Direct Slicing of STEP Based NURBS Models for Solid Freeform Fabrication

B. Starly, A. Lau, W. Sun
Dept of Mechanical Engineering and Mechanics, Drexel University
Philadelphia, PA 19104

and

W. Lau, T. Bradbury, A. Youssef, C. Gaylo
Therics, Inc. 115 Campus Drive
Princeton, NJ 08540
Reviewed, accepted August 13, 2003

Abstract
Direct slicing of CAD models to generate process planning instructions for solid freeform fabrication may overcome inherent disadvantages of using STL format in terms of the process accuracy, ease of file management, and incorporation of multiple materials. This paper will present the results of our development of a direct slicing algorithm for layered freeform fabrication. The direct slicing algorithm was based on a neutral, international standard (ISO 10303) STEP-formatted NURBS geometric representation and is intended to be independent of any commercial CAD software. The following aspects of the development effort will be presented: 1) Determination of optimal build direction based upon STEP-based NURBS models; 2) Adaptive subdivision of NURBS data for geometric refinement; and 3) Ray-casting slice generation into sets of raster patterns. Feasibility studies applying the direct slicing algorithm to example models and the generation of fabrication planning instructions involving multi-material structures will also be presented.

Keywords: Direct slicing, STEP, NURBS, free form shapes, adaptive subdivision.

---

1. Introduction

Layered **M**anufacturing (LM) or **S**olid **F**reeform **F**abrication (SFF), a new class of manufacturing techniques introduced during the mid 1980s have grown rapidly over the past decade because of its proven ability to reduce product development cycle time. In the current industrial **R**apid **P**rototyping (RP) practice, the 3D CAD data are first converted to an intermediate **ST**ereo**L**ithography (STL) format, a tessellation procedure where the model is approximated by triangles, sliced and then fabricated by the machine. But with the rapid growth of the RP industry, particularly RP towards rapid manufacturing, rapid tooling and biomedical applications, there has been a growing dissatisfaction with this format among the RP community due to the limitations inherent within the format. The geometric description used to represent solid CAD objects significantly affects the accuracy and quality of the final parts produced with this technology especially in the case of freeform shapes. The limitation of the current STL format generated through a tessellation procedure of the CAD model can be summarized as follows [1]:

Tessellation involves approximation of surfaces with triangular facets which is undesirable in general, particularly when we are dealing with models that contain freeform shapes.
As model precision demands become more stringent, the number of facets required to adequately approximate these freeform shapes will increase. This would account for huge STL file sizes, and thus increasing the chances of errors.
Many CAD systems fail to generate valid model tessellations and often involve manual fixation of errors raising the need to improve tessellation procedures. However, this calls for implementing robust and efficient procedures which may be difficult and would be computationally expensive to implement.
STL format fails to include other design content within the model such as topology, internal material variations, multi-material regions.

Manufacturers of free form surface models are in need of something other than the currently available STL format. Direct Slicing of CAD models without the intermediate STL format seem to be a promising approach in this direction. Direct slicing of the solid model keeps the geometric and topological robustness from the original data. Its advantages include greater model accuracy, pre-processing time reduction, elimination of checking and repairing routines, and file size reduction [2]. Since the direct mathematical formulation of the surface is used, the full data of the original solid modeler is therefore available and the loss that occurs during tessellation is avoided. Both direct slicing and adaptive direct slicing improve the accuracy and surface quality of the final RP parts. The approach also eliminates the verification and repairing processes, decreases human intervention, increases the robustness of data transfer, slicing and other preprocessing parameters. With this approach of direct input from computer aided design models, CAD and RP vendors would be able to sell their software/machines as a fully integrated CAD/CAM RP system.

Accuracy was not an issue that was addressed upon in the early days of RP simply because the parts could only be used for prototyping and design verification purposes. A wealth of research is available in published literature [3-5] regarding STL slicing and the different methods through which it can be achieved. However, improving the accuracy of an RP part has become the focus of the RP community under the increasing need for prototyping functional parts with engineering properties and dimensional tolerances comparable to conventionally produced parts. One among the first to try out direct slicing was Rajagopalan et al [6], where they have directly sliced the Non-Uniform Rational B-spline Surfaces (NURBS) in an I-DEAS based CAD system. The process relied on I-DEAS to perform the slicing which made it package-specific. With regards to improving surface finish, variable thickness slicing methods (adaptive slicing) for handling peaks, flat areas and staircase effect have also been proposed by Dolenc and Makela [7]. Jamieson and Hacker [2] developed a direct slicing algorithm based on the Unigraphics slice modules which directly sliced the model using first constant layer thickness. Consecutive contours were compared and if the difference was small, they were accepted. If not, a middle slice is created and the process of comparison performed again. The procedure is repeated until the difference between any two consecutive slice contours is either small or the minimum layer thickness has been reached. It has been recognized that in order to improve the surface quality of a part built by an LM technique, it is necessary to minimize the staircase effect which is inherent in all LM techniques. In a more recent study, Weiyin et al [8] developed an adaptive direct slicing algorithm that operates directly on NURBS based models to generate skin contours and then uses a selective hatching strategy to reduce the build time of the model. Almost all of the papers published above in some way depend on external modeling packages to

