Title:
Point Cloud Computation for Object Slicing in 3-D Printing

Authors:
William Oropallo, woropall@mail.usf.edu, University of South Florida
Les A. Piegl, lespiegl@mail.usf.edu, University of South Florida
Paul Rosen, prosen@usf.edu, University of South Florida
Khairan Rajab, khairanr@gmail.com, Najran University

Keywords:
3-D printing, NURBS, Point cloud, Object slicing

DOI: 10.14733/cadconfP.2016.34-38

Introduction:
3-D printing has become a viable technology in the past several decades. The current state-of-the-art is to model the object using NURBS. Once the modeling has been completed, the object is converted into a tessellated model and saved as an STL file. The file is then passed onto a slicer that cuts the triangulated model into closed polygonal sections. The interior of the sections is filled with the required material and glued to the layer underneath. In principle this all sounds well, however, there are several fundamental problems with the tessellation as well as the slicing [2-4,9-12,14]. This paper investigates the possibility to use a point cloud to address this issue. It uses the original NURBS model and converts the model into a point cloud, based on layer thickness and accuracy requirements, for direct slicing. The method requires no expensive tessellation, has no anomalies, needs no model repair [6,7,13] and no conversion [8]. The only major computational requirement is point evaluation which can be done error free and in an inexpensive manner. Such an approach may have been prohibitive a decade or so ago. However, with the proliferation of powerful hardware and the abundance of memory, point-based approaches are more than viable today. Add the possibility of parallelization via a cheap multi-core GPU, the point-based approach becomes better suited in today's applications than the triangle-based ones

Overview of the Approach:
Objects to be printed (manufactured) are assumed to be bounded by B-spline surfaces, i.e. any object is considered as a collection of B-spline patches [5]. Holes and cavities are covered by B-spline patches as well and the inner part of the object is determined by the orientation of the covering surfaces. We take two parameters from the 3-D printer: (1) the layer thickness which we assume to be constant throughout the printing process, and (2) the accuracy, i.e. the addressability of the printer, the printer head can move from position to position with at least that much distance.

The overview of our approach is as follows:

- Take each B-spline surface that bounds the object and perform the operation below for each surface.
- Decompose each surface into small sub-patches that are Bezier surfaces.
- Create a global data structure that holds all the sub-patches.
- For each slicer, create an active list of sub-surfaces that may intersect the slicing plane (a reference of the sub-patch in the global data structure).

- Sample the patches in the active list based on the required tolerance and keep the resulting point cloud in a temporary data structure.
- As the slicer moves up, refresh the active list and the point cloud: drop surfaces (and the points on them) that no longer intersect the plane and add the ones that became intersecting. Sample the new surfaces and add the points to the active point cloud.

Since the method deals with a point cloud, a completely global approach would require the storage, manipulation as well as the processing of a large amount of data. What we do in this work is based on the principle of *coherence*; we are looking at a small band of surfaces and the corresponding points that are sampled off of those surfaces. In other words, we maintain a dynamic list for both the sub-patches as well as the points belonging to those surfaces.

Surface Decomposition:
The method takes an object bounded by B-spline surfaces and decomposes the surfaces into tiny patches based on the layer thickness. To make this process useful, the following issues need to be addressed.

Slicer Band
A few slicers are grouped together to reduce the number of surfaces during the decomposition. The band is used for decomposition only (the examples below are for 3-times the layer thickness).

Surface Extents
Surface area estimation is needed to determine the number of knots to be inserted to obtain the sub-surfaces. The u- and v-extents of the surfaces are used to intelligently estimate the required number of knots.

Sub-patch Computation
Surface sub-patches, localized to the slicer band, are computed via knot refinement. The required number of knots are obtained via the approximate surface extents. Figure 1 shows an example.

