ELSEVIER

Computer-Aided Design 37 (2005) 387–397

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

Direct slicing of STEP based NURBS models for layered manufacturing

B. Starly[a], A. Lau[a], W. Sun[a,*], W. Lau[b], T. Bradbury[b]

[a]*Department of Mechanical Engineering and Mechanics, Drexel University, Philadelphia, PA 19104, USA*
[b]*Therics, Inc., 115 Campus Drive, Princeton, NJ 08540, USA*

Received 19 November 2003; received in revised form 4 June 2004; accepted 26 June 2004

Abstract

Direct slicing of CAD models to generate process planning instructions for solid freeform fabrication may overcome inherent disadvantages of using stereolithography 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 non-uniform rational B-spline (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. The development also provides for multi-material slicing and will provide an effective tool in heterogeneous slicing processes.

Keywords: Direct slicing; Solid freeform fabrication; Layered manufacturing; Biomodeling; Medical rapid prototyping

1. Introduction

The growth of solid freeform fabrication (SFF) or layered manufacturing (LM) in biomedical applications has created the need for better technology in terms of model representation, accuracy and functionality [1]. SFF models in medicine have been used mainly for assisting diagnosis, surgical planning, and manufacture of orthopedic implants. However, the technology has recently seen applications in the field of tissue engineering specifically in the fabrication of tissue scaffold structures. Traditionally, LM process involves the conversion of the 3D models into STereo-Lithography (.STL) format, a tessellation procedure where the model is approximated by triangles, sliced and then fabricated by the machine. The procedure worked well in the manufacturing industry which used RP to evaluate product designs and aesthetics by producing actual prototypes of the proposed design. However, the geometric description used to represent solid CAD objects significantly affects the accuracy and quality of the final parts produced especially in the case of freeform shapes. This is particularly relevant in the field of computer aided tissue engineering [2] which involves the use of these freeform shapes to model the external shape of the scaffold that intends to replace actual body sections. The complexity of the internal and external shape of these scaffolds [3,4] requires a more direct method in conversion of CAD data to process planning instructions. The process of tessellation and representation in STL format is inadequate for designing biomimetic scaffolds with complex internal architecture. This is due to the large number of triangles required to represent small features in the scaffold which results in failure during the conversion process. Direct slicing of CAD models without the intermediate .STL format is preferred because it helps keep the geometric and topological robustness that the original data have and no intermediate conversion process is required. This approach has advantages over the indirect slicing method which include greater model accuracy, pre-processing time reduction, checking and elimination of repair routines, file size reduction and more importantly support for multi material slicing.

* Corresponding author. Tel.: +1-215-895-5810; fax: +1-215-895-2094.
E-mail address: sunwei@drexel.edu (W. Sun).

0010-4485//$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2004.06.014

The field of direct slicing for LM has been researched on for some time. Some of the earliest works were applied to traditional industrial applications and did not really focus on medical prototyping applications. Jamieson and Hacker [5] developed an adaptive direct slicing algorithm based on the unigraphics based slice modules which directly sliced models using a 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. Rajagopalan et al. [6], have directly sliced non-uniform rational B-spline (NURBS) based models in an I-DEAS based CAD system. The process relied on I-DEAS system functions to perform the slicing which made it package-specific. In a more recent study, Ma and Peiren [7] 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. All of the papers published above depend on external modeling packages to perform the slicing process which in turn limits the capability on the level of control and variety that can be achieved. Zhao and Laperriere [8] in their work worked on an adaptive direct slicing approach independent of CAD vendors in which they make use of AUTOCAD API functions to perform the slicing operation. Using API functions does provide more control but they have showed inappropriate slicing when applied to scaffolds with complex external shapes. Fig. 1 shows failure at a particular z-level during slicing of a sinus implant using the slice functions available from a major CAD vendor.

Therefore, the objective of this paper is to present a direct slicing approach that is independent of any CAD modeling package and that which makes use of STEP as the starting input file of the model to be prototyped. The approach described here is feasible for both traditional manufacturing designs as well as for biomedical applications. 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 suited for the 3DP™ machines. However, the work can be extended to include contour slice patterns along with a hatching strategy depending on the intended RP machine. A point that we would like to stress is that the proposed method in no way replaces the STL based method of indirect slicing which is more suited to traditional manufacturing components. The following method however works well for freeform shapes that have complex outer geometry which in its STL representation carries drawbacks as mentioned in the first paragraph. Since the design of scaffolds for tissue engineering applications involve complex outer geometrical shapes, the direct method of slicing would be more advantageous than the indirect method of slicing.

Fig. 1. API slice errors from a major CAD vendor.

The paper is organized as follows. The following section will briefly detail the generalized process of RP manufacturing in the clinical medicine and tissue engineering scenario. A brief description of the NURBS equation that we need to solve for is also described. The third section details the process of ray-casting and the steps taken to obtain the ray-raster patterns during slicing. The key steps of optimal orientation, adaptive refinement, root finding and evaluation of points are detailed. Section 4 gives several case study characteristic models that have been sliced to showcase the efficacy of the proposed method. The paper concludes with a summary and future research initiatives.