perform the slicing which in turn limits the capability on the level of control and variety that can be achieved. A STEP-based transfer of model data to a process planning system has been made use of by Gilman et al [9] using commercially available software. A Boundary Representation (B-rep) solid model of a part is translated into STEP format for transfer of design data to the process planning software. LM data is generated using faceted boundary representation and then transferred to the RP machine for prototyping. It has been recommended that the development of STEP specifically for the purpose of LM process planning due to its flexibility in terms of representing 3D CAD data as well as 2D contour slices thereby simplifying the standardization procedure with one common platform [1].

The objective of this paper is to present a method on a direct slicing approach that is independent of any CAD modeling package and will make use of STEP as the starting input file of the model to be prototyped. We have currently focused on NURBS based freeform shapes to demonstrate the capability of the algorithm and the proposed methodology with regards to raster line pattern layout. The following section will briefly detail our proposed methodology and some of the key steps in the process of direct slicing. This is followed by a review of the NURBS equations that need to be numerically solved to obtain the slice ray patterns during the slicing operation. The key steps of optimal orientation, adaptive refinement, root finding and evaluation of points are detailed. Section 5 gives several case studies example models that have been sliced to showcase the proposed method. The paper concludes with a summary and future research initiatives.

2. Direct Slicing of NURBS using Ray Casting

Non-Uniform Rational B-Spline (NURBS) are the industry standard tools for the representation and computer aided-design of freeform models [10] in the field of automotive design, ship design etc. Central to the problem of slicing NURBS surfaces is the determination of intersection points between the slicing plane and the model. This is also a problem that is researched upon by the computer graphics community with regards to Ray-tracing of NURBS surfaces [11-14]. Ray tracing of free form surfaces determines the visible parts by constructing rays from the viewpoint through the pixels of the image plane into the scene. The rays are intersected with the surfaces of the scene and the first surface hit determines the color of the pixel. If the ray misses all objects, the corresponding pixel is shaded with the background color. Ray tracing handles shadows, multiple specular reflections, and texture mapping in a very easy straight-forward manner. This is important to the designer since it would help him assess the surface quality and texture and hence help in a better design process. The basic approaches in most ray-tracing algorithms with regard to determination of the intersection points remain the same and only vary with regards to efficiency in terms of memory usage and the speed taken to ray trace a particular scene. Most ray tracing algorithms have performed while tessellating freeform shapes to gain speed and reduce complexity. However direct intersection using the exact mathematical equations has also been performed [14]. In our proposed methodology, we have used the same basic approach using the exact mathematical equations to deal with finding the intersection points which will be detailed in the following sections. We have however extended it to the slicing domain for use in the LM manufacturing scenario. The difference lies in the fact that in LM, intersection points both in the form of entry and exit points need to be determined along with the vector layout pattern of the rays within the model. Rather than a tracing operation, a ray casting method is performed in order to perform the slicing procedure.

Within STEP files, solid and surface models may be represented as rational or non-uniform rational B-spline surfaces. Unlike STL files where the facet information of the triangle is used to obtain the slice contour, direct slicing works on using the exact mathematic representation of the freeform shapes in computing the slice contours or tool patterns. A rational B-spline surface (10) is expressed parametrically in the form,

$$S(u,v) = \frac{\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} W_{ij} P_{ij} b_{ik}(u) b_{jl}(v)}{\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} W_{ij} b_{ik}(u) b_{jl}(v)} \tag{1}$$

where parameters u and v range from zero to one, n and m the degree of the surface in u and v direction. The P_{ij} terms are 3D net control points of the control polygon and W_{ij} terms their corresponding weights, b_{ik} and b_{jl} are B-spline basis functions of order k and l respectively. The B-spline basis functions are defined by the Cox-deBoor recursion formulas as given by:

$$b_{j1}(s) = \begin{cases} 1 & \text{if } ku_j \le u \le ku_{j+1} \\ 0 & \text{otherwise} \end{cases} \qquad b_{l1}(s) = \begin{cases} 1 & \text{if } kv_l \le v \le kv_{l+1} \\ 0 & \text{otherwise} \end{cases}$$

and

$$b_{jk}(u) = \frac{(u - ku_j) b_{(j)(k-1)}(u)}{ku_{j+k-1} - ku_j} + \frac{(ku_{j+k+1} - u) b_{(j+1)(k-1)}(u)}{ku_{j+k+1} - ku_{j+1}}$$
$$b_{lk}(v) = \frac{(v - kv_l) b_{(l)(k-1)}(v)}{kv_{l+k-1} - kv_l} + \frac{(kv_{l+k+1} - v) b_{(l+1)(k-1)}(v)}{kv_{l+k+1} - kv_{l+1}} \tag{2}$$

where the values of ku_j and kv_l are defined by the knot vector associated with the NURBS surface in the u and v direction respectively. The STEP file contains all information that is required to define the NURBS uniquely and a STEP reader may be employed to extract the relevant information. For further discussion on B-splines and their properties, the author refers them to [12].

The core aspect of our approach is outlined in Figure 1. The NURBS surface is geometrically refined using an adaptive subdivision procedure to break them down into smaller domains of parametric values. Bounding boxes are then used to cover up the entire surface based on these smaller domains. The rays shoot out intersecting the model at several boxes. For a box that is hit, a root finding procedure is initiated to converge at the intersection point. The procedure is repeated for all rays that are cast onto the slice plane and for every slice plane that intersects the NURBS model.

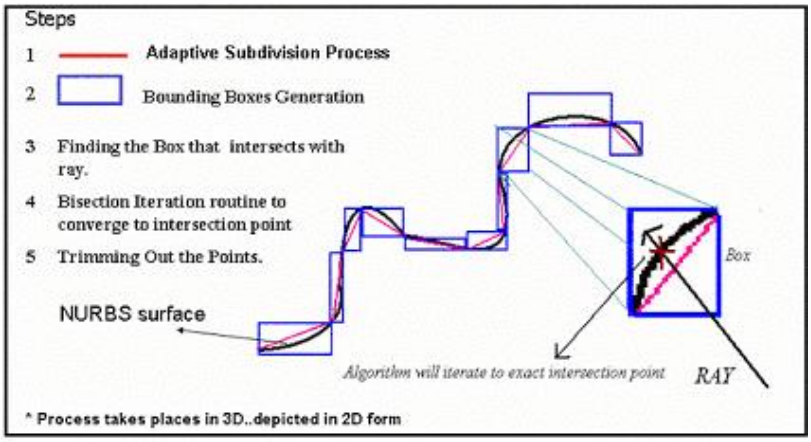


Figure 1: Step by Step procedure on calculation of NURBS intersection points

3. Ray Casting of NURBS surfaces

In the ray casting approach, the bounding box of the model is first determined allowing the start position of the ray to be defined from any predefined corner of the box which then shoots out across intersecting the model. A ray is defined as having an origin 'o' and a unit direction vector 'd' and can be defined as:

$$r(s) = \bar{o} + s * \bar{d} \tag{3}$$

Using the method followed by Kajiya [11], the ray r(s) can be rewritten as an intersection between 2 planes given by $\{p|\ P_1.(p,1) = 0\}$ and $\{p|\ P_2.(p,1) = 0\}$ where $P_1=(N_1,d_1)$ and $P_2=(N_2,d_2)$. The normal to the first plane is defined as

$$N_1 = \begin{cases} (d_y, -d_x, 0) & \text{if } |d_x| > |d_y| \text{ and } |d_x| > |d_z| \\ (0, d_z, -d_y) & \text{otherwise} \end{cases} \tag{4}$$

N_2 will always be perpendicular to the ray direction and the plane N_1 , hence

$$\overline{N_2} = \overline{N_1} x \bar{d} \tag{5}$$

Since both planes contain the origin '$\bar{o}$', it can be deduced that P1.(o,1) = P2.(o,1) = 0. Thus,

$$\begin{aligned} d_1 &= - \overline{N_1} \,.\, \bar{o} \\ d_2 &= - \overline{N_2} \,.\, \bar{o} \end{aligned} \tag{6}$$

An intersection point that needs to be calculated should satisfy the following two conditions,

