Utilizing Topological Information to Increase Scan Vector Generation Efficiency

Stephen J. Rock
Michael J. Wozny

Rensselaer Design Research Center
Rensselaer Polytechnic Institute
Troy, New York 12180

Abstract

Demands for increased Solid Freeform Fabrication precision and speed suggest the need for advanced scanning techniques, such as boundary tracing, half-lap and multiple orientation scanning, or 'intelligent' scanning. Since most SFF processes construct parts from parallel material layers, separating model slicing and scan conversion functions appears to be a powerful approach. Both can benefit from increased topological information. This paper addresses the issue of improving model slicing by utilizing topological data to increase performance, and consequently, improve the efficiency with which scan vectors can be generated.

1 Introduction

Most Solid Freeform Fabrication (SFF) processes construct objects from parallel material layers [1]. In the case of Selective Laser Sintering (SLS), laser energy is used to selectively fuse successive layers of powdered material to create a three dimensional solid part [2]. Laser energy is directed by a pair of computer controlled galvanometers attached to small mirrors; however, existing beam-position-controller hardware is limited to generating only linear beam traversals across the sintering surface [3]. Consequently, we will assume that coordinate pairs defining each linear beam traversal are the lowest level information required to control SFF hardware. These coordinate pairs define linear segments which are termed *scan vectors* when generated for scanning control.

There are many possible interface levels and techniques for transforming high level CAD data to low level scan vectors. Information complexity is typically reduced at each successive interface. Data volume tends to experience an inverse relationship as it nears the goal of scan vector data. Unfortunately, the information loss associated with each transition toward scan vectors carries with it increased validity checking difficulty and decreased flexibility. For example, given contours created by intersecting a plane with a model, termed *slices*, it is difficult to generate slices with a different slice plane orientation. Or, given parallel scan vectors, it is difficult to accurately generate boundary contours. By passing a three dimensional model to the process controller, flexibility is increased as is the ability to assure model validity. Ideally, native CAD system data comprising a part would be passed to the process controller which would then generate scan vectors from it. This is impractical, in part, because of the large number of geometric modeling entities and representations used by various modeling systems, as well as proprietary issues. Compatibility maintenance would be a never-ending and costly job. Thus, an intermediate format is the best compromise.

1.1 Intermediate Data Exchange Format

An intermediate three dimensional model format is used to transfer information between CAD systems and SFF processes. Facet models provide the greatest common denominator between the multitude of CAD systems which must be supported. Flexibility required for part re-orientation and nesting exists, without the undue burden of recognizing

a nearly infinite number of model entities. Using facet models provides a particular advantage for industries using proprietary surface representations. By tessellating surface models to the precision required for SFF processing, a company is able to control access to its surface representations by providing manufacturing or outside service bureaus with models containing no details about the representation from which the data was generated.

Such intermediate model formats exist. The STL format represents solids as a series of triangular planar facets, each with a corresponding face normal [4]. This format is simple and has become the de-facto industry standard in the SFF community. Unfortunately, STL files carry redundant information which makes them unnecessarily large. They contain nearly the minimal information required to represent a solid model, and this makes model processing a computationally expensive task. However, an alternative format which includes both facet models and CSG primitives, named the *RPI format*, has been developed [5]. It carries facet topological information and is less redundant than its STL counterpart. The task of ensuring a model is physically realizable and contains a valid representation (i.e., no missing facets, etc.), termed *model validity checking*, is simplified by the presence of topological information.

The slicing algorithm presented in this paper requires topological information, and the experimental implementation makes use of a sub-set of the RPI format. This avoids the costly step of inferring model topology from an unconnected set of facets.

1.2 Scan Vector Generation Approaches

There are many ways to generate scan vectors from a CAD model. Initial work at the University of Texas at Austin used a voxel based approach [6]. A three-dimensional matrix of voxels, or volume elements, was superimposed on a model and the model was evaluated at each voxel. Scan vectors were then generated by searching for transitions of adjacent voxels in the matrix. While any orientation could be selected for voxel matrix positioning, scan vector directions were limited to the primary axial directions of the voxel matrix. Models were typically quantized to 20 mil x 20 mil x 5 mil voxels. This assumes a beam diameter of 20 mils and a slice thickness of 5 mils. Available scanner resolution is finer than 20 mils, so unnecessary quantization error was introduced along one scanning axis. To overcome this limitation, the model needs to be sampled at a higher resolution.

