

DETC2005/DAC-85408

FILLETING AND ROUNDING USING A POINT-BASED METHOD

Yong Chen ^{a*}
Sr. Software Engineer

Hongqing Wang ^b
Graduate Research
Assistant

David W. Rosen ^b
Professor

Jarek Rossignac ^c
Professor

^a 3D Systems Inc., 26081 Avenue Hall, Valencia, CA 91354

^b The George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332

^c GVU Center, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332

* Corresponding Author: 661-295-5600, chen@3dsystems.com.

ABSTRACT

Rounds and fillets are important design features. We introduce a new point-based method for constant radius rounding and filleting. Based on the mathematical definitions of offsetting operations, discrete offsetting operations are introduced. Steps of our approach are discussed and analyzed. The methodology has been implemented and tested. We present the experimental results on accuracy, memory and running time for various input geometries and radius. Based on the test results, the method is very robust for all kinds of geometries.

KEYWORDS

Geometric Modeling, Point-Sampled Geometry, Offsets, Polygonal Meshes,

1 INTRODUCTION

Rounds and fillets are transitional faces found in most machined, cast and molded parts. They are important mechanical design features which serve to relieve stress concentration, to simplify fabrication, or simply to improve appearance. Currently most commercial CAD packages have rounding and filleting functions. However, we noticed these functions are not reliable for imported triangle meshes and translated CAD models. We also noticed, even for native CAD models, the functions will fail for some geometry with intersected concave features.

Adding fillets and rounds in a CAD model can be difficult because it involves not only geometrical calculations but also topological modifications. In recent years, more and more applications utilized volumetric representations based on voxels

(Kaufman, Cohen et al. 1993) and points (Pauly, Keiser et al. 2003) (Adams and Dutre 2003). Volumetric representations are attractive due to some of their unique properties. For example, Boolean operations are inherently simple, robust, and insensitive to model complexity; and surfaces with highly complex topology can be represented easily and the global consistency of the surface is guaranteed in reconstruction.

In this paper, we present a filleting and rounding method based on a set of sampling points. We chose polygonal meshes as our input models because they can be easily converted from other CAD formats or acquired by using 3D scanners. The main process of our approach illustrated by a simply example is presented in Figure 1. An input polygonal model shrinks by a ball of radius r followed by growing the temporary result with the same ball. The generated polygonal model is the input model with fillets. A point-based method is used in both shrinking and growing processes.

Our method has a sound mathematical basis and is very robust regardless of the input geometries. Besides adding fillets and rounds, our method can also be utilized to identify non-manufacturable features for a given size tool and possible interference between cutting tools and models.

2 REVIEW OF RELATED WORK

Many prior works studied the plane continuity of filleting and rounding based on *Bezier* and *NURBS* surface representation (Elber and Cohen 1997). Rossignac and Requicha (1986) studied filleting and rounding based on regularized sets and presented their relations to surface offsetting. However, most earlier work on surface offsetting such as (Rossignac and Requicha 1986) (Farouki 1985) (Satoh and Chiyokura 1991) (Frosyth 1995) all offset surfaces of

models first, then trim or extend these offset surfaces to reconstruct a closed 3D model. Due to the complexity of trimming and extending operations, the approach is difficult to implement robustly.

Some alternative representations and approaches were studied and presented before.

A ray representation method was presented in (Hartquist, Menon et al. 1999), which is a set of line segments that lie inside the solid and are generated by clipping a regular grid of lines against a solid model. Hartquist et al. proposed the strategy of using ray representations and parallel computation as primary media for offsets, sweeps, and *Minkowski* operations.

A method based on distance volume and fast marching method was presented for offsetting CSG models (Breen and Mauch 1999). The approach calculates the shortest distance to the CSG model at a set of points within a narrow band around the evaluated surface. Additionally, a second set of points, labeled the zero set, which lies on the CSG model's surface are computed. A point in the zero set is associated with each point in the narrow band. Once the narrow band and zero set are calculated a fast marching method is employed to propagate the shortest distance and closest point information out to the remaining voxels in the volume. Other works on calculating distance maps and their representations can also be found (Gibson 1999; Frisken, Perry et al. 2000). However, most of these works are developed for displaying purpose in computer graphics domain. No accuracy analysis of offsetting results

was found in the literatures while it is a main concern for manufacturing purpose.

More recently, Kim et al. (2003) presented a five-stage pipeline to approximate the swept volume of a polyhedron along a given trajectory. The main approach is based on computing unsigned distance fields on a uniform grid, classifying all grid points using fast marching front propagation and reconstructing volume from iso-surfaces.

Based on moving least squares (MLS) presented in (Levin 1998), Pauly et al. (2003) presented a point-based approach for interactively modeling shapes. Adams and Dutre (2003) presented a point-based approach for interactive Boolean operations.

The remainder of this paper has been organized in the following manner. Section 3 presents the basic definitions and problem formulation. Section 4 presents the steps of our method and the analysis on accuracy, performance and memory requirements. Section 5 demonstrates the accuracy, the efficiency, and robustness of the approach in different test cases. An application of our approach in simulating manufacturing process is also discussed in Section 5. Finally, Section 6 gives the conclusions of our research.

3 DEFINITIONS AND FORMULATION

In this paper, we mainly focused on the constant-radius rounds and fillets because they are commonly used in product designs. Global operations are studied although the method can be extended to local operations on individual faces and edges.