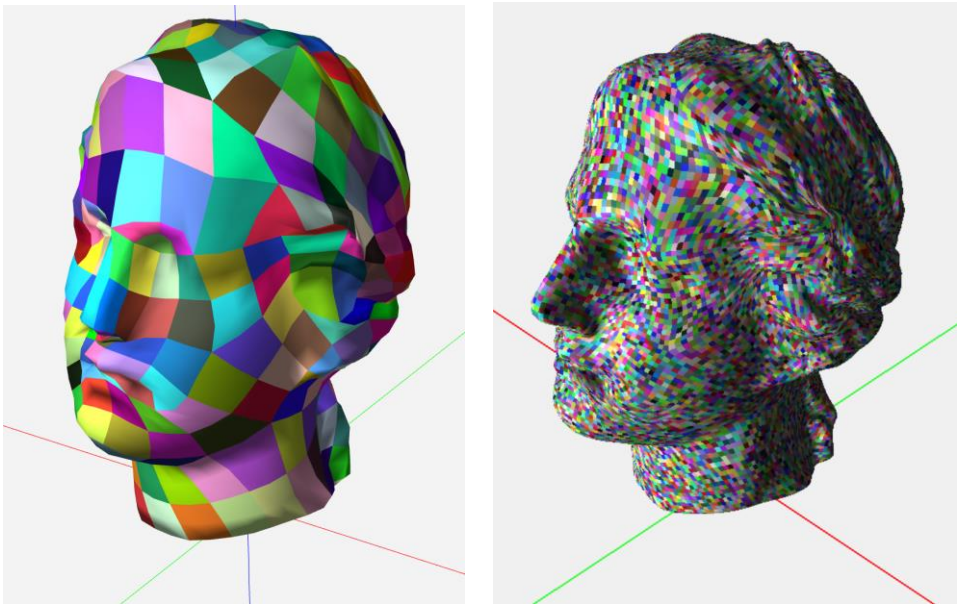

Fig. 1: Surface decomposition: (left) object covered by B-spline surfaces, and (right) sub-patches after decomposition for 3-times the layer thickness (model courtesy of Direct Dimensions, Inc.).

Table 1 below shows how the number of surfaces depends on the layer thickness. Please note that for engineering tolerances, the number of surfaces are quite small, and although tighter tolerances can produce millions of patches, in the age of big data, this is not at all large.

Layer thickness	0.001	0.01	0.1	0.2	0.3	0.4	0.5
No. of patches	8,387,525	196,000	57,975	37,075	19,975	15,700	15,075

Tab. 1: Number of sub-patches as a function of the layer thickness.

Surface Lists
A dynamic local surface list is setup to hold sub-patches that are local to the current slicer. Surfaces are added to the list if they become intersecting as the slicer moves up, and dropped off the list if they are no longer participating in intersection. Figure 2 illustrates the surfaces for slice number 228, and Figure 3 provides information on how many surfaces are added and removed from the local list, and how many participate in the slicing.

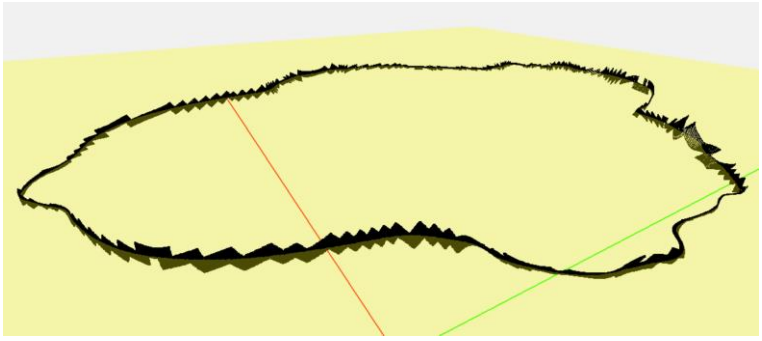

Fig. 2: Surfaces on the local list corresponding to slice number 228 of the head model.

Fig. 3: Surface patch utilization of the local list: the number of added and removed surfaces are shown in green and red, respectively, while the total active surfaces are marked in yellow.

Surface Sampling:
Three approaches will be investigated: (1) derivative based, (2) divide-and-conquer based, and (3) uniform sampling followed by quad checking.

Sampling via Bounds on Derivatives

This method calculates points on the surface so that the triangular faces formed on those points do not deviate more than the required tolerance [1]. While it is a straightforward approach, it requires the computation of the bounds of second derivatives. It also generates large triangles, large gaps in the point cloud, in places of low curvature. The applicability of this method is restricted to simple surfaces with very low curvature variation.

Sampling via Divide-and-Conquer

Divide-and-conquer is a typical B-spline technique that uses midpoint subdivision. It is tightly coupled with the geometry of the surface and produces very good results. However, the computational costs are recursion, distance and flatness tests for each subdivided patch.

Sampling via a Hybrid Method

One can combine the previous two methods into one that has a better computational efficiency at the expense of the quality of the point cloud. The method first computes a set of points at uniform parametric locations, estimated based on surface extents and the accuracy. Then the computed tiny quads are checked for distance and flatness conditions. Figure 4 shows an example of the head model using the divide-and conquer method.

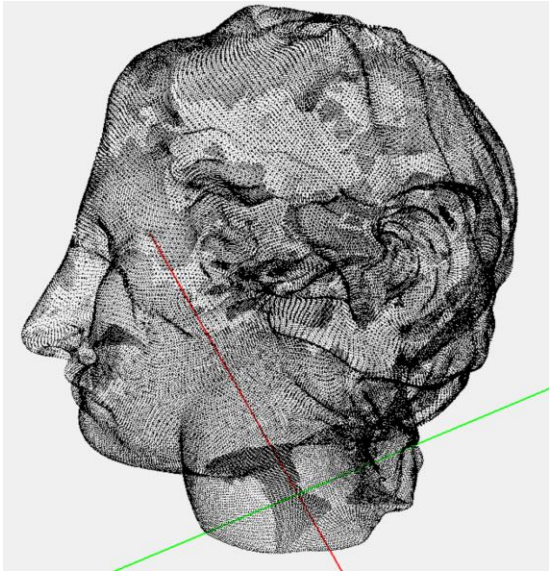

Fig. 4: Point cloud computed at 0.5 tolerance.