2. Overview of direct slicing using STEP based NURBS models

2.1. General biomodeling application process path

The first step involves the acquisition of CT/MRI images of the bone under replacement. These CT images are loaded in commercially available image processing software to reconstruct them to CAD models through reverse engineering techniques. The CAD model is now sliced to generate process plan instructions for the relevant RP machine. The machine prototypes the required scaffold for the desired application. Although tissue engineering applications processes would involve more stages until final fabrication, the steps remain the same and a similar process path needs to be followed. For further details regarding reverse engineering techniques and CAD based model reconstruction, the authors refer them to [9]. Our focus in this paper will be the conversion of CAD models to process planning instructions for the RP machine. Although the conventional method is to convert the CAD model to an STL based model, our effort is to directly slice the CAD model and transfer information to the machine eliminating the need for the intermediate file format. The overall process path for a general medical rapid prototype production in building the external shape of a scaffold is as shown in Fig. 2.

The process data flow is shown in Fig. 3 that is implemented within a Fabrication Planning software framework. Initially, the CAD STEP file is input into the software framework wherein a STEP reader is implemented to extract out the model features. The B-rep features of the model are viewed using graphic kernels for easier manipulation and display. The model features in its NURBS representation is then transferred to the slicing module. Each cross sectional layer is then extracted from the model based on the slicing parameters and then converted to machine job instructions. A print job database is also maintained for database records and future retrievals.

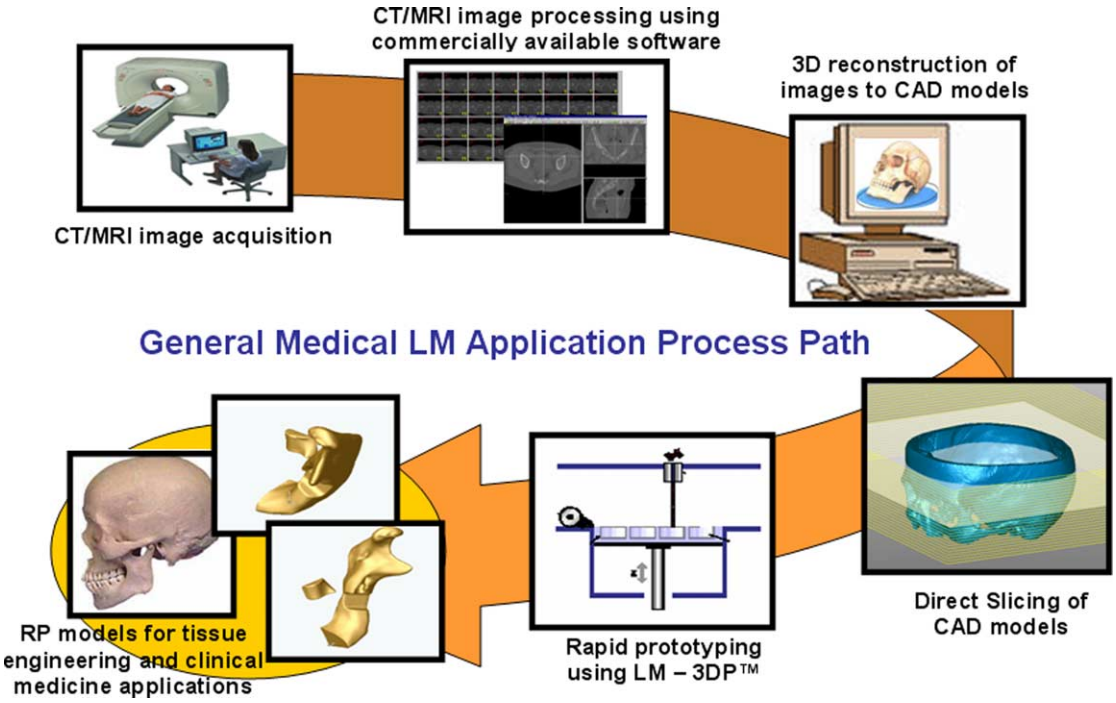

Fig. 2. General biomodeling and medical rapid prototyping process path.

2.2. ISO 10303(STEP) based NURBS representation

NURBS are the industry standard tools for the representation and computer aided-design of freeform models [10, 11] in the field of automotive design, ship design etc. 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 by using the exact mathematical representation of the freeform shapes in computing the slice contours or tool patterns. A rational B-spline surface is expressed parametrically in the form

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

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:

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

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

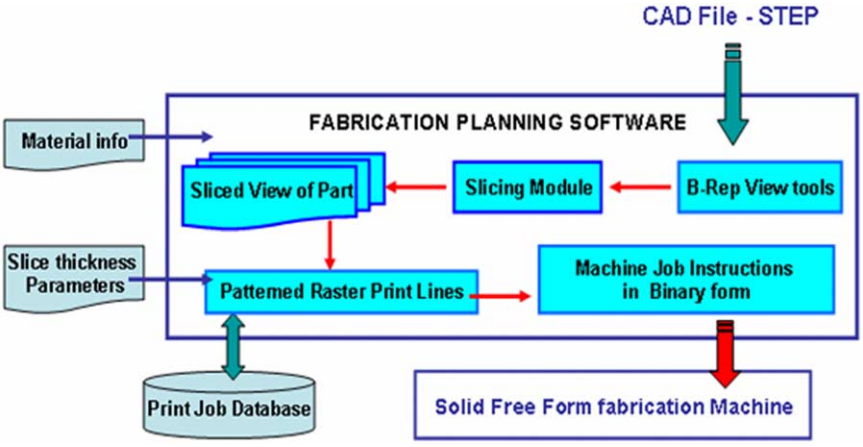

Fig. 3. Process flow during fabrication.

```
#Identifier ID =CARTESIAN_POINT('', (x-coordinate, y-coordinate, z-coordinate));
•.
•.
#Identifier ID =(BOUNDED_SURFACE()B_SPLINE_SURFACE (degree_u, degree_v, ((control points
identifier IDs ),( ), ..UNSPECIFIED.,.F.,.F.,.U.)B_SPLINE_SURFACE_WITH_KNOTS(( multiplicities in u),
(multiplicities in v), (knot vectors in u), ( knot vectors in v),. UNSPECIFIED.)
GEOMETRIC_REPRESENTATION_ITEM(), RATIONAL_B_SPLINE_SURFACE ((weights associated with
control points),(),...REPRESENTATION_ITEM('')SURFACE());
```

Fig. 4. STEP description of a uniform B-spline surface.

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. For further discussion on B-splines and their properties, the author refers to [10]. The STEP file contains all information that is required to define the NURBS uniquely and a STEP reader is employed to extract the relevant information. Fig. 4 depicts the format of a B-spline description in a ISO 10303 STEP file [12]. An ID is given to each control point as contained within the CARTESIAN_POINT statement. This is followed by the definition of the B-spline surface in terms of the parameters needed to define it—degree of surface in u and v direction, multiplicities and knot vectors in u and v direction and the weight associated with each control point.

3. Direct slicing of NURBS surfaces using ray-casting

Central to the problem of slicing NURBS surfaces is the determination of intersection points between the slicing plane and the model. This is also researched by the computer graphics community with regards to ray-tracing of NURBS surfaces [13–17]. Ray tracing of free form surfaces are commonly used in 3D visualization of models and real time rendering of images. The basic approaches in most ray-tracing algorithms in the 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 freeform ray tracing algorithms have performed while tessellating the models to gain speed and reduce complexity of solution. Several methods of obtaining ray-NURBS intersections have appeared in literature. Almost all of them involve some precursor method of divide and conquer before obtaining the roots of the intersection equation. Adaptive based subdivision [13–16] and curvature based subdivision processes [17] serve well in dividing the model space into sub-patches. These sub-patches are then enveloped by bounding volumes to restrict the parameter domain in which the solution needs to be found.

In our proposed methodology, we have used the same basic approach using the mathematical NURBS equations to solve for the intersection points. 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/raster layout pattern of the rays within the model. The raster pattern of the rays would involve connecting corresponding entry and exit points to define machine level instructions for LM. Vector patterning on the other hand would involve connecting entry and exit points in such a manner that they form closed loops within the layer. These vector paths are then converted to machine instructions for fabrication. The core aspect of our approach is outlined in Fig. 5. 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

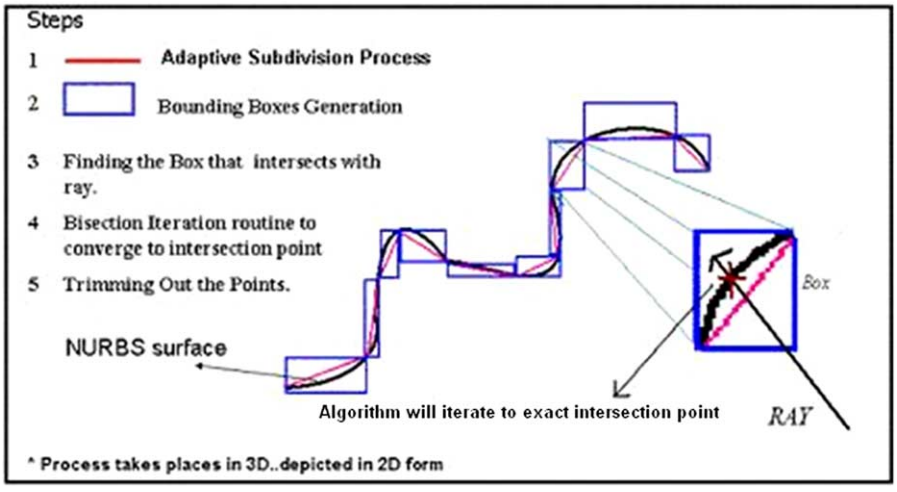

Fig. 5. Step by step procedure on calculation of NURBS intersection points.

up the entire surface based on these smaller domains. The rays then shoot out intersecting the model at several boxes and 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.