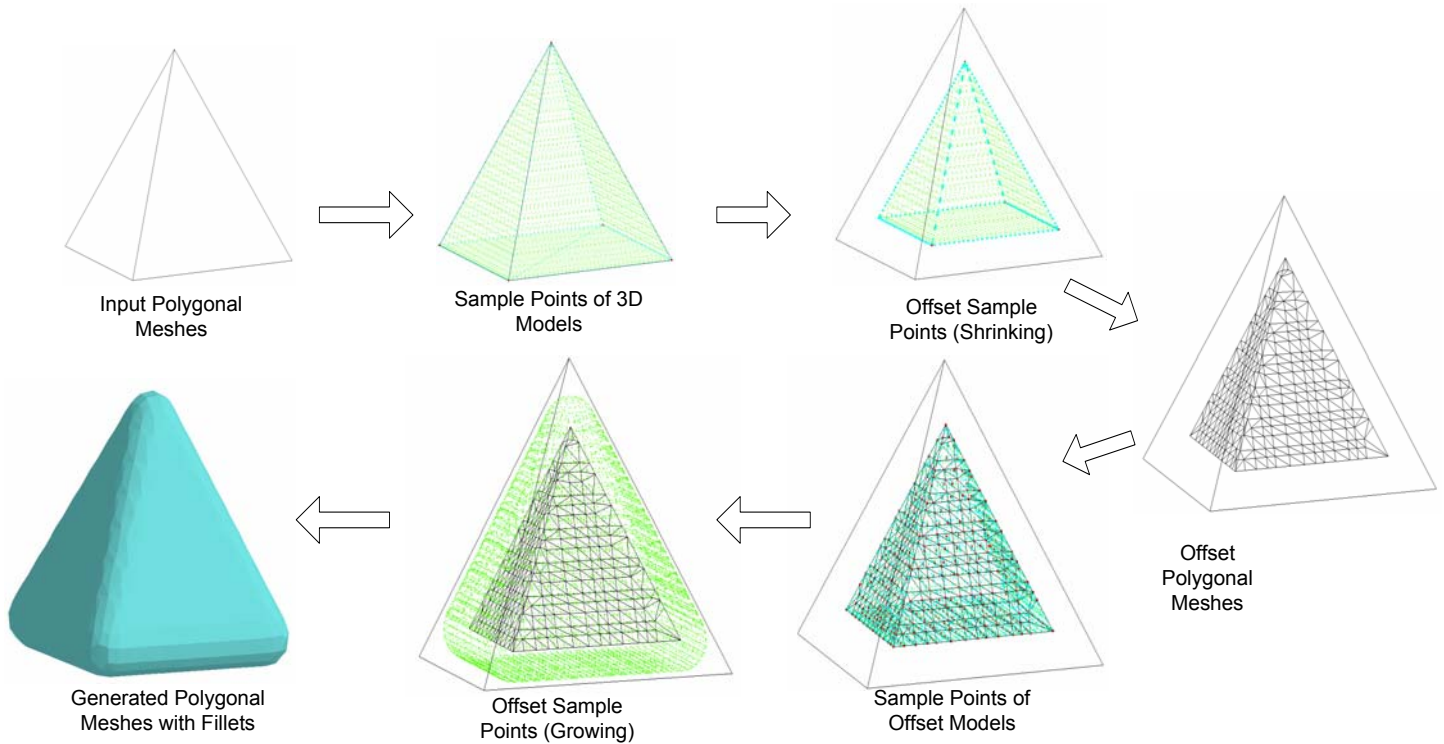


Figure 1 The Main Process of Point-based Filletting and Rounding Method

3.1 Filleting and Rounding Related to Offsetting

Many operations in mathematical morphology can be defined in terms of the *Minkowski* sum and difference. Suppose two sets A and B are closed and regular subsets of Euclidean space \mathbf{E}^2 or \mathbf{E}^3 . The *Minkowski* sum of two sets A and B , denoted $A \oplus B$, is defined as $C = A \oplus B = \{a + b \mid a \in A, b \in B\}$, where “+” denotes the normal vector addition of two points. The Minkowski difference, denoted $A \otimes B$ is $\overline{A \oplus B}$. Suppose a ball with radius r is defined as b_r , we can define two offsetting operations, S grown by r as $S \uparrow_r = S \oplus b_r$, and S shrunk by r as $S \downarrow_r = S \otimes b_r$ for any model S .

Set regularization is an operation that takes a set S into the topological closure of the interior of S . Based on regularized set and its properties, Rossignac and Requicha (1986) proved that constant radius rounding and filleting can be modeled by a combination of the two offsetting operations as:

- (1) S filleted by r can be defined as $F_r(S) = S \uparrow_r \downarrow_r$.

That is, fillets can be generated for a model S by first growing S by radius r , then shrinking the generated model S' by r .

- (2) S rounded by r can be defined as $R_r(S) = S \downarrow_r \uparrow_r$.

That is, rounds can be generated for a model S by first shrinking S by radius r , then growing the generated model S' by r .

A simple 2D example is given in Figure 2 and Figure 3 to illustrate the filleting and rounding individually. These definitions and properties are the foundations of our point-based method.