Slice	0	1	2	10	104	228	291	394	453
$\varepsilon = 0.1$	289	13986	17599	40687	26142	42227	43281	32972	1214
$\varepsilon = 0.01$	16641	1065024	1668899	1912570	500548	1212010	895429	516049	152100

Tab. 2: Number of sampling points to be processed for each layer.

Table 2 provides data on how many points are registered with each slicer. For one tenth of a millimeter, the maximum number of points is under 50,000. For one hundredth of a millimeter, this number jumps to about 2 million! This may sound like a large number, however, hundreds of megabytes considered manageable even on a laptop machine.

A couple of improvements are possible. First, simple surfaces, such as planes and cylinders, can safely be sampled at uniform parametric locations without the need for distance and flatness checks. All we need to know is the surface type. Second, we can reduce the band to include one layer only, thereby reducing the size of the subdivided surfaces. At the extreme, we can generate surfaces that are so tiny that they intersect only one slicer. In this case, no distance of flatness checks are needed; we only need size check and the storage of a large number of sub-surfaces.

References:

[1] Filip, D.; Magedson, R.; Markot, R.: Surface algorithms using bounds on derivatives, Computer Aided Geometric Design, 3, 1986, 295-311. http://dx.doi.org/10.1016/0167-8396(86)90005-1

[2] Ma, W.; But, W.-C.; He, P.: NURBS-based adaptive slicing for efficient rapid prototyping, Computer-Aided Design, 36, 2004, 1309-1325. http://dx.doi.org/10.1016/j.cad.2004.02.001

[3] Oropallo, W.; Piegl, L.: Ten challenges in 3D printing, Engineering with Computers, 32(1), 2016, 135-148. http://dx.doi.org/10.1007/s00366-015-0407-0

[4] Pandey, P.; Reddy, V.; Dhande, S.: Slicing procedures in layered manufacturing: a review, Rapid Prototyping Journal, 9(5), 2003, 274-288. http://dx.doi.org/10.1108/13552540310502185

[5] Piegl, L.; Tiller, W.: The NURBS Book, Springer-Verlag, New York, NY, 1997. http://dx.doi.org/10.1007/978-3-642-59223-2

[6] Piegl, L.; Rajab, K.; Smarodzinava, V.; Valavanis, K.: Fault-tolerant computing in a knowledge-guided NURBS environment, Computer-Aided Design and Applications, 6(6), 809-823, 2009. http://dx.doi.org/10.3722/cadaps.2009.809-823

[7] Rajab, K.; Piegl, L.; Smarodzinava, V.; CAD model repair using knowledge-guided NURBS, Engineering with Computers, 29(4), 477-486, 2013. http://dx.doi.org/10.1007/s00366-012-0264-z

[8] Shah, J. J.; Ameta, G.; Shen, Z.; Davidson, J.: Navigating the tolerance analysis maze, Computer-Aided Design and Applications, 4(5), 2007, 705-718. http://dx.doi.org/10.1080/16864360.2007.10738504

[9] Sikder, S.; Barari, A.; Kishawy, H.: Effect of adaptive slicing on surface integrity in additive manufacturing, Proc. ASME International Design Engineering Technical Conference, DETC2014-35559, 2014. http://dx.doi.org/10.1115/detc2014-35559

[10] Sun, S.; Chiang, H.; Lee, M.: Adaptive direct slicing of a commercial CAD model for use in rapid prototyping, International Journal of Advanced Manufacturing Technology, 34, 2007, 689-701. http://dx.doi.org/10.1007/s00170-006-0651-y

[11] Topcu, O.; Tascioglu, Y.; Unver, H.: A method for slicing CAD models in binary STL format, Sixth International Advanced Technologies Symposium, Elazig, Turkey, 141-145, 2011.

[12] Wong, K.; Hernandez, A.: A review of additive manufacturing, International Scholarly Research Network, ISRN Mechanical Engineering, 2012, ID 208760.

[13] Yau, H.-T.; Kuo, C.-C.; Yeh, C.-H.: Extension of the surface reconstruction algorithm to the global stitching and repairing of STL models, Computer-Aided Design, 35, 2003, 477-486. http://dx.doi.org/10.1016/S0010-4485(02)00078-7

[14] Zhang, L.-C.; Han, M.; Huang, S.-H.: An effective error-tolerance slicing algorithm for STL files, International Journal of Advanced Manufacturing Technology, 20, 2002, 363-367. http://dx.doi.org/10.1007/s001700200164