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:

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

Using the method followed by Kajiya [13], the ray $r(s)$ can be rewritten as an intersection between two 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

$$N_2 = N_1 \times \hat{d} \tag{5}$$

Since, both planes contain the origin 'o', it can be deduced that $P_1.(o,1)=P_2.(o,1)=0$. Thus,

$$d_1 = -N_1.\hat{o} \quad d_2 = -N_2.\hat{o} \tag{6}$$

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

$$P_1.(S(u^*, v^*), 1) = 0 \quad P_2.(S(u^*, v^*), 1) = 0 \tag{7}$$

The above equation needs to be solved using numerical means and we have employed the bisection iteration routine to determine the values u^* and v^* that will satisfy (7) from the function as expressed in (8):

$$F(u, v) = \begin{pmatrix} N_1.S(u, v) + d_1 \\ N_2.S(u, v) + d_2 \end{pmatrix} \tag{8}$$

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. The generalized slice algorithm is as shown in Fig. 6.

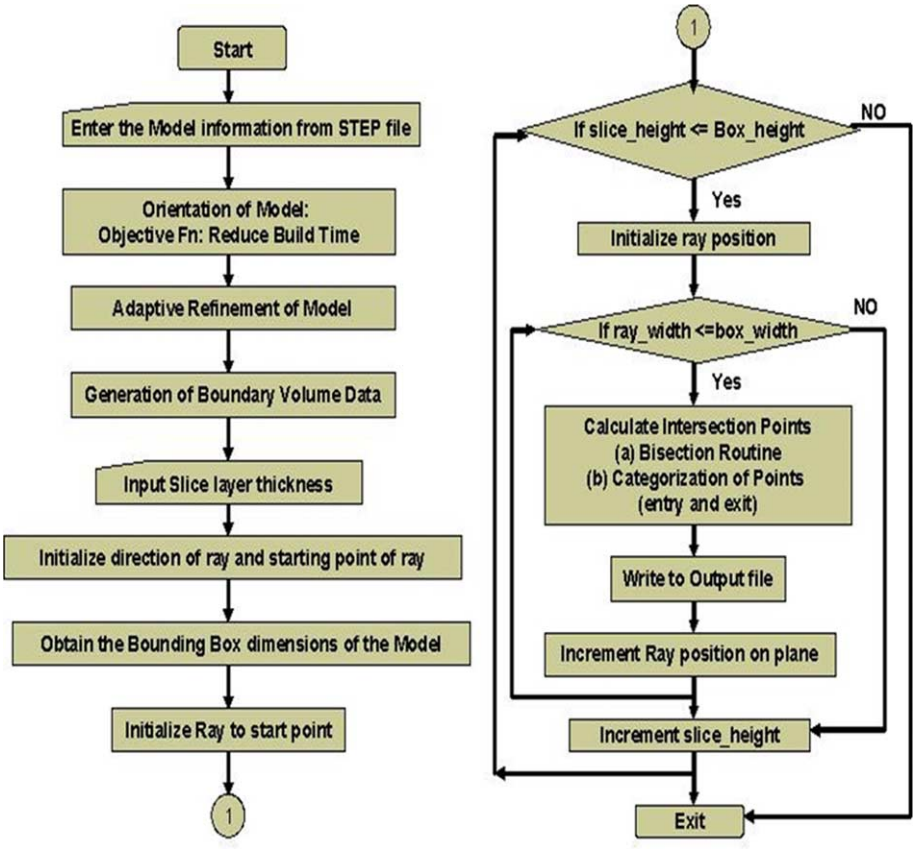

Fig. 6. General slice algorithm.

3.1. Step 1: orientation of CAD model to minimize build height

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 [18], while some others orient the part based on certain critical features of the model [19]. 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 $S_{i\ldots n}(u,v)$ that is enclosed in a bounding box of height H (a function of orientation angle 'θ'), the objective function can be mathematically expressed as:

$$\text{Min } Z = H(\theta) \quad \text{subject to } 0 < \theta < 360 \tag{9}$$

where Z is the build height of the model and θ 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.

3.2. Step 2: refinement of NURBS surfaces

Refinement or subdivision of the NURBS surface is the addition of more control points to a surface without changing its original shape. The process is implemented using the Oslo algorithm [20]. The basic idea is to take in the original set of knot vectors that make up the surface and add new knot values into them creating more control points corresponding to the new knot vectors. If the addition of the new knot values at the same parametric value 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 have a set of control points at the region where they join. The procedure is illustrated in Fig. 7. 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 (Fig. 7).

The adaptive subdivision of the NURBS surface continues as long as a subdivision or flatness criteria is met. Regions that have a higher degree of curvature are subdivided more than regions that have a lesser degree in curvature and hence the term 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 as 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.

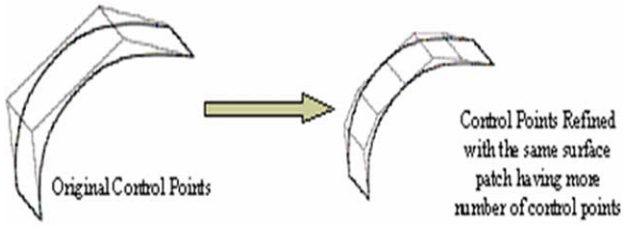

Fig. 7. Refined process of NURBS using the Oslo algorithm.