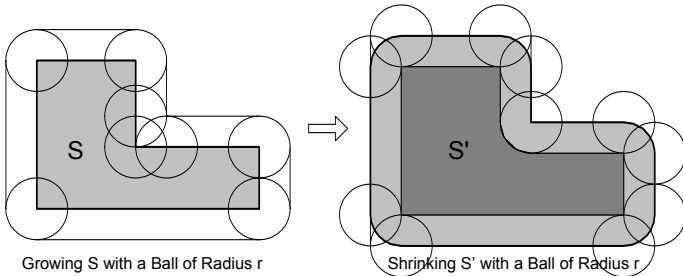


Figure 2 A 2D Example of Filleting

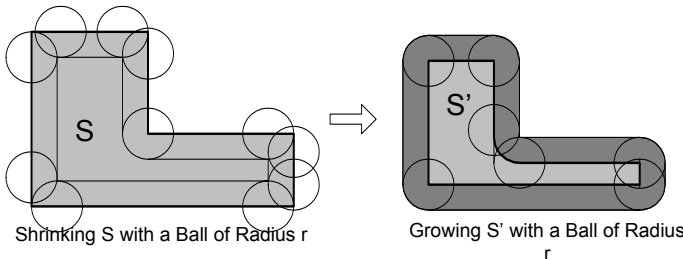


Figure 3 A 2D Example of Rounding

3.2 Continuous Offsetting Operation

The regularized positive offset of a regular set S by a positive distance r can be defined as $S \uparrow^* r = \{p : \exists q \in S, \|p - q\| \leq r\}$. In terms of point/set distances, Nadler (1978) gave another equivalent definition $S \uparrow^* r = \{p : d(p, S) \leq r\}$, where $d(p, S) = \inf_{q \in S} \|p - q\|$ and \inf denotes the *greatest lower bound*. From this definition, if p is exterior to S , then the closet points q lies in ∂S , which is the topological boundary of S . That is, $\partial(S \uparrow^* r) \subset \{p : d(p, S) = r\}$.

The regularized negative offset of a non-empty S is defined as the complement of the positive offset of the complement of S . So the analogous result in terms of point/set distances for a negative offset of solid S is $\partial(S \downarrow^* r) \subset \{p : d(p, c^* S) = r\}$, where c^* denotes regularized complement.

As $\partial S = V(S) \cup E(S) \cup F(S)$, where $V(S)$, $E(S)$, and $F(S)$

refer to a vertex, an edge, or a face of S , any point q in ∂S must be in one of the sets. The point p to make $d(p, S) = r$ related to q is considered as follows.

- (1) Faces $F(S)$: Suppose $q \in F$. Vector $p - q$ must be normal to F at q , that is, $p = q + rn$.

- (2) Edges $E(S)$: Suppose $q \in E$. p lies in a circle of radius r and center q in the normal plane. Furthermore, suppose the normals of two neighboring faces (f_1, f_2) of q are n_1, n_2 . p lies in the arc of the circle determined by n_1 and n_2 (including the normal point), because points outside the arc are closer to f_1 and f_2 .

- (3) Vertices $V(S)$: Suppose $q \in V$. p lies in a sphere of radius r and center q . Furthermore, suppose the neighboring edges of q are e_1, e_2, \dots, e_i . p lies in a region of the sphere which are formed by the positive normal offset of edges $e_1 \sim e_i$.

Notice although all the points p generated by points q on ∂S are at a distance r from q , they may be at a smaller distance to other points on ∂S . In this paper, we used point representation to handle the global verification of $d(p, S) = r$. Accordingly, the offsetting operations defined for continuous boundary cases need to be converted into discrete offsetting operations.

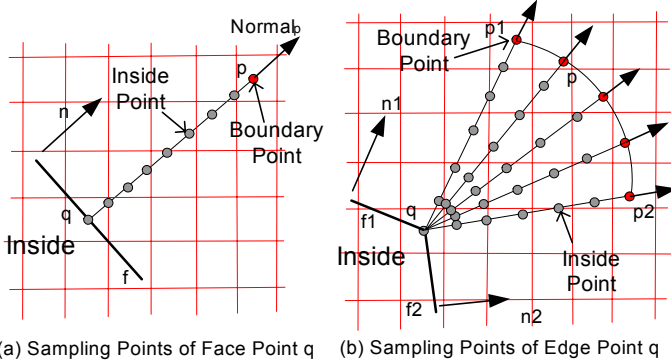
3.3 Discrete Offsetting Operation

Suppose the sampling rate is SR_p , which is defined as the number of sampling points per unit length along an axis. Correspondingly the sample point size SS_{point} is $1/SR_p$. Two kinds of points, *boundary points* and *inside points*, are defined here. The *boundary points* are points p with $d(p, q) = r$ for a point q on ∂S . Notice a boundary point p may not be on $\partial(S \uparrow^* r)$ or $\partial(S \downarrow^* r)$ if its distances to any other points on ∂S are smaller than r . The *inside points* are points p with

$d(p, q) < r$. The *inside points* are used for the global verification of $d(p, S) = r$ so boundary points that are not on offset boundaries are identified. The related discrete offsetting operations are defined as follows.

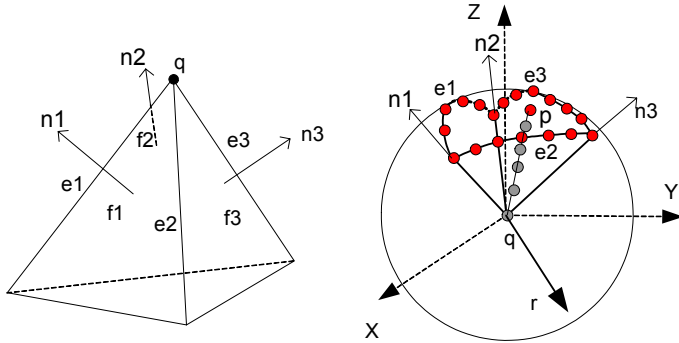
(1) Faces $F(S)$: For each point q of F , a boundary point $p = q + r\mathbf{n}$, where \mathbf{n} is the unit normal of F and r is the offset distance. Correspondingly, a set of inside points can be generated by sampling line pq using SR_p (Figure 4.a).

