holes of any shape. The easiest way of finding the type of a feature is by studying the slices and the contours within each slice. The following rules will provide methods to distinguish between protrusions and depressions. These rules are simple, definite, unambiguous, and true for any solid.

Rule 1: When a solid is sliced by an imaginary plane, the resulting slice may contain a single or multiple, non-intersecting, closed contours. This is true regardless of the angle and the position of the slice.

Before going further, it is necessary to define certain key words used in the preceding part of this section.

1. Empty contour: An empty contour is one which does not contain any other contour within its bounds. In other words, an empty contour does not surround any other contour. See Figure 5.
2. Unbounded contour: An unbounded contour is one which is not surrounded or bounded by any other contour.

Rule 2: If a single or multiple empty contours are surrounded by a single larger contour, all inner contours represent depression features.
This can be seen in Figure 6, where a rectangular block with several holes is shown. The slice at AA has several empty contours surrounded by a single large contour. Obviously, all inner contours represent holes and hence the rule 2 is satisfied.

Rule 3: If two or more contours are nested within each other, all even-numbered contours represent depressions while all odd-numbered contours represent protrusions, the outermost contour being number 1. Figure 7, where a cylindrical block with multiple holes one inside the other is shown, illustrates rule 3.

Rule 4: Any unbounded contour represents a protrusion feature.
Figure 8, where a number of protrusions are shown on the top of a rectangular block,

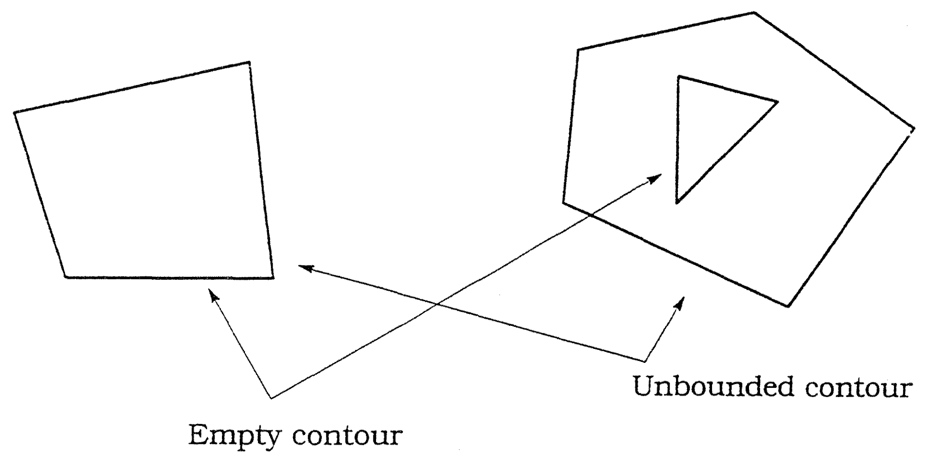


Figure 5. Empty and Unbounded Contours

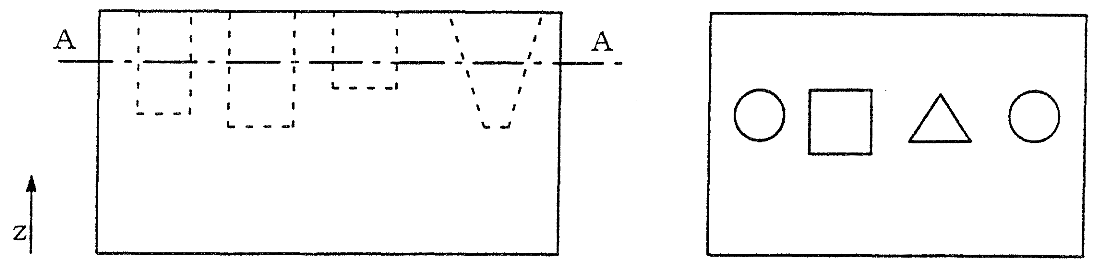

A
A
Z
Front view of a block with four holes
Slice at AA (view along Z)