3.3. 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 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 [15,21–23]. 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 bounding volumes is easier and a ray-box intersection involves less computation and hence speeds 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. Fig. 4 gives an illustration in 2D on how the boxes get generated. Failure to enclose the sub-patch surface completely results in rays hits being missed out and hence gaps in the sliced model.

3.4. Step 4: numerical solution—bisection iteration routine

There are a variety of numerical methods that are available to solve for the intersection points. Among these are bisection algorithms, linear interpolation, Newton

iteration and fixed point iteration [24] or a combination of two algorithms. Although the latter two methods are fast, in some cases they do result in solution divergence rather than convergence. This depends on the actual initial guess to the polynomial equation. Our problem reduces to finding the value (u^*, v^*) that corresponds to the intersection point of the ray and the NURBS surface. Almost all root finding techniques employ a tolerance value within which the error bound is defined. Selection of one process over the other entirely depends on time and the particular equation that is intended to be solved [24]. Though ray tracing methods in graphic applications employ Newton's iteration or its variants since it proves to be a faster process, we have used the bisection iteration routine mainly due to its simplicity and robustness.

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 by iterating towards the solution of one variable. However, we need to solve for two 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, ε, the first variable is incremented by a pre-defined amount and the procedure repeated. The process continues until a solution is found, the limit of the first variable is achieved or a specified number of iterations have been completed.

We use three 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, ε, given by:

$$F(u_n, v_n) < \varepsilon \tag{10}$$

we then report a successful hit. The value of ε 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 due to satisfying the maximum iteration condition and on the other hand a bigger tolerance will result in intersection point offset errors. The second condition is also a success criteria in which if the absolute value of successive iterations values are small enough equal to a preset tolerance, the iteration exits giving out the roots. This is expressed simply as follows

$$|F(u_n, v_n)| < \varepsilon * |F(u_{n-1}, v_{n-1})| \tag{11}$$

The third 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

$$F(u_{n-1}, v_{n-1}) \cong F(u_n, v_n) \quad \text{subject to } F_n(u, v) > \varepsilon \tag{12}$$

where n denotes the nth step of iteration.

3.5. 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 according to its hit type. We have done this by calculating the normal of the surface at the point under consideration. This normal vector ($\hat{N}$) can be calculated by taking the cross product of the tangent vectors in the u and v direction evaluated at the given point. If it is assumed that the ray shoots across in the y direction, a dot product between the ray and the normal vector $\hat{N}$, would help in the classification of the points as to be an entry point, an exit point or an edge point. The points once classified are stored in a pre-defined format to be visually represented using a graphical interface and also using this data for conversion to machine instructions.

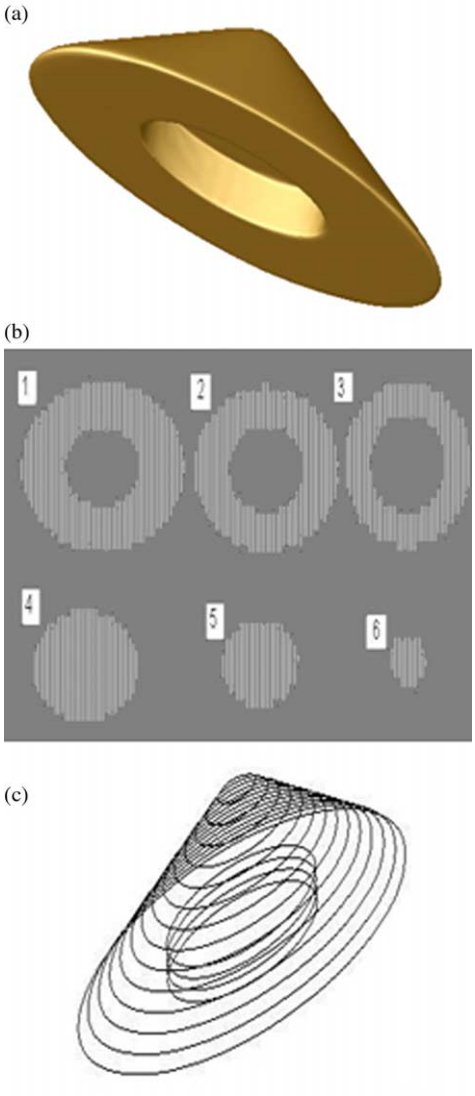

Fig. 8. (a) Case study example model 1—cone with 44 NURBS surfaces. (b) Slice raster lines at different slice layers. (c) Entry and exit points displayed.