(2) Edges $E(S)$: For each point q of E , an arc with radius r is defined by the big circle in a sphere centered in q . The starting and ending points of the arc are determined by n_1 and n_2 , where n_1 and n_2 are the unit normal of two neighboring faces f_1 and f_2 . The arc is sampled first into some boundary points. For each point p in the arc, a set of inside points can be generated by sampling line pq using SR_p (Figure 4.b).



(a) Sampling Points of Face Point q

(b) Sampling Points of Edge Point q



(c) Sampling Points of Vertex Point q

Figure 4 Discrete Offsetting Operation

(3) Vertices $V(S)$: For each point q of V , a region formed by a set of arcs is defined on a sphere with radius r centered in q . The boundary of the region are determined by e_1, e_2, \dots, e_i where $e_1 \sim e_i$ are the neighboring edges of q . The region is sampled first into some boundary points. For each point p in the region, a set of inside points can be generated by sampling line pq using SR_p (Figure 4.c).

A boundary point p of $F(S) \cup E(S) \cup V(S)$ is a point in $\partial(S \uparrow^* r)$ if it satisfies $d(p, \partial S) = r$ and $p \notin S$. Therefore, by getting rid of all boundary points p with $d(p, \partial S) < r$, the remaining points are a set of sampling

points of $\partial(S \uparrow^* r)$, and we propose an algorithm for reconstructing an approximation of the offset boundary from these samples.

The detailed approaches of our method are discussed in Section 4 with the error and performance analyses.

4 THE POINT-BASED METHOD AND ITS ANALYSIS

4.1 Method

The steps of our point-based filleting and rounding method are described as follows.

- (1) Determine sample point size SS_{Point} based on error requirement ϵ and the smallest feature size of the input solid model;
 - (2) Generate a set of sample points SP_{circle} and SP_{sphere} as the sampling points of a circle and a ball with radius r ;
 - (3) Repeat (4)~(9) for two iterations. For fillet, $r_1 = r$ (outside) and $r_2 = -r$ (inside); for rounding, $r_1 = -r$ (inside) and $r_2 = r$ (outside).
 - (4) Generate a set of sample points SP_{model} ;
 - (5) Iterate each point in SP_{model} , generate a set of boundary point P_B based on the sample point type (vertex, edge or face) and SP_{circle} and SP_{sphere} ;
 - (6) Iterate each point p in P_B and delete boundary points p' which are within the distance r from the ball center related to q ;
 - (7) Calculate a set of boundary points related to edges and vertices of the offset model based on the deleted boundary points;
 - (8) Construct a surface model from the remaining boundary points;
 - (9) If necessary, simplify the generated surface model.
- An explanation of each step is provided as follows.

Our approach uses an approximation scheme. As shown in Section 4 and 5, the accuracy of filleting and rounding results is related to the sampling size SS_{Point} and r . Based on our analysis, we estimated the maximum error is $\frac{SS_{Point}}{6}$ for

$$\frac{r}{SS_{Point}} = 6 \quad (\text{Section 4.2}).$$

Therefore to achieve a given accuracy,

we can set the value of SS_{Point} for a given r . Besides sampling points, we also used a set of uniform grids to speed up the processing speed (refer to Figure 4.a and b). That is, all the sampling and boundary points are stored in some hash tables which are identified by the grid position (ix, iy, iz) . So if a grid is entirely within the distance of r , all the points within the grid can be deleted. A good grid size, based on our experience, is $2 * SS_{Point}$, which is used in all the following tests. Since we used grid-based techniques in reconstructing a surface model from points (step 8), another determining factor of SS_{Point} is the smallest feature of the result model. It is well known that the smallest feature for grid-based techniques can not be smaller than the grid size. However, it should be noticed that if we use some grid-less technique in this step (e.g. Ball-pivoting algorithm as discussed later), this requirement will not be necessary.

Before the offsetting processes start in Step (3), a set of uniform sample points in a circle and a sphere can be generated based on the ratio $\frac{r}{SS_{point}}$. (Du, Gunzburger et al. 2003).

Obviously, when ratio $\frac{r}{SS_{point}}$ increases, the spherical surface gets bigger, and more sampling points will be generated for the approximation.

Pfister *et al.* (2000) presented surfel (surface element or surface voxel) and related techniques to render complex geometric objects. In this paper, we used similar principles in generating sample points from geometric models as those presented in (Pfister, Zwicker et al. 2000) in generating surfels. However, there are two differences to be noticed: (i) we sampled edges and vertices in addition to surfaces; (ii) we sample models from the center of a cube instead of its boundary due to the offsetting operation. An example of sampling points for a cube is given in Figure 5.a. In the figure, vertex sample points are shown in red; edge sample points are shown in cyan and face sample points are shown in green.

In Step (6), a boundary point p is used to judge if another candidate boundary point p' is within the distance r from sample point q related to p . For all the grids that pq intersects, if a grid is entirely within the distance r from q , we can mark the grid and delete all boundary points within the grid later. If the grid is not entirely inside, we need to iterate each boundary point within the grid to judge if it is within the distance r from q . We used a hash table data structure to associate points with grids. Based on the point type of q , we used different judging approaches to minimize the sampling errors during conversion of continuous geometries into discrete points.