The ray-casting approach to model slicing provides another alternative for scan vector generation and was studied at Rensselaer Polytechnic Institute [7]. This technique generates scan vectors by firing parallel rays in three-space through a model. The first, and all additional, odd ray-object intersections mark the start of a scan vector. Each subsequent intersection delimits the scan vector end point. The parallel rays can be directed along any space vector; however, the technique can only produce parallel scan vectors. Unlike the voxel based approach, ray/object intersections are computed exactly and thus can make full use of available scanner resolution in the direction of the scan vector. Memory requirements also do not become unwieldy as slicing resolution increases.

The drawback to these two approaches is that they assume only parallel scan vectors are required. While this is consistent with an abstraction of 2D raster graphics, the galvanometer scanning system is capable of vector traversals which more closely resembles early vector graphics technology. There is an intuitive benefit to advanced scanning techniques such as boundary tracing, half-lap and multiple orientation scanning, or 'intelligent' scanning. These advanced scanning methods are difficult to implement with either the voxel based or ray-casting approaches. Such advanced scanning techniques require two dimensional topological information in the slice plane. It is best to pass this information during the slicing process instead of attempting to later infer it in the plane. Separation of the model slicing and scan conversion functions provides a sequential dimensionality reduction from 3D to 2D to 1D space. Three dimensional models are converted to two dimensional slices which are then used to generate one dimensional scan vectors. All information required by each stage should be passed by the preceding stage.

No inferencing should be necessary as this is typically a costly task. The sequential dimensionality reduction allows algorithms to be tailored to the information level at which they are operating and thus ignore higher level information. Such modularity will also simplify research on specific parts of the CAD interface to SFF.

Intermediate facet representations may be surpassed by more sophisticated representations in the future; however, they will likely predominate as the highest level model input to SFF processes in the near term and continue to be supported as one entry point to SFF processes in the future. Linear scan vectors have been defined as the lowest level SFF process input given available controller technology, and separation of the slicing and scan conversion functions has been identified as a useful means of generating scan vectors.

This paper focuses on the development and experimental results of a topologically based facet model slicing algorithm. An overview of potential scanning methodologies is also presented to illustrate that topology in the slice plane can further increase scan vector generation capability.

2 Facet Model Slicing

Facet model slicing performance can be increased by utilizing topological information. Instead of randomly intersecting each facet with a slice plane and having to construct the slice contour later, it is possible to march from facet to neighboring facet collecting contour information as the march proceeds. One method of generating each contour segment during the march is to sequentially intersect each facet (a constrained plane) with the slice plane (an infinite plane). A second method utilizes edge information and generates contour segments by marching from edge to edge. Each edge can be treated as an infinite line and this can be intersected with the infinite slice plane. The second method is obviously easier.

Certain topological information is required to make use of the second method. Each face must reference its three neighboring faces and three edges. Each edge must reference its two vertices and the two faces which define that edge. More detailed topological information is unnecessary for the slicing algorithm; however, face normal information is required and should be carried into the two dimensional slice contours for efficient scan conversion.