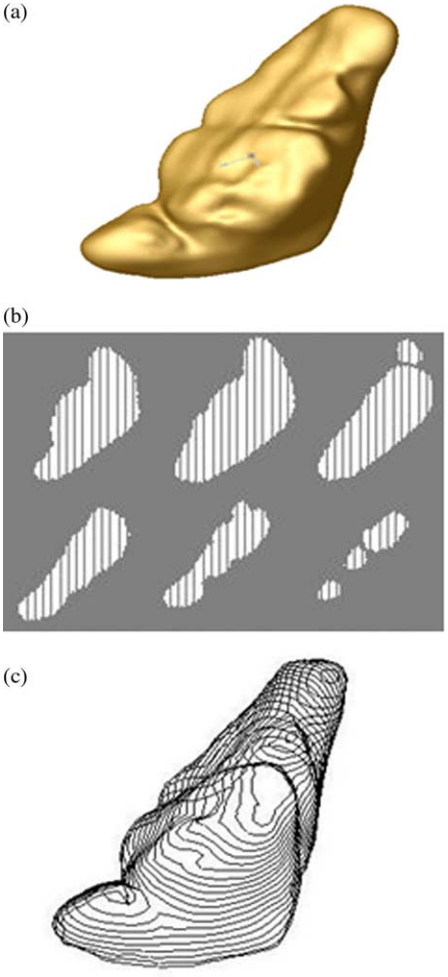

Fig. 9. (a) Case study example model 2—sinus graft 1 with 102 NURBS surfaces. (b) Slice raster lines at different slice layers. (c) Entry and exit points displayed.

4. Algorithm testing using characteristic models

The algorithms were written in C++ and tested for a variety of model shapes that included simple NURBS surfaces as well as complex curved ones. Three of the test cases each of them with special characteristics are as shown below. All the models were sliced at a constant layer thickness with constant ray offsets on the slice plane.

4.1. Model 1 (cone)

This model consists of about 44 NURBS surface definitions which were extracted from the STEP input CAD file using a STEP reader. A subdivision factor of 0.01 and a tolerance value (ε) of 0.01 were used for adaptive refinement and root finding, respectively. The model tests the capability of the algorithm to detect interior cavities as depicted by the hole in the cone at the bottom surface. Fig. 8(b) shows each of the raster patterns at different slice levels. Fig. 8(c) displays the entry and exit points at every slice layer. These points could further be joined to define vectors path pattern which can then be used in RP machines that require contour information for fabrication.

4.2. Model 2 (sinus graft model 1)

The second model contained 102 NURBS surface definitions, extracted through the same method using the reader and the data transferred to the algorithm. This model has the characteristic of having more complex curved NURBS contours along with multiple entry and exit points. 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 (ε) of 0.01 was used for the root finding routine. Fig. 9(a) shows the model optimally oriented and Fig. 9(b) illustrates the sliced model part layout at different slice levels.

4.3. Model 3 (sinus graft model 2)

The third model is more complex not only in terms of having more NURBS surfaces but also in terms of its overall shape. This model depicts the capability of the algorithm to

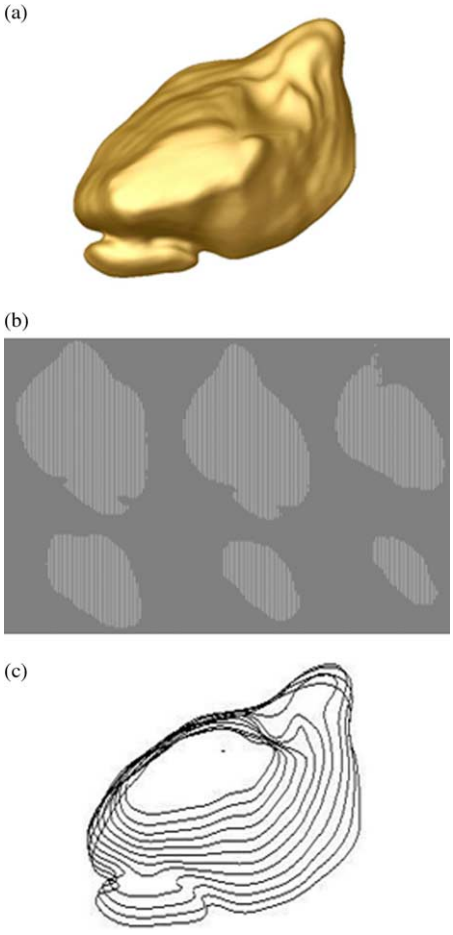

Fig. 10. (a) Model 3 sinus graft 2 layout—135 NURBS surfaces. (b) Slice raster lines at different slice layers. (c) Sliced entry and exit points displayed.

handle multiple entry and exit points that are close to each other as well as accurate slicing procedures at tangent edges of the model. A subdivision factor of 0.1 and a tolerance value (ε) of 0.01 were used for the process. Fig. 10(a) illustrates the model and Fig. 10(b), the corresponding slice layout on the fabrication bed for different layers. It can be seen that the algorithm did work well with key features detected at the slice plane shown.

4.4. Model 4 (sinus graft model 3)

The fourth test model data is neither complex nor is it a big file. The model contains 78 NURB surfaces and was used to highlight how the proposed method has overcome errors caused due to STL conversion. This particular model for some reason failed to slice using commercially available slice software. The parameters and tolerance values were 0.1 and 0.01, respectively. Fig. 11(a) illustrates the model and Fig. 11(b), the corresponding slice layout on the fabrication bed for different layers.