- (i) Face: we used a segment plane with size $\frac{SS_{point}}{\sqrt{2}}$ and centered at q to judge p' . Suppose point q' is the projection of p' onto the face f .
 - (a) if $\|q'q\| < \frac{SS_{point}}{\sqrt{2}}$,
 - (a.1) if $\|q'p'\| < r$, p' is inside;
 - (a.2) otherwise, it is outside.

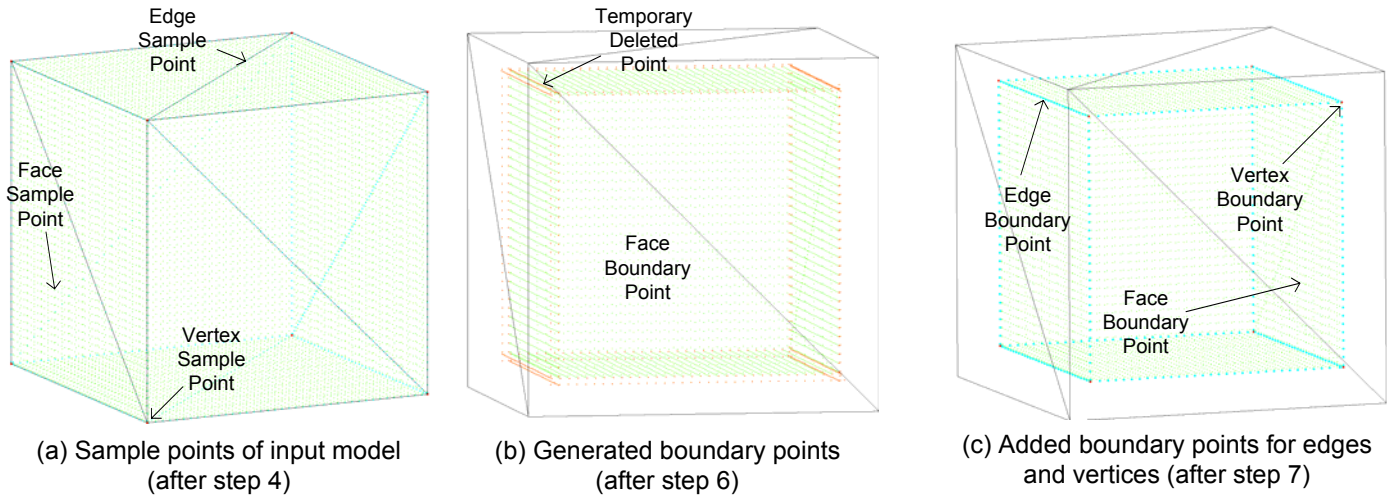


Figure 5. Offsetting a cube by $r=-0.1$ (shrinking).

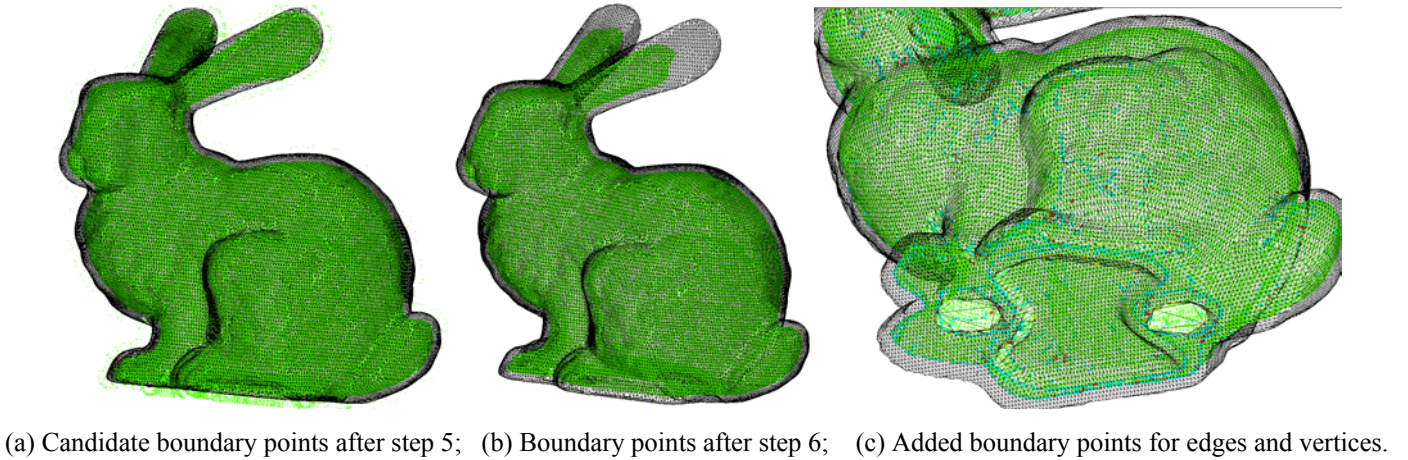


Figure 6. A Test Case of Point-based Offsetting Approach.

- (b) otherwise,
 - (b.1) if $\|qp'\| < r$, p' is inside;
 - (b.2) otherwise, it is outside.
- (ii) Edge: we used a cylinder with size SS_{point} and centered at q to judge p' . Suppose point q' is the projection of p' onto the edge e .
 - (a) if $\|q'q\| < \frac{SS_{point}}{2}$,
 - (a.1) if $\|q'p'\| < r$, p' is inside;
 - (a.2) otherwise, it is outside.
 - (b) otherwise,
 - (b.1) if $\|qp'\| < r$, p' is inside;
 - (b.2) otherwise, it is outside.
- (iii) Vertex: we used a sphere to judge p' .
 - (a) $\|p'q\| < r$, p' inside;
 - (b) otherwise, it is outside.