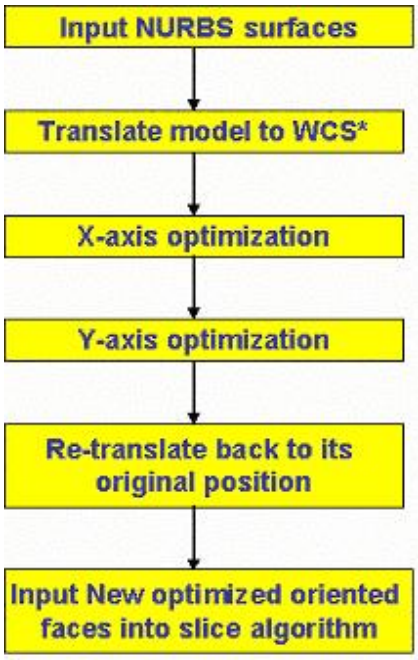
$$\begin{aligned} \overline{P_1} .(S(u,v),1) &= 0 \\ \overline{P_2} .(S(u,v),1) &= 0 \end{aligned} \tag{7}$$

The above equation needs to be solved using numerical means and we have employed the Bisection Iteration routine to determine the values of u and v that will satisfy (7). However before the root finding operation begins, a number of pre-processing steps are performed. In the steps that follow,

we explain the details regarding optimal orientation, refinement using adaptive subdivision, generation of the bounding volumes, root finding, evaluation and identification of output points.

Step 1: Orientation

The model orientation within the fabrication bed affects the build time, part strength and surface finish. Thus before the part is sliced, a minimization of certain objective criteria specified by the designer will be done to find the optimal orientation for slicing the model. A number of orientation schemes have been devised. Some base their orientation with the largest convex hull of the object as the base [15], while some others orient the part based on certain critical features of the model [16]. We have used a scheme in which the model is incrementally oriented about user specified axes to obtain the least possible build height dimension. Given a set of n NURBS surfaces S(u,v) that is enclosed in a bounding box of height H (a function of orientation angle "_"), the objective function can be mathematically expressed as:



$$\text{Min } Z = H(\_) \quad \text{subject to } 0 < \_ < 360 \qquad (8)$$

where Z is the build height of the model and _ is the orientation of the model with respect to the object coordinate axes. The optimally oriented NURBS faces then act as the input to the second phase. The algorithm steps are as shown in Figure2.

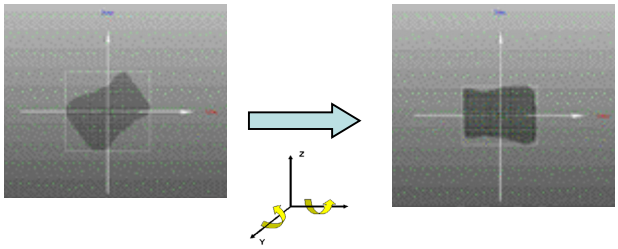


Figure 2: Optimization Algorithm

Step 2: Refinement

Refinement or subdivision of the NURBS surface is the addition of more control points to a surface without changing its process. This process is implemented using the Oslo Algorithm [17]. The basic idea would be to take in the original set of knot vectors that make up the surface and add new knot values into them creating more number of control points corresponding to the new knot vectors. If the addition of the new knot values at the same parametric value where the number added is equal to the order of the curve, then the two new surfaces created will have the same shape as the original unrefined surface. They would each

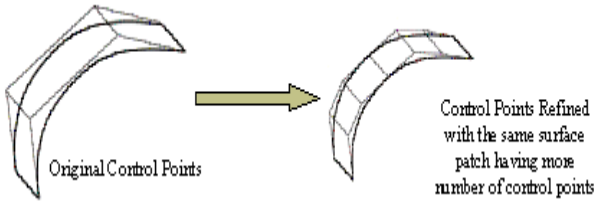


Figure 3: Refined Process of NURBS using the Oslo Algorithm

have a set of control points at the region where they join. The procedure is illustrated in Fig 3. This is done for the following two reasons. Numerical methods work better and faster when the parametric domain is smaller and no multiple value roots exists within the patch. Secondly, by refining the mesh, the various sub-patches which are essentially NURBS by themselves can be enclosed using bounding volumes thereby enabling the slice algorithm to determine which sub-patch contains the actual root. This stems from the fact that it is easier to determine the collision of rays with primitive bounding volumes than NURBS surfaces. Hence by identifying the sub-patch in which the solution exists, the domain in which the numeric solver has to work is limited and hence results in better chances of finding the roots.