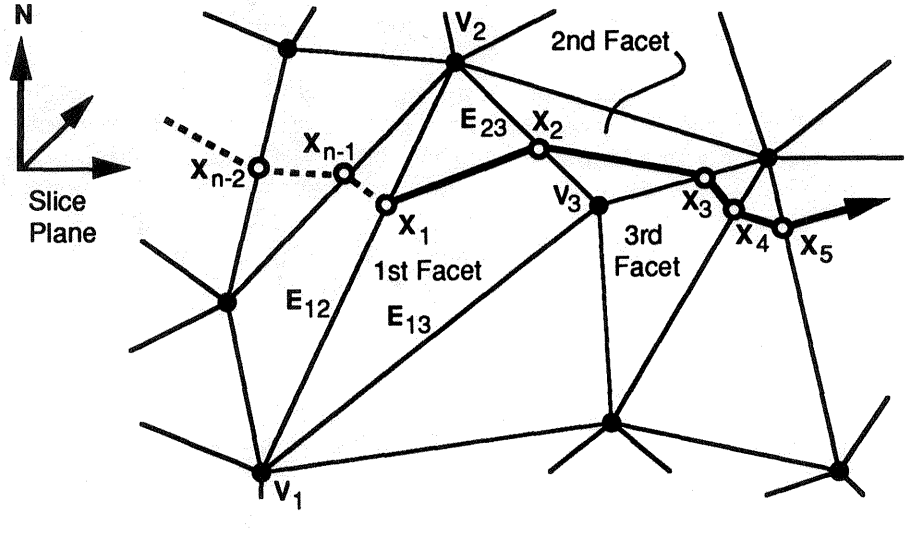

Figure 2.1 - Connected Facets in Three-Space

2.1 Fundamental Algorithm

The fundamental algorithm begins by locating an edge, for instance E_{12} in Figure 2.1, which is intersected by the slice plane. Since E_{12} is known to intersect the slice plane, the intersection point $\mathbf{X_1}$ (bold variables indicate vector values) can be determined by treating the edge as an infinite line and intersecting it with the infinite slice plane. If the line is represented as a parametric line, as shown in Figure 2.2, a parametric value ranging from zero to one will specify the intersection location.

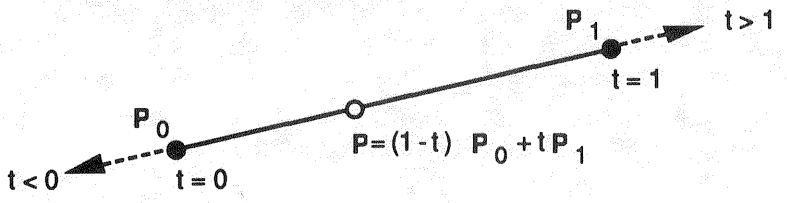

Figure 2.2 - Parametric Line Representation

The slice plane intersection parameter *d* can be computed,

$$d = \frac{\mathbf{N} \cdot (\mathbf{V_1} - \mathbf{P})}{\mathbf{N} \cdot (\mathbf{V_1} - \mathbf{V_2})} \qquad (2.1)$$

given a slice plane defined by a plane unit normal **N**, a point on the plane **P**, and the edge vertices $\mathbf{V_1}$ and $\mathbf{V_2}$. The intersection point in three-space can be determined,

$$\mathbf{X} = (1 - d)\,\mathbf{V_1} + d\,\mathbf{V_2} \qquad (2.2)$$

given the intersection parameter *d* and the edge vertices $\mathbf{V_1}$ and $\mathbf{V_2}$.

2.2 Marching Orientation

After calculating the edge intersection, the algorithm must determine a direction in which to march while creating the slice contour. This decision is arbitrary but determines which face sharing edge E_{12} is made the 'first facet'. The algorithm must next determine which remaining edge in this facet is intersected by the slice plane. This is found by evaluating the location of vertex $\mathbf{V_3}$, the vertex opposite from the previously intersected edge, with respect to the slice plane. Given the slice plane unit normal **N**, a point on the plane **P**, and the vertex under test $\mathbf{V_3}$, the dot product,

$$\mathbf{N} \cdot (\mathbf{V_3} - \mathbf{P}) \qquad (2.3)$$

can be computed and only its sign must be tested. If the resulting sign is positive, $\mathbf{V_3}$ is on or above the slice plane; where above is in the direction of the slice plane normal. If the resulting sign is negative, $\mathbf{V_3}$ is below the slice plane. If, by convention, $\mathbf{V_1}$ is below the slice plane and $\mathbf{V_2}$ is above the slice plane, then the next edge is easily determined. When $\mathbf{V_3}$ is above the slice plane, E_{13} will be the next edge intersected. When $\mathbf{V_3}$ is below the slice plane, E_{23} will become the next edge intersected. The special case where an edge lies in the slice plane is avoided by perturbing all vertices coincident with any of the slice planes by an insignificant epsilon. While initial marching orientation selection is arbitrary, it is important to ensure that marching proceeds in a unidirectional fashion. This cycle continues for subsequent facets until the initial edge is reached. When this occurs, one complete slice contour has been generated.

2.3 Multiple Contours in a Slice

Multiple contours may exist in one slice. After completing a march for one contour, a search is performed to locate an unvisited edge which intersects the slice plane. The status of an edge (visited/unvisited) is maintained by a flag associated with each edge. If one is found, the marching algorithm is repeated and another contour is generated. After marching is completed for all contours in a given slice, these flags are cleared and the same steps can be applied for subsequent slices.