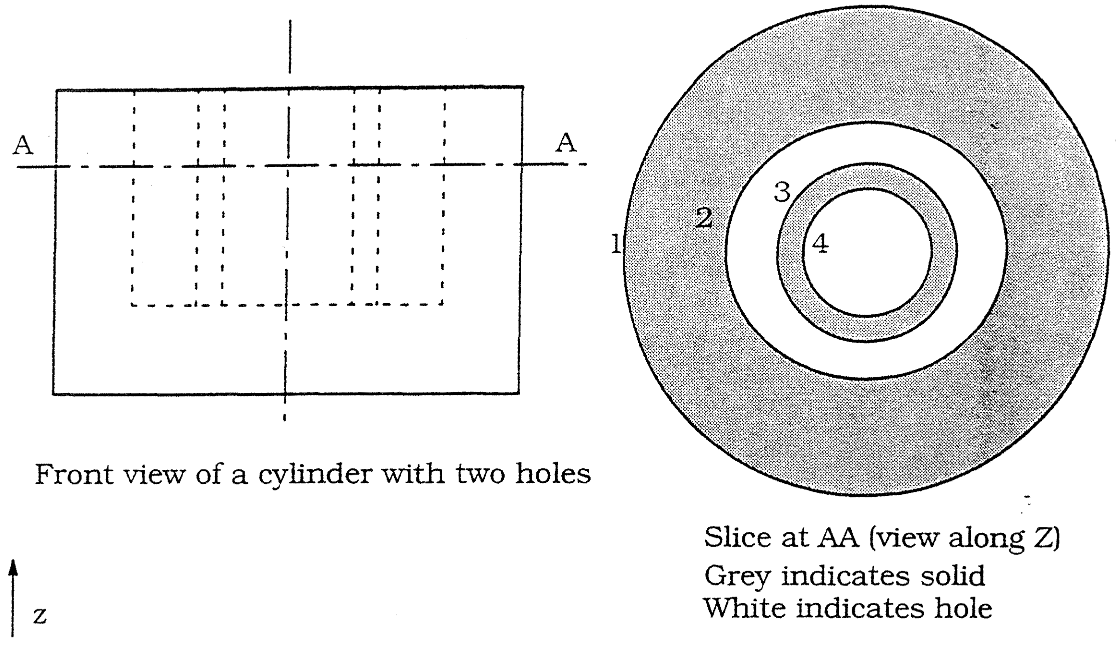
Figure 6. Multiple Empty Contours within a Larger Contour

clearly illustrates this point. The slice at AA contains several unbounded contours and all are, obviously, protrusions. This satisfies rule 4.

Shape and Orientation of Features

Once, location and type of a feature are determined, further details can be obtained by studying the unit normals of the facets the feature is comprised of. The following assumes that holes and projections, and base faces are already identified using the methods suggested above.

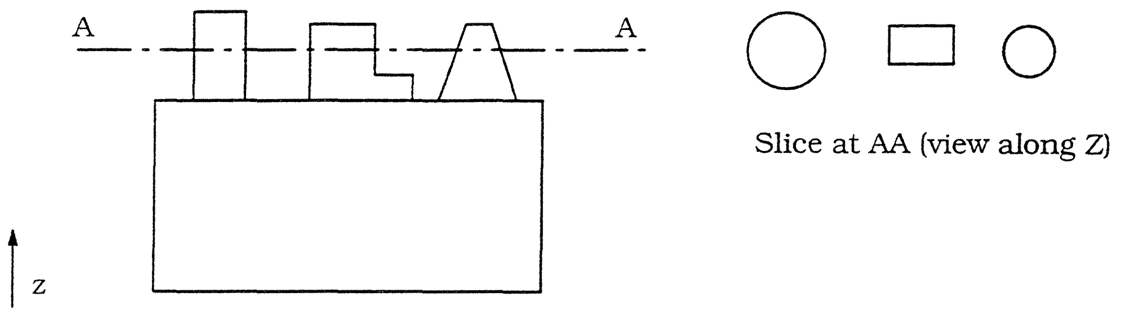
A feature is rectangularly shaped, if it is bounded by four faces, and each face is perpendicular to its adjacent face. Again, all four must be perpendicular to a common plane. If a fifth face that connects all four faces exists, the feature is closed at one end. It is closed at both ends if a sixth face also exists connecting the first four faces. Furthermore, the feature can be a rectangular pad or a rectangular hole depending upon its nature (protrusion or depression). Triangular pads or holes are bounded by three faces, where the sum of the included angles between the faces is exactly 180^0 and all three are perpendicular to a common plane. The type of triangle can also be easily established by studying the included angle between faces. This logic can be extended to identify pentag-



Front view of a cylinder with two holes

Slice at AA (view along Z)
Grey indicates solid
White indicates hole

Figure 7. Multiple Contours Surrounding Each Other



Slice at AA (view along Z)

Front view of a block with three protrusions

Figure 8. Unbounded Contours

onal, hexagonal or any polygonal feature. Orientation of the feature can be established by determining the angle between the slice plane and one of the bounding faces. Different types of features that can be easily identified are shown in Figure 9.