The adaptive subdivision of the NURBS surface continues as long as a subdivision or flatness criteria is met. Regions that have more curvature are subdivided more than regions that are more or less flat and hence the name adaptive subdivision. Each new sub-patch contains all information that defines the NURBS and an appropriate ID is given to it. The refinement procedure is extensively used in the tessellation of parametric surfaces and has been studied extensively by a number of researchers. However, our main criteria in refinement of the mesh opposed to the tessellation procedure is not to ensure accuracy in representation but more in guaranteeing that the convergence occurs within the refined sub-patch. In this regard, selection of the subdivision factor is an important step, and an appropriate value controls to a great extent the success of accurate slicing.

Step 3: Boundary Volume Data structure Generation

As the refinement procedure of NURBS progresses, the boundary volume data structure gets filled up; sub-patches are stored in the data structure along with a unique id. The main idea

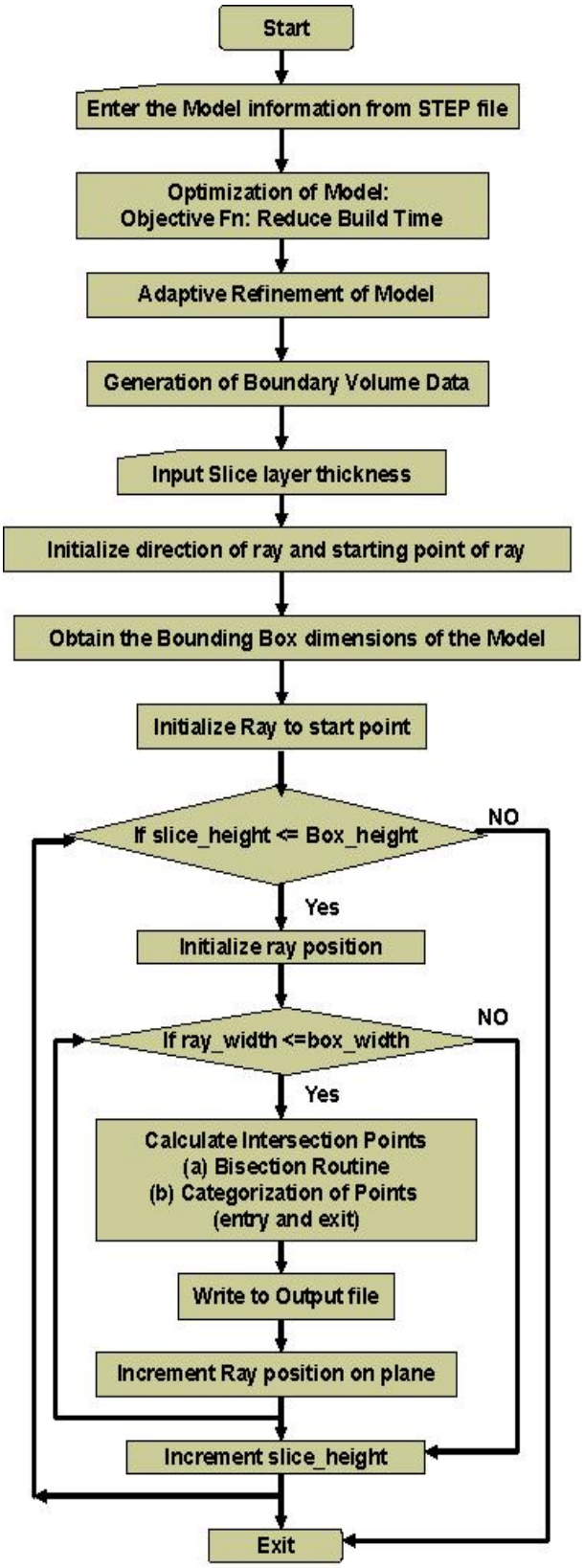


Figure 4: General Slicing Algorithm

behind the storage of these sub-patches is the generation of the boundary volume. Boundary volumes are usually primitives that enclose the sub-patch completely. Some of the candidate primitives that can be used are oriented boxes, spheres and parallelepipeds. The selection of a primitive depends on the tightness of fit and speed of intersection calculation with the ray. We have used in our approach axis aligned boxes that are oriented with the main coordinate axis of the model. Once they are axis aligned, the process of creating the boundary volumes is easier, and a ray-box intersection will involve less computation and hence speed up the process.

The bounding box is created using the control net points created from the refinement stage of the process. An important aspect to note is that the control mesh of a NURBS patch will always enclose the surface and therefore any convex shape surrounding the control net would also enclose the surface. Figure 1 gives an illustration in 2D on how the boxes get generated. Failure to enclose the sub-patch surface completely results in ray hits being missed and hence gaps in the sliced model.