2.4 Algorithm Enhancement - Linear Interpolation

There are many ways to enhance the above algorithm. Investment in a little pre-processing often pays a healthy dividend. The intersection and vertex evaluation computations outlined above do not need to be performed each time marching takes place. The *d* parameter being computed in equation 2.1 could be found by linear interpolation if the distance from each vertex to the slice plane was known. Furthermore, these distance values do not need to be recomputed for each slice plane position change. Since the slice plane normal remains constant during model slicing, it is possible to pre-compute *d* (distance) values for all model vertices from some reference slice plane. These values can simply be offset based on the active slice plane position relative to the reference slice plane.

Computations become even simpler with this scheme. The *d* parameter computation found in Equation 2.1 can be replaced with

$$d = \frac{d_{slice} - d_1}{d_2 - d_1} \qquad (2.4)$$

which is only.a scalar equation. Here, d_1 is the distance from vertex $\mathbf{V_1}$ to the reference slice plane, d_2 is the distance from vertex $\mathbf{V_2}$ to the reference slice plane, and d_{slice} is the distance from the active slice plane to the reference slice plane. Equation 2.2 can again be used to determine the intersection point in three-space. This technique eliminates the effective re-calculation of vertex distance values which occurs when more than one slice plane intersects a model edge. It replaces repetitive computation of two vector differences and two dot products with two scalar differences. This preprocessing is beneficial if there is, on average, more than one slice plane intersection per model edge.

2.5 Algorithm Enhancement - Spatial Partitioning

Spatial partitioning provides a second enhancement method. The slicing algorithm is straight-forward after it locates an edge on which to begin marching. In a large model, however, repeatedly locating an edge could be costly. Also recall that each edge has a flag to indicate whether it has been visited and this flag must be cleared after each slice. By partitioning edges, edge searching and flag clearing will require less time. The model space can be divided into regions, or bins, along the direction of the slice plane normal. All model edges are searched, and each edge is 'registered' with all bins it intersects. Figure 2.3 illustrates edge spatial partitioning.

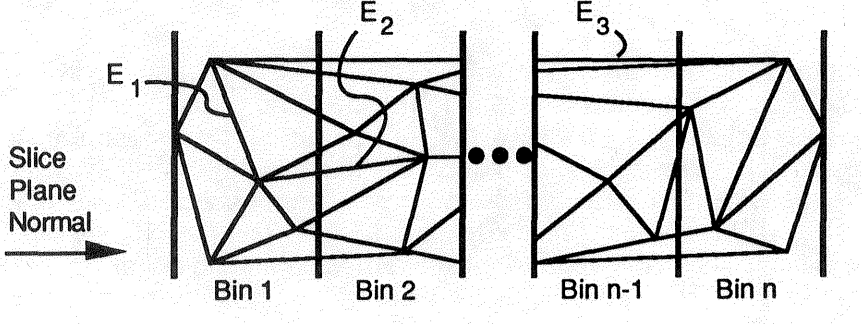

Figure 2.3 - Edge Spatial Partitioning

Some edges, such as edge E_1 will be partitioned into only one bin. Other edges will cross bin boundaries, such as edge E_2, and be partitioned into multiple bins. It is even possible to have an edge referenced by every bin. This is the case for edge E_3.

This enhancement, like the first one, is data dependent. If most model facets are long slivers whose greatest length is oriented along the slice plane normal vector, they will be referenced by most partitions. This will consume additional memory and provide no computational speed increase. In fact, the use of bins will either require one more layer of

pointer references to edge data or become severely memory intensive if complete edge data is replicated multiple times.

Cases exist where both enhancements fail. While a thorough study has not been conducted on the statistics of facet models, it is unlikely that model data is too unfavorable to benefit from the enhancement techniques presented. Most facet models are large in comparison to the size of each facet. Many facets are typically required to represent even the simplest parts, and this is especially true for surface approximations.

3 Results

The slicing algorithm described, with both enhancements, was implemented and experimentally evaluated. The experimental implementation generates three-dimensional polyline contour data which serves as input to a graphics display utility program. Figure 3.1 shows a facet model of a tooth nerve cavity reconstructed from CT scan imaging data [8]. It is represented by over 7000 facets. Figure 3.2 shows the contours of this facet model generated by the topological slicing algorithm presented. Note that multiple disjoint contours exist in some slice planes.