After processing each boundary point p , all the boundary points which are inside the offsetting distance r are marked as “Deleted”, and will be deleted after Step (7). An example of the boundary points with and without “Deleted” points is shown in Figure 6.a and b individually. In the figure, all the boundary points are shown in green. The wire-frame of the original bunny model is shown in gray.

In order to accurately reconstruct a mesh from the boundary points, we explicitly calculated a set of points related to edges and vertices of the generated meshes, and added them to the boundary points (vertex boundary points and edge boundary points are shown in red and cyan individually in Figure 5.c and Figure 6.c). These intersection points are calculated based on some of the deleted boundary points which are within the distance range of $(r-SS_{point}, r)$. We called these points as *Temporary Deleted Points* (shown in orange color in Figure 5.b). Based on the positions and normals of a *Temporary deleted point* and its neighboring *Temporary deleted points*, we used quadratic error function (QEF) (Garland 1999) to calculate the intersection point. The point dimension is the number of non-zero eigen-values during the QEF calculation (1=plane, 2=edge, 3=vertex).

The approach to reconstruct a surface model from a set of points was studied intensively in 3D scanning applications. In this study, we adapted Dual-Contour method (Ju, Losasso et al. 2002) in reconstructing polygonal meshes. Two major differences are: (1) For each cube, we used boundary points within the cube in the calculation of quadratic error function (QEF) for a vertex; (2) An additional step is added for a cube which does not have a boundary point while it has an edge that exhibits a sign change. The approach we used for the step is based on the vertices of its neighboring cubes and its neighboring boundary points.

We are also exploring some other grid-less reconstructing method. A good candidate for our purpose is Ball-Pivoting algorithm (Bernardini, Mittleman et al. 1999), which does not

require a set of grids to identify corners inside or outside of sampling points.

In many cases, the generated polygonal meshes may be too dense especially for small sample point size. Research on polygonal mesh simplification (Garland 1999) can also be used to reduce the mesh size of the generated models.

The three key steps in our point-based offsetting operations are further demonstrated in Figure 6 using the Stanford Bunny model. The data structures and algorithms of these steps are explained in more details in (Chen, Wang, et al, 2005).

The theoretical analysis of our approach is presented in the remaining of the section.

4.2 Error Analysis

Our method of generating filleting and rounding results for an input surface model is an approximation scheme. There are four approximation steps in the process. They are listed as follows with their corresponding error analysis.

- (i) Input Surface Model \rightarrow Sample Points (Step 4): Each sample point is taken from the boundary of the input surface model and represented in floats. The discontinuity of the surfaces is also captured since edges and vertices are sampled individually. The approximation error in this step is very small and can be omitted.
- (ii) Sample Points \rightarrow Boundary Points (Step 5): Each boundary point is taken from an arc or a sphere with radius r and represented in floats. So the approximation error in this step is very small and can be omitted.
- (iii) Deleting Boundary Points which are within offsetting distance r (Step 6): The approximation of a vertex sample point by a sphere is accurate; the approximation error of an edge sample point by a cylinder is less than $(\frac{r}{SS_{point}} - \sqrt{(\frac{r}{SS_{point}})^2 - \frac{1}{4}}) \cdot SS_{point}$; the approximation error of a face sample point by a segment plane is less than $(\frac{r}{SS_{point}} - \sqrt{(\frac{r}{SS_{point}})^2 - \frac{1}{2}}) \cdot SS_{point}$. So if $\frac{r}{SS_{point}} = 4$, the maximum error we will misjudge a boundary point inside or outside the r distance is less than $0.06 \cdot SS_{point}$.
- (iv) Boundary Points \rightarrow Surface Model (Step 8): In the offset models, only three kinds of geometries exist. They are faces related to $F(S)$, cylinders related to $E(S)$ and spheres related to $V(S)$. The error generated in this step is geometry-dependent. We used a grid-based method which converts multiple boundary points into a single grid point. Therefore, any features that are smaller than a grid size will disappear. However, if all features are comparably bigger than the grid size and grid points are selected from boundary points to make the offset meshes

uniform, the reconstructed meshes will be accurate for offset faces; the maximum error for cylinders in the offset meshes is $(\frac{r}{SS_{Grid}} - \sqrt{(\frac{r}{SS_{Grid}})^2 - \frac{1}{4}}) \cdot SS_{Grid}$

by using the linear interpolation of polygonal meshes; the maximum error for spheres in the offset meshes is $(\frac{r}{SS_{Grid}} - \sqrt{(\frac{r}{SS_{Grid}})^2 - \frac{1}{2}}) \cdot SS_{Grid}$. So if $SS_{Grid} = 2 \cdot SS_{Point}$ and $\frac{r}{SS_{Point}} = 4$, the maximum error in this

approximation is $0.26 \cdot SS_{Point}$. Comparing to the marching cube method (Lorenson and Cline 1987), a benefit of the Dual-contour method we used is it will preserve the sharp features.

By considering all the steps, if a grid size is $2 \cdot SS_{Point}$ and all features are bigger than the grid size, the approximation error of our method is less than $\frac{SS_{Point}}{4}$ for $\frac{r}{SS_{Point}} = 4$ and $\frac{SS_{Point}}{6}$ for $\frac{r}{SS_{Point}} = 6$. The approximation error will be smaller if we reduce grid size or use some grid-less reconstructing method.