Step 4: Numerical Solution - Bisection Iteration Routine

There are a variety of numerical methods that are available to solve for the intersection points. Among them include bisection algorithm, linear interpolation, Newton Iteration, and fixed point iteration [18]. Although the latter two methods are fast, in some cases they do result in solution divergence rather than convergence. Our problem reduces to finding the value (u*, v*) that corresponds to the intersection point of the ray and the NURBS surface. Though ray-tracing methods in model rendering employ Newton's iteration since it proves to be a faster process, we have used the bisection iteration routine mainly due to its simplicity and robustness. Time is not a critical factor in slicing as opposed to obtaining the roots of the equation.

The surface sub-patch contained within the boxes that were hit by the ray are retrieved from the boundary volume data structure and passed onto the solver method. The routine works with iterating towards the solution of one variable. However, we need to solve for 2 variables (u, v) that satisfy the equation. This is achieved by keeping one variable constant and iterating towards the best value of the second variable that satisfies the equation. If the error in the points generated does not satisfy a tolerance, _, the first variable is incremented by a pre-defined amount and the procedure repeated. The process continues until either a solution is found or the limit of the first variable is achieved.

We use two criteria to decide when to terminate the Bisection iteration routine. The first condition is our success criteria: if we are closer to the desired point by some determined tolerance, _, given by:

$$| F (u , v)| < \_ \quad (9)$$

we then report a successful hit. The value of _ determines two aspects, first, the accuracy of our results and second, success in reporting a hit. A tight value might result in the routine reporting a miss and on the other hand a bigger tolerance will result in intersection point offset errors. The second condition is the failure criterion, meaning that if this condition is met, the routine exits reporting a miss. If during the routine, the error calculated is approximately same as the error in the previous iteration, the iteration exits reporting a miss provided the success criteria has not been met. This is mathematically written as

$$F_n(u,v) = F_{n-1}(u,v) \quad \text{subject to } F_n(u,v) > \_ \tag{10}$$

where n is the n^{th} step of iteration

Step 5: Categorization of Intersection Points (Entry or Exit)

Once the intersection points are found out for every ray that is cast onto the slice planes, these points need to be classified as an entry or an exit point. We have done this by calculating the normal of the surface at the point under consideration. The normal vector can be calculated by the cross product of the tangent vectors in the u and v direction evaluated at the point given respectively by equations (11) and (12) as

$$T_u = \frac{\delta}{\delta u} S(u,v) \tag{11}$$

$$T_v = \frac{\delta}{\delta v} S(u,v) \tag{12}$$

The normal vector at this point is then given by $\overline{N} = T_u \times T_v$ (13)

If it is assumed that the ray shoots across in the y direction, then a point is classified to be as: an entry point If $N_y < 0$; an exit point if $N_y > 0$; and an edge point if $N_y=0$. The points once classified are stored in a predefined format to be displayed on screen and for conversion to machine instructions.

4. Implementation and Case Study examples

The algorithms were implemented in C++ and tested for a variety of model shapes that included simple NURBS surfaces as well as complex curved ones. Figure 4 gives the generalized algorithm for ray-casting of NURBS surfaces that includes all of the steps outlined above. Three of the test example cases are as shown below in Figures 6 to 8.

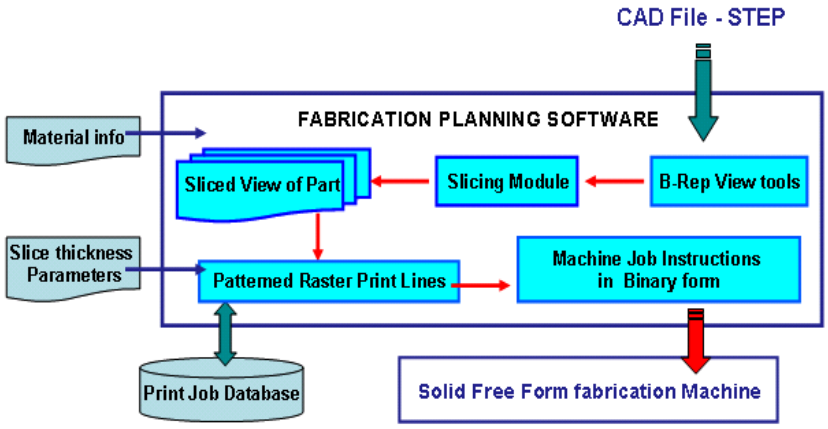
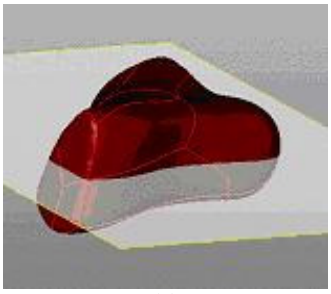