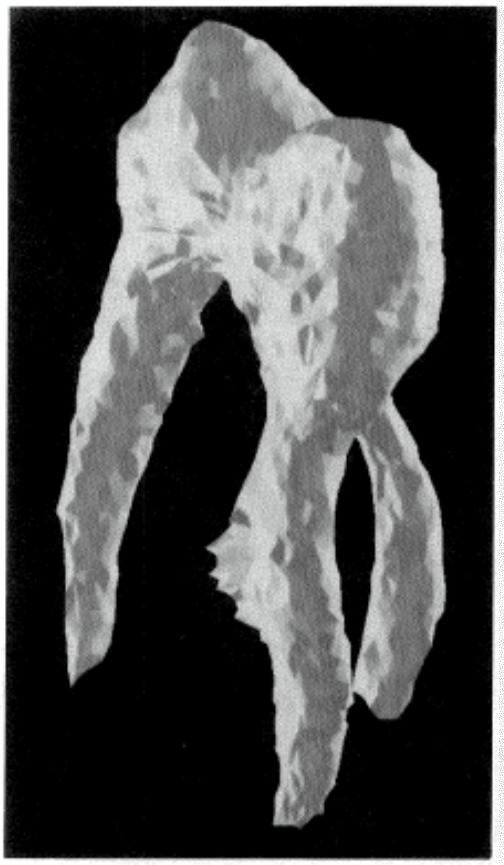

Figure 3.1 - Tooth Nerve Cavity

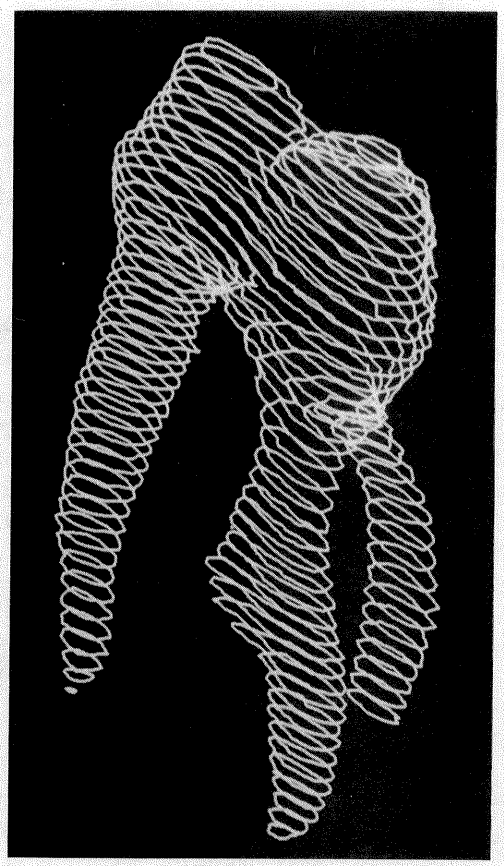

Figure 3.2 - Sliced Contours

The experimental implementation serves to prove the concept, but it has not been fully optimized. Attention was focused on the slicing algorithm with minimal concern given to optimizing the preprocessing phases. With this understood, computational performance can be analyzed. Figure 3.3 contrasts computational cost, both of preprocessing and actual contour generation, with the number of slices required.