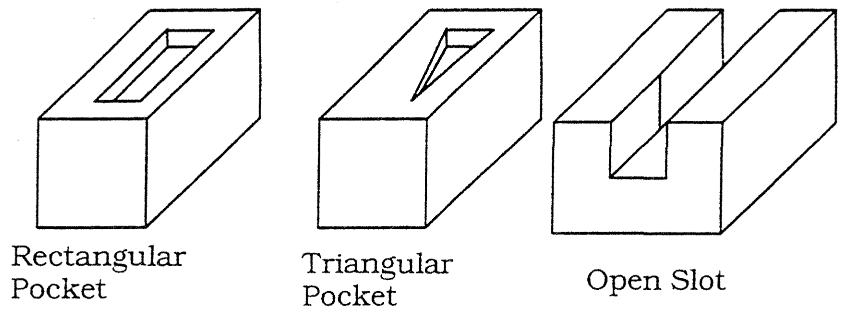


Figure 9. Typical Features Which Can Be Identified from Tesselated Models

One limitation with slice based feature recognition is that features parallel or at certain angles to the slice plane will be difficult to identify. In order for a comprehensive feature recognition, slicing may have to be done at different angles and the resulting slices studied independently and together. For example, in Figure 10, holes A and B, which are parallel to the slice plane, cannot be identified. Hole C, which is perpendicular to the slice plane, can be easily identified as a depression feature. In order to identify A and B, slicing has to be done along x or y.Feature recognition is a vast area and the work done here is only a small effort toward that goal.

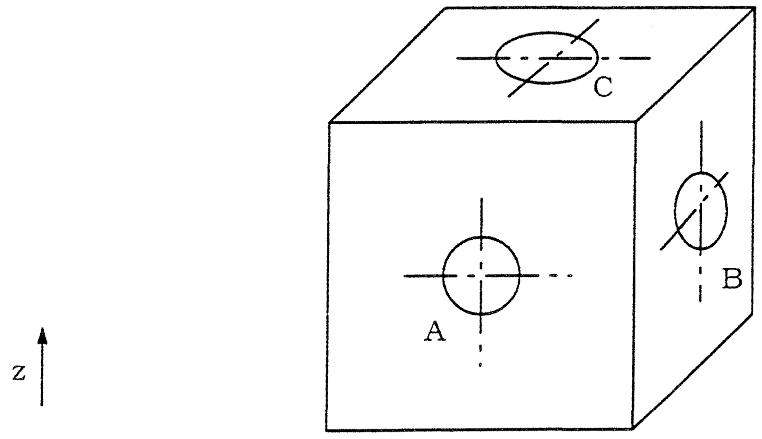


Figure 10. Block with Three Holes Along the Three Axes

Conclusions

This work describes a possible methodology for the identification of features from layered or tessellated data. A simple rule base is used to identify such features. The method does have significant limitations since features are only recognized when along the build axis. However, if one considers what features are used for, especially to identify hatching areas, then this feature recognition is sufficient. If the features are needed for design purposes, then the approach has to be performed three times, one for each coordinate axis as the build direction.

References

1. 3D Systems Inc., (Valencia, CA), 1989, "Stereolithography Interface Specification."
2. Chalasani, K. L., Grogan, B., Bagchi. A., Jara–Almonte, C. C., Ogale, A. A., and Dooley, R. L., "An Algorithm to Slice 3–D Shapes for Reconstruction in Prototyping Systems", Computers in Engineering – Volume One, ASME 1991.
3. Chalasani, K. L., "Design and Implementation of Slicing and Hatching Algorithms for Freeform Fabrication", M. S. Thesis, Clemson University, Clemson, S. C., 1992.
4. Dolenc, A., and Makela, I., "Slicing Procedures for Layered Manufacturing Techniques", Otaniemi 1992 I TKO–B83, Helsinki University of Technology, Finland.
5. Frank, D., Geuer. A., and Fadel, G. S., " Definition of Rules for a Preferred Orientation of Rapid Prototyping Parts with the Help of an Expert System", Technical University of Munich and Clemson University, 1993.
6. Grogan, B., "Development of a Slicing Algorithm for Laminated Part Manufacturing", Undergraduate Research. Clemson University, Clemson, S. C., 1990.
7. Kirschman, C., and Jara–Almonte, C. C., "A Parallel Slicing Algorithm For Solid Freeform Fabrication Processes", Proceedings of the 1992 Solid Freeform Fabrication Symposium, Austin, TX, August 3–5.
8. Mortenson. M. E., "Geometric Modeling", John Wily and Sons, New York, N. Y., 1985.
9. Rock, S. J., and Wozny, "A Flexible File Format for Solid Freeform Fabrication", M. J., Rensselaer Design Research Center, Rensselaer Polytechnic Institute, Troy, N.Y., 1991.
10. Tata, K. M., "A Marching Algorithm for Adaptive Slicing of 3–D Parts for Reconstruction in Prototyping Systems", Course Project for CE 690, taught by Dr. N. M. Aziz, Clemson University, Clemson, S.C., 1993.
11. Vouzelaud, A. F., and Bagchi. A., "Adaptive Laminated Machining for Prototyping Dies and Molds", Solid Freeform Fabrication Symposium, Proceedings, August 3–5, 1992, University of Texas at Austin.