Figure 5: Process Flow during Fabrication

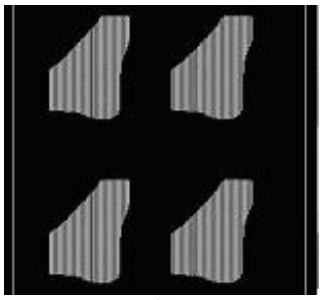
All the models were sliced at a constant layer thickness and ray offsets on the slice plane. Figure 5 details the data process planning steps right from the CAD file input stage to that of the final fabrication stage. For heterogeneous models, intersections of a ray to each geometric body are

calculated in sequence. Thus the material associated with the geometric body being sliced can be captured and stored in the slicing data structure.

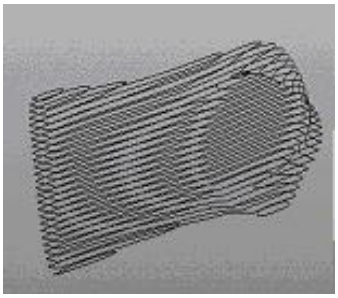
Model 1: This model is relatively small with about 16 NURBS surface definitions and was extracted from the STEP input CAD file using a STEP reader. A subdivision factor of 0.01 and a tolerance value of 0.01 were used for adaptive refinement and root finding respectively. Figure 6(a)



(a)

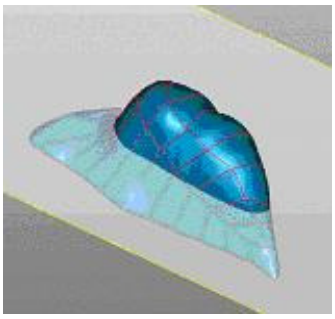


(b)



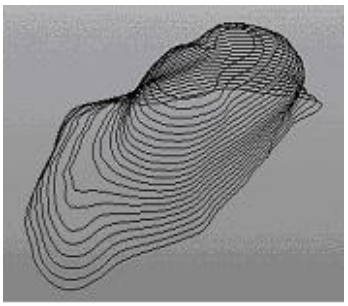
(c)

Figure 6(a): Case Study Example Model 1 with 16 NURBS surfaces.
Figure 6(b): Slice Raster Lines at a slice layer.
Figure 6(c) : Entry and Exit Points displayed



(a)

(b)

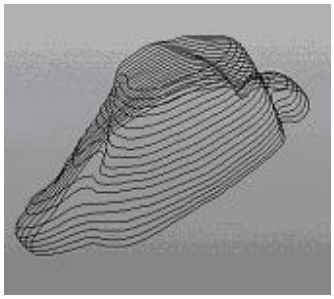


(c)

Figure 7(a): Case Study Example Model 2 with 102 NURBS surfaces.
Figure 7(b): Slice Raster Lines at a slice layer.
Figure 7(c): Entry and Exit Points displayed.

(a)

(b)



(c)

Figure 8(a): Model 3 Layout – 135 NURBS surfaces.
Figure 8(b): Sliced View layout
Figure 8(c) : Sliced entry and exit points displayed

shows the model optimally oriented with the least height in the z-direction. Figure 6(b) shows a slice layout on the fabrication bed at the slice position defined by the slice plane as shown in Figure 6(a).

Model 2: The second model contained 102 NURBS surface definitions, extracted through the same method using the reader and the data transferred to the algorithm. In this case a subdivision factor of 0.1 was sufficient for appropriate slicing with no perceivable slice errors during part layout. The same tolerance value of 0.01 was used for the root finding routine. Figure 7(a) shows the model optimally oriented and Figure 7(b) illustrating the sliced model part layout.

Model 3: The third model is more complex not only in terms of having more number of NURBS surface but also in terms of its overall shape. This model depicts the capability of the algorithm to handle multiple exit, entry points as well as accurate slicing at edges of the model. A subdivision factor of 0.1 and a tolerance value of 0.01 was used for the process. Figure 8(a) illustrates the model and Figure 8(b) the corresponding slice layout on the fabrication bed. It can be seen that the model was sliced appropriately with key features detected at the slice plane shown.