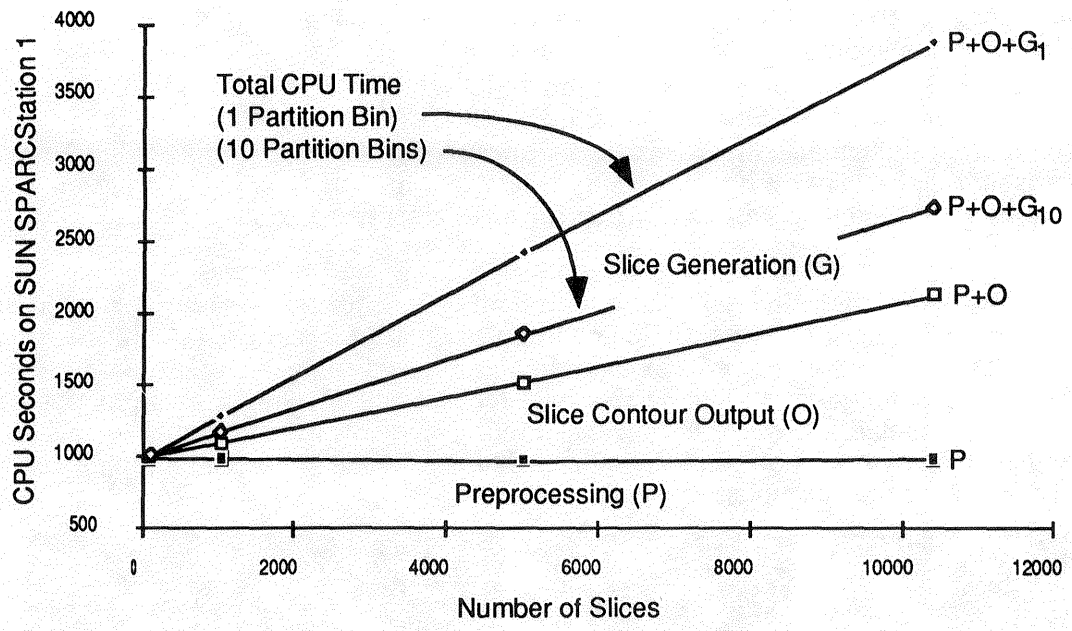

Figure 3.3 - Computational Cost Graph

Preprocessing includes parsing the RPI format file, connecting its topology, performing spatial partitioning, and precomputing d values for each vertex. Slice Contour Output time is contributed by the function which writes the three dimensional contour data for previewing. The time required for slice generation, added to Preprocessing and Slice Contour Output costs, represents the total CPU time required to perform model slice contour generation. The benefit of utilizing spatial partitioning optimization is shown by the difference between the Total CPU Time plots.

Only one orientation of the tooth nerve cavity shown in Figure 3.1 was used to generate this data. It is impossible to predict with accuracy the computational cost of model slicing based solely on the number of slices required. Part geometry, facet geometry and distribution, and slicing orientation all affect processing cost. No alternative slicing method performance data was available; however, actual marching about a model contour is linear with respect to the number of facets in the model. No searching is required during a march because topological information is used to direct the march. Computational expense is incured when searching for an initial edge on which to begin the march, and this expense is reduced by spatial partitioning. For the test case shown in Figures 3.1 and 3.2, performance better than 3 slices per second was achieved.

4 Scanning Methodologies

Topological data has improved model slicing algorithm performance. It will also be useful for taking the resulting contours and generating scan vectors. Efforts to date have focused on the raster graphics abstraction where all scan vectors are parallel and aligned with one of the scanning axes. As physical process limitations become increasingly challenging, more reliance must be placed on innovative scanning techniques.

A number of scanning methodologies were mentioned earlier. They should be generally applicable to any SFF technology which creates parts by directing energy. For illustrative purposes, their use in Selective Laser Sintering is described. Unfortunately, experimental data on these techniques is unavailable.

4.1 Boundary Tracing

Boundary tracing may improve part surface finish by tracing the part boundary, offset by the appropriate beam radius. This should allow greater beam switching time and

stabilization tolerances on the scan vectors which are used to fill the interior of each part layer. It may also provide antialiasing effects, illustrated in Figure 4.1, when only parallel scan vectors are used to fill a part layer. The darkened regions of the rightmost figure indicate additional material solidified as a result of boundary tracing. Topological information in the slice plane will allow boundaries to be traced in a continuous fashion, instead of as disjoint vectors.

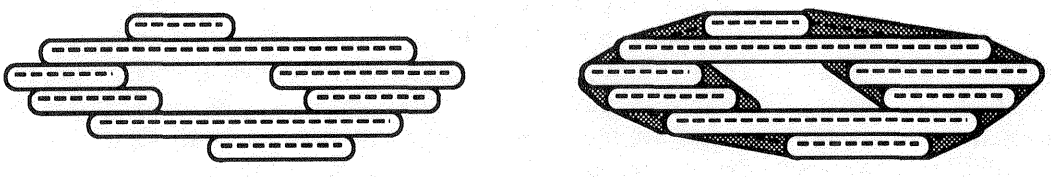

Figure 4.1 - Antialiasing Effects of Boundary Tracing

4.2 Half-lap and Multiple Orientation Scanning

Half-lap and Multiple Orientation scanning may be useful to create parts of increased structural integrity or density. Few laser beams are of ideally uniform energy distribution across their diameter. Even if they were, the geometry of the ideally circular beam traversing the sintering surface yields non-uniform energy deposition into the surface. Half-lap scanning proposes the laser beam be offset by only half of its beam diameter between neighboring fill vectors. Alternatively, multiple fill orientations could be used between adjacent part layers or re-scanned on the same material layer. This may keep shear planes from developing in the part and hence make it structurally stronger. Topology in the plane may be useful when orienting the multiple scan directions.