4.3 Performance Analysis

Suppose the size of the input surface model is S_x, S_y, S_z and S is the maximum of S_x, S_y , and S_z . Our algorithm has the following computational complexity:

Steps (1), (2), (4) (Sample Point Generation): The scan conversion algorithm goes through each triangle to check if it intersects the center of a cube. Since we only need to calculate the cubes within the bounding box of the triangle, the computational time is $O(N_{Triangle} \frac{S^2}{SS_{Point}^2})$. The

computational time to sample points and edges are $O(N_{Vertex})$ and $O(N_{Edge} \frac{S}{SS_{Point}})$ respectively, which

are generally much smaller than the time of sampling triangles.

Step (5), (6), (7) (Boundary Point Generation): The time to process a sample point is constant for constant $\frac{r}{SS_{Point}}$, although different constant factors

are related to various types of offset points (vertices, edges or faces). Therefore the computational time is proportional to the number of sample points, which is $O(N_{Triangle} \frac{S^2}{SS_{Point}^2} + N_{Edge} \frac{S}{SS_{Point}} + N_{Vertex})$;

Step (8) (Isosurface Extraction): The computational time is proportional to the number of grids, which is $O(\frac{S^3}{SS_{Point}^3})$.

Therefore the total Complexity of our method is

$$O(\frac{S^3}{SS_{Point}^3} + N_{Triangle} \frac{S^2}{SS_{Point}^2}).$$

4.4 Memory Analysis

The main memory requirements of our algorithms are an array of grids, a list of sample points and a list of boundary points: (i) The size of the grid array is $(\frac{S_x S_y S_z}{SS_{Point}^3}) \times \frac{1}{4}$ bytes

since each grid only needs two bits; (ii) The size of the sample point list depends on the surface area of the input model. It is well known that the surface area of a model is proportional to $Size^2$ while its volume is proportional to $Size^3$. Therefore the size of the sample list is estimated as $\frac{Area}{SS_{Point}^2} \times 17$ bytes since each sample point takes 17 bytes in our implementation; (iii)

similarly, the size of boundary point list is $\frac{Area}{SS_{Point}^2} \times 30$ bytes since each boundary point takes 30 bytes in our implementation.

After theoretically analyzing the accuracy and performance, we present the experimental results for various tests in the next section.

5 RESULTS AND APPLICATIONS

We used C++ programming language with *Microsoft Visual C++* compiler to implement the presented algorithms. In this section, we present our test results by highlighting the accuracy of the results, and the performance and robustness of the algorithms.

Table 1. Accuracy Test Results

Sample Point Size	Fillet (r=0.1)			Round (r=0.1)		
	Max Error	Average Error	Standard Dev.	Max Error	Average Error	Standard Dev.
0.025	0.0127	0.00125	0.00116	0.0106	0.00065	0.0012
0.0125	0.0021	0.0002	0.00016	0.0044	0.00019	0.00034
0.01	0.0009	0.00012	0.00008	0.0029	0.00014	0.00018
0.005	0.0003	0.00005	0.00003	0.0015	0.00006	0.00007

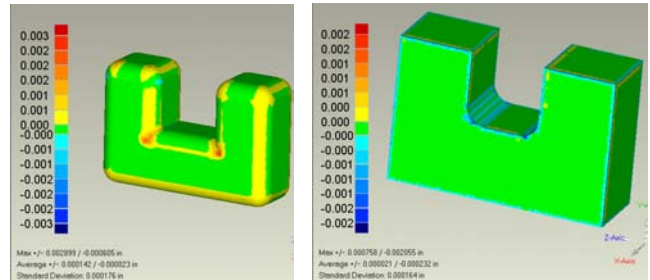


Figure 7. Comparing Results for Accuracy Test.

5.1 Experimental Accuracy

Our approach generates a polygonal model to approximate the input model with fillets or rounds. As analyzed in Section 4, the accuracy of our approach depends on the sample point size SS_{Point} and radius r . To test the accuracy of our method, we used a simple part (a $0.4 \times 0.8 \times 1.2$ cube with a groove in the middle) and construct its constant radius fillets and rounds ($r=0.1$ inch) in *SolidWorks* 2005 (www.solidworks.com). We assume the results generated by *SolidWorks* are accurate since it uses exact geometry calculations. We used commercial software, *Geomagic Qualify 7* (www.geomagic.com), to compare the results generated by our algorithms and the results generated by *SolidWorks*. Four different sample point sizes were tested from 0.005 to 0.025 inch. The results generated by *SolidWorks* are set as the reference. The comparison results (maximum errors, average error, and standard deviations) related to the sample point size are shown in Table 1. Obviously, the accuracy of the results increased with smaller sample point size. The screen captures of two results are also shown in Figure 7.

5.2 Algorithm Performance

Beside accuracy tests, we also performed a number of tests on different geometries by using various offset distances and radius sizes. The test results for a Beethoven statue (XYZ size: $1.25 \times 1.0 \times 0.7$) as shown in Figure 9 are provided in Table 2. All the tests were done in a PC with a 1.7 GHz *Pentium Xeon* processor and 2GB *DRAM* running *Windows XP*.

It is quite obvious that the running time in the first iteration is much less than the running time in the second iteration. This is because in the current implementation, we generated a temporary solid model as the result of the first iteration, and use it as the input for the second iteration. Comparing to the given geometry, the polygonal meshes are too dense with much more

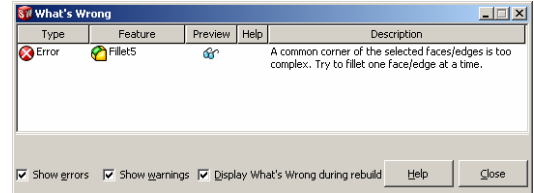


Figure 8. A Typical Fillet Error Given by SolidWorks.

vertex and edge sample points. One possible fix is to decimate the generated polygonal meshes before running the second iteration. Another approach which is more promising is to directly convert the generated boundary points in the first iteration into the sampling points in the second iteration. So no temporary solid model is generated. All the approaches should greatly reduce the running time.