The advantages and disadvantages offered by the STL and STEP based methods can be seen from Fig. 12. A comparison has been drawn by the two methods of slicing in terms of file storage and processing time. All results are

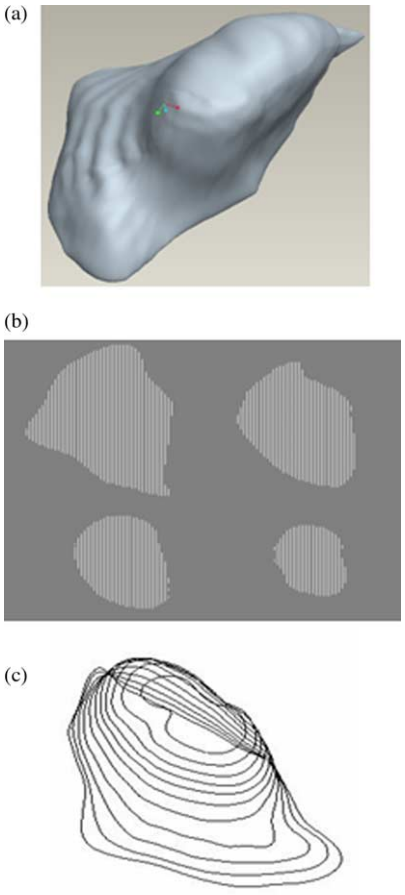

Fig. 11. (a) Model 4 sinus graft layout 3—44 NURBS surfaces. (b) Sliced raster lines at different slice layers. (c) Slice entry and exit points displayed.

Name	Format	Features	File Size	Time*
Model 1 Cone	STEP	44 NURB	164KB	52 sec
	STL	52,868 tri	12.25MB	4 sec
Model 2 Sinus Graft 1	STEP	102 NURB	860KB	71 sec
	STL	85,868 tri	25.5 MB	5 sec
Model – 3 Sinus Graft 2	STEP	135 NURB	1.09MB	72 sec
	STL	74,120 tri	23.2 MB	3 sec
Model – 4 Sinus Graft 3	STEP	78 NURB	924KB	72 sec
	STL	33,878 tri	8.55MB	failed

Fig. 12. Comparison of STL and STEP based slicing.

obtained from a Pentium 4 processor, 1GB RAM system. While direct slicing takes more time to process, it has comparatively less storage space and file check routines. Though the STL format is the quickest method to slice models, they involve higher storage costs and may require file checks and repair routines depending on the complexity of the surface. As can be seen from Fig. 12, Model-4 failed to slice using commercially available STL slicing software. Our direct method of slicing has performed its function well with no errors. The STL file may have contained errors in either inverted normals for some of the triangular surfaces or there might be small gaps between triangular patches. These errors can be corrected using STI file repair routines and then sliced normally.

5. Conclusion

Direct slicing of CAD model promises to offset the disadvantages posed by the STL format particularly with freeform shapes. The limitation of the current STL format 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 increasing the chances of errors.
- Many CAD systems fail to generate valid model tessellations and often involves manually fixing 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.

The direct slicing method does not involve file repair routines and file sizes are much smaller to handle. It also throws up 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 time 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 a direct slicing method on heterogeneous multi-material models with an appropriate slice layer format. The procedure would also be extended to leverage the benefits offered by direct slicing in designing biomimetic scaffolds for tissue engineering applications. Effort would also be put into quantifying the exact accuracy obtained by direct slicing rather than using the indirect STL format.

References

[1] Marsan A, Kumar V, Dutta D, Pratt MJ. An assessment of data requirements and data transfer formats for layered manufacturing, Technical Report NISTIR 6216, 1998. Gaithersburg, MD: National Institute of Standards and Technology.

[2] Sun W, Lal P. Recent development on computer aided tissue engineering—a review. Comput Methods Programs Biomed 2002; 67:85–103.

[3] Sun W, Darling A, Starly B, Nam J. Computer aided tissue engineering: overview, scope and challenges. J Biotechnol Appl Chem 2004;39(1):29–47.

[4] Starly B, Sun W, Lau A, Fang Z. Biomimetic model for heterogeneous bone scaffold. In: Vossoughi J, editor. Biomedical engineering: recent developments. Medical and Engineering Publishers, Inc; 2002. p. 305–6.

[5] Jamieson R, Hacker H. Direct slicing of CAD models for rapid prototyping. Rapid Prototyping J 1995;1(2):130–47.

[6] Rajagopalan M, Aziz NM, Huey CO. A model for interfacing geometric modeling data with rapid prototyping systems. Adv Eng Software 1995;23:89–96.

[7] Ma W, Peiren W. An adaptive slicing and selective hatching strategy for layered manufacturing. J Mater Process Technol 1999;89:191–7.

[8] Zhao Z, Laperriere L. Adaptive direct slicing of the solid model for rapid prototyping. Proceedings of the First Conference on Rapid Prototyping and Manufacturing '98, Beijing 1998;470–4.