5. Conclusion

Direct Slicing of CAD model promises to offset the disadvantages posed by the STL format. It does not involve any file repair routines and file sizes are much smaller to handle. It also facilitates the possibility of slicing multi-material volumes or heterogeneous models, a definite advantage over STL files. Since the exact representation is used, complete design information is carried over to the fabrication stage with no loss in information. Although direct slicing does take longer to slice, this problem can be offset by efficient algorithms in terms of reducing memory usage and faster computing power. A point to note is also a careful selection of the subdivision factor and the tolerance value. We have found that under continued trials, an intuition is developed on the selection of the right parameters which works best for the models. Future research would involve the development of an adaptive direct slicing method on heterogeneous models and an appropriate slice layer format in terms of STEP that would be able to transfer slice information across different RP platforms. Effort would also be put into quantifying the exact accuracy obtained by direct slicing rather than using the STL format.

6. References

1. Anne L. Marsan, Vinod Kumar, Debasish Dutta, and Michael J. Pratt, "*An Assessment of Data Requirements and Data Transfer Formats for Layered Manufacturing"*, Technical Report NISTIR 6216, National Institute of Standards and Technology, Gaithersburg, MD.
2. Jamieson, R., Hacker, H. "Direct slicing of CAD models for rapid prototyping" Rapid Prototyping Journal, Vol 1(2), 1995.
3. Luo R C.; and Y. W. Ma, "A Slicing Algorithm for Rapid Pototyping and Manufacturing", 1995 IEEE International Conf. on Robotics and Automation, Nagoya, Japan, May 1995,Vol (3); pp 2841 -2846,
4. Suh, Yong Seok and Wozny, Michael J. "Adaptive Slicing for Solid Freeform Fabrication Processes," Solid Freeform Fabrication Symposium 1993, H. L. Marcus et al., eds. University of Texas, Austin, August 1993, pp. 404-410.
5. Tait S, Smith., Rida T. Farouki , Mohammed Al- Khandari, "Optimal Slicing of free form surfaces", Computer Aided Geometric Design(19), 2002, 43-64
6. Rajagopalan, M., Aziz, N.M., Huey, C.O. "A model for interfacing geometric modeling data with rapid prototyping systems", Advances in Engineering Software, Vol 23, 1995; pp 89-96.

7. Dolenc and I. Mäkelä, "Slicing procedures for layered manufacturing techniques," *Computer-Aided Design*, Vol. 26(2), 1994; pp. 119-126.
8. Weiyin M , Peiren M," An adaptive slicing and selective hatching strategy for layered manufacturing ", Journal of Materials Processing Technology; Vol 89, 1999; pp 191-197
9. Gilman, Charles R. and Rock, Stephen J. "The Use of STEP to Integrate Design and Solid Freeform Fabrication," Solid Freeform Fabrication Symposium, University of Texas, Austin, August 1995.
10. Rogers D F, "Introduction to NURBS: with Historical Perspective", Morgan Kaufmann Press, 2000, ISBN 1558606696
11. Kajiya J T, "Ray tracing parametric patches", Computer Graphics SIGGRAPH '82 Proceedings, Vol 16(3), 1982; pp 245-254.
12. Daniel L, Jacob Gonczarowski, "Improved techniques for ray tracing parametric surfaces", The visual computer, Vol 6(3),1990; pp 134-152.
13. Michael AJ Sweeney, Richard Bartels, " Ray tracing free form B-spline surfaces", IEEE Computer Graphics & Applications, Vol 6(2),1986;
14. William M, Elaine C, Russell F, Peter S, "Practical Ray tracing of Trimmed NURBS surfaces", Journal of Graphics Tools, Vol 5(1), 2000; pp 27-52.
15. Sreeram, Puduhai N. and Dutta, D. "Determination of Optimal Orientation Based on Variable Slicing Thickness in Layered Manufacturing," Proceedings of the ASME Winter Annual Conference, San Francisco, CA; Nov. 1995.
16. Frank, Dietmar and Fadel, Georges. "Preferred Direction of Build for Rapid Prototyping Processes," Proceedings of the Fifth International Conference on Rapid Prototyping, R. P. Chartoff, A. J. Lightman, and J. A. Schenk, eds. University of Dayton, June 1994; pp. 191-200.
17. Bartels R, John B and Brian B, " An introduction to splines for Use in computer graphics and geometric modeling", Morgan Kaufmann Press, Los Altos, CA, 1987; ISBN 1558604006
18. William H P, Saul A, Wm T, Brian P, "Numerical recipes in C : the art of scientific computing", Cambridge University Press; 2nd edition,1993, ISBN 0-521-43108-5