4.3 "Intelligent" Scanning

Finally, topological information may be useful for implementing an 'intelligent' scanning technique which attempts to scan a part with as few discontinuous scan vectors as possible. Assuming there is a setup cost associated with each disjoint scan vector, reducing the number of such scan vectors will increase production efficiency. Without the topological information for each slice, "intelligent" scanning will be difficult at best.

5 Conclusion

5.1 Conclusions

By improving the slicing component of scan vector generation, overall scan vector generation speed increases. Given the test case performance, it is clear that the slicing can be made an on-line operation at the process controller. This will enable operators to nest parts or re-orient parts at run time. Other process parameters, such as beam diameter, slice thickness, or scanning methodology, can then be altered without requiring additional time or resources for off-line processing. The implementation must be benchmarked using a broad sampling of industrial models, but with few extreme case exceptions, it is unlikely that algorithm performance will degrade significantly.

It is also unlikely that parallel scan vectors alone will be adequate for the future of SFF technology. Innovative scanning techniques will be required to push the part precision envelope further. Topological information, both for the three dimensional model and for each model slice, will likely continue to be an asset for scan vector generation.

5.2 Future Research

Based on the graph of Figure 3.3, preprocessing phases of model slicing should clearly be made more efficient. This may include adaptive partitioning, where partition bin size and location is based on model statistics instead of a fixed number of partitions. Statistical analysis could also be used to determine the correct enhancement approach(s). Contour offsetting capability must be developed to allow compensation for the beam spot size. The current implementation displays the actual slice contours without beam compensation offset. Research must also be conducted to determine just what constitutes

the most 'intelligent' scanning approach and whether it can be generated at reasonable speeds. Support for multiple solids must be provided. This can be done with an external solid modeling engine or additional capability can be added to the slicer. If slicer addition is selected, two options exist. The solids can be combined in three-space to form one facet model, or this step can be performed on the contours resulting from multiple solids.

Acknowledgements

This research was supported by NSF Grant DDM-8914212 as a subcontract through the University of Texas Solid Freeform Fabrication program, and other grants of the Rensselaer Design Research Center (RDRC) Industrial Associates Program. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation or any of the industrial sponsors.

We would like to thank Dick Aubin, Pratt & Whitney (a division of United Technologies), for providing a number of industrial STL models. Research by James Miller, RDRC, generated the facet model of the tooth nerve cavity from CT scan data provided by William Lorensen, General Electric Corporate Research and Development. A special thanks to James Miller for all the valuable comments and ideas on early drafts of this paper.

REFERENCES

1. D.L.Bourell, J.J.Beaman, H.L.Marcus, J.W.Barlow, "Solid Freeform Fabrication An Advanced Manufacturing Approach," Solid Freeform Fabrication Symposium Proceedings, University of Texas at Austin, August 1990.

2. Carl R. Deckard, "Part Generation by Layerwise Selective Sintering," M.S. Thesis, University of Texas at Austin, 1986.

3. "DE Series Digital Electronics", General Scanning Inc., GSI30260, 1989.

4. "Stereolithography Interface Specification," 3D Systems, Inc., June 1988.

5. Stephen J. Rock and Michael J. Wozny, "A Flexible File Format for Solid Freeform Fabrication," to appear in Soild Freeform Fabrication Symposium Proceedings, University of Texas at Austin, August 1991.

6. Wei-Ren Chang, "CAD/CAM for the Selective Laser Sintering Process," M.S. Thesis, University of Texas at Austin, 1989.

7. Stephen J. Rock and Michael J. Wozny, "The Ray-casting Technique for SLS Scan Vector Generation," Rensselaer Design Research Center Technical Report, TR-91019, 1991.

8. James V. Miller, "On GDM's: Geometrically Deformed Models for the Extraction of Closed Shapes from Volume Data," M.S. Thesis, Rensselaer Polytechnic Institute, Dec. 1990.