[9] Sun W, Starly B, Darling A, Gomez C. Computer aided tissue engineering: application to biomimetic modeling and design of tissue scaffolds. J Biotechnol Appl Chem 2004;39(1):49–58.

[10] Rogers DF. Introduction to NURBS: with historical perspective. Los Altos, CA: Morgan Kaufmann Press; 2000. ISBN 1558606696.

[11] Piegl LA, Tiller W. The NURBS book, (2nd ed). New York, NY: Verlag; 1997. ISBN 3540615458.

[12] Owen J. STEP—an introduction, (2nd ed). Information Geometers Ltd; 1997. ISBN 1-874728-11-9.

[13] Kajiya JT. Ray tracing parametric patches. Computer Graphics SIGGRAPH '82 Proceedings 1982;16(3):245–54.

[14] Daniel L, Gonczarowski J. Improved techniques for ray tracing parametric surfaces. Visual Comput 1990;6(3):134–52.

[15] Sweeney MAJ, Bartels R. Ray tracing free form B-spline surfaces. IEEE Comput Graphics Appl 1986;6(2).

[16] Sederberg TW, Nishita T. Curve intersection using Bezier clipping. Comput Aided Des 1990;22(9):538–49.

[17] Martin W, Cohen E, Russell F, Peter S. Practical ray tracing of trimmed NURBS surfaces. J Graphics Tools 2000;5(1):27–52.

[18] Puduhai NS, Dutta D. Determination of optimal orientation based on variable slicing thickness in layered manufacturing. Proceedings of the ASME Winter Annual Conference, November 1995, San Francisco, CA 1995.

[19] Frank D, Fadel G. Preferred direction of build for rapid prototyping processes. In: Chartoff RP, Lightman AJ, Schenk JA, editors. Proceedings of the Fifth International Conference on Rapid Prototyping. University of Dayton; 1994. p. 191–200.

[20] Bartels R, John B, Brian B. An introduction to splines for use in computer graphics and geometric modeling. Los Altos, CA: Morgan Kaufmann Press; 1987. ISBN 1558604006.

[21] Yang CG. On speeding up ray tracing of B-spline surfaces. Comput Aided Des 1987;19(3).

[22] Barth W, Stürzlinger W. Efficient ray tracing for Bezier and B-spline surfaces. Comput Graphics 1993;17(4).

[23] Fournier A, Buchanan J. Chebyshev polynomials for boxing and intersection of parametric curves and surfaces. Eurographics 1994; 13(3).

[24] William HP, Saul A, Brian P. Numerical recipes in C: the art of scientific computing, (2nd ed). Cambridge: Cambridge University Press; 1993. ISBN 0-521-43108-5.

Binil Starly is currently a Ph.D candidate in the Department of Mechanical Engineering and Mechanics at Drexel University, Philadelphia, Pennsylvania, U.S.A. He received his BS degree in Mechanical Engineering from College of Engineering, Trivandrum, India. His current research interests are in the biomimetic design of tissue scaffolds from patient specific image data and the freeform fabrication process planning.

Alan Lau earned his S.B. (1972), S.M. (1974) and Ph.D. (1982) degrees from Massachusetts Institute of Technology where he majored in mechanical engineering. At present he is a professor and associate department head in Mechanical Engineering and Mechanics Department at Drexel University. His research interests include mechanical behavior of engineered and biological materials, deformation and fracture at multiple size scales, damage tolerance and structural integrity of aircrafts and human transports, and computational modeling and simulation for biomedical applications.

Wei Sun received his B.S. (1982) and M.S. (1984) degrees in Solid Mechanics from Hefei University of Technology, Hefei, China, and his M.S. (1988) and Ph.D. (1992) degrees in Mechanical Engineering from Drexel University, Philadelphia, Pennsylvania, U.S.A. Wei Sun is currently appointed as Associate Professor in the Department of Mechanical Engineering and Mechanics at Drexel University. His research and educational interests are in the interdisciplinary area of CAD/CAM, Computer-Aided Tissue Engineering, Modeling, Design and Simulation of Heterogeneous Structures, and Solid Freeform Fabrication.

Wing Lau is currently a Senior Engineer of Therics Inc., and at the same time, working for the Ph.D degree in Mechanical Engineering at Drexel University, Philadelphia, Pennsylvania, U.S.A. He received his BS (1985) degree in Mechanical Engineering from Temple University, M.S. (1987) degree in Mechanical Engineering from Drexel University, and M.S. (1992) degree in Computer Science from Villanova University. His current research interests include: biomaterials characterization by combining analytical modeling, experimental (micro/nano-indentation) data and imaging technique; biomimetic design of tissue scaffolds and the freeform fabrication process planning.

Tom Bradbury is currently the Director of Technology Development at Therics, Inc. in Princeton, NJ. Therics holds the proprietary license for three dimensional printing from MIT for medical applications with the focus on bone and soft tissue engineering for orthopedics. Tom holds a BS degree in Computer & Information Science from the University of Delaware and is in process of completing his MS degree in Computer Science at Lasalle University in Philadelphia, PA.