5.3 Algorithm Robustness

The method used by CAD software such as *SolidWorks* is based on exact geometry calculations. It is generally more accurate and faster for models in native CAD formats than our method for models in polygonal meshes. However, the biggest benefit of our approach is its robustness. That is, it will always work regardless of the input geometry and the method is comparably easier to be implemented robustly. We tested many cases which failed in *SolidWorks* while correct results are generated in our system. Even for some simple cases as shown in Figure 10, we got an error message when trying to add fillets around common corners. The error message given by *SolidWorks* for the case in Figure 9.b is shown in Figure 8. We tested the same models using our approach and the results generated by our system are shown in Figure 10. We also tested some more complex polygonal meshes. One of the test cases and the filleting and rounding results are shown in Figure 9.

5.4 Simulation of Manufacturing Results

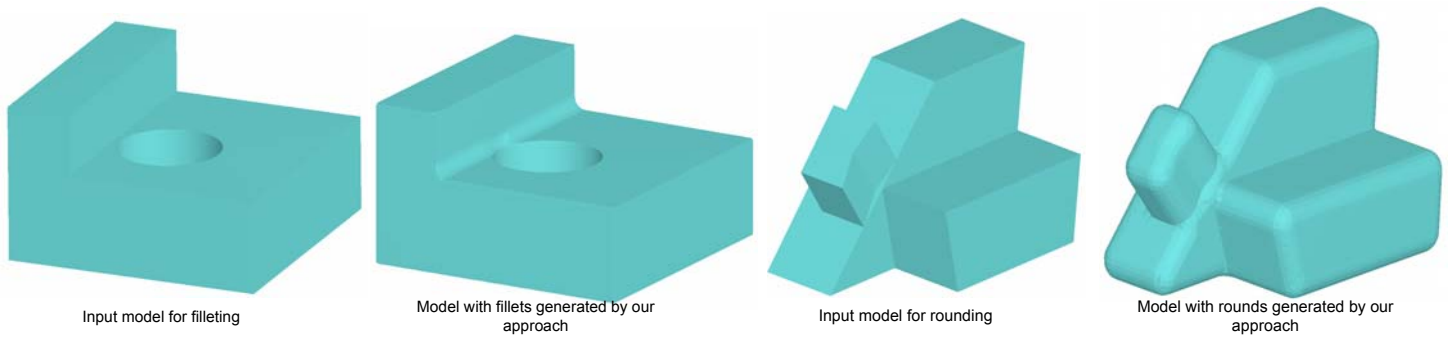
Essentially, the filleting results are the simulation of cutting manufacturing results (finding the cutting tool path then deleting materials), and the rounding results are the simulation of additive manufacturing results (finding the laser tool path then adding materials). Therefore our results can also be used to identify non-manufacturable features for a given size tool. Also notice that all the features will be deleted in the generated results if the features intersect the tools during the fabrication process. An example is shown in Figure 11 for the rounding process. The ball radius we used is 0.01 inch. The generated result has some features totally deleted as shown in Figure 11.a and b. A planar section of the region with identified non-manufacturable features is shown in Figure 11.c. From the planar section, it is quite obvious that the tool size we selected is inappropriate since the size of a feature is smaller than the tool size. It is necessary to choose another tool with a smaller size, or the designer has to modify his/her design in this feature in order to make it manufacturable.

Table 2. Performance Test Results.

Fillet ($r = 0.02$)							
Sample Point Size	Iteration	Vertex Sample Point #	Edge Sample Point #	Face Sample Point #	Boundary Point number	Memory (MB)	Running Time (Sec)
0.002	1 (Grow)	2521	175795	968076	1571086	211	108
	2 (Shrink)	154177	1643093	1076475	4684140	385	1407
0.003	1 (Grow)	2521	117162	430717	771720	115	64
	2 (Shrink)	103059	906688	483983	2590505	170	695
0.004	1 (Grow)	2521	87912	241479	2077158	56	38
	2 (Shrink)	57850	155966	275353	2337228	109	258
0.005	1 (Grow)	2521	70318	154811	327143	35	30
	2 (Shrink)	37123	331965	177073	936292	67	147
Round ($r = 0.02$)							
Sample Point Size	Iteration	Vertex Sample Point #	Edge Sample Point #	Face Sample Point #	Boundary Point number	Memory (MB)	Running Time (Sec)
0.002	1 (Shrink)	2521	175795	968076	1425749	205	102
	2 (Grow)	123265	1333797	881201	4267200	324	1170
0.003	1 (Shrink)	2521	117162	430717	690346	112	60
	2 (Grow)	82201	729478	394233	2315333	169	505
0.004	1 (Shrink)	2521	87912	241479	398918	55	36
	2 (Grow)	46151	418071	224702	1249060	91	190
0.005	1 (Shrink)	2521	70318	154811	286756	33	10
	2 (Grow)	29346	268580	144393	821891	56	115



Figure 9. A Test Case for Polygonal Meshes ($r = 0.02$, sample point size = 0.003).



(a) A Test Case for Constant Radius Filleting

(b) A Test Case for Constant Radius Rounding

Figure 10. Two Test Cases Failed in Commercial CAD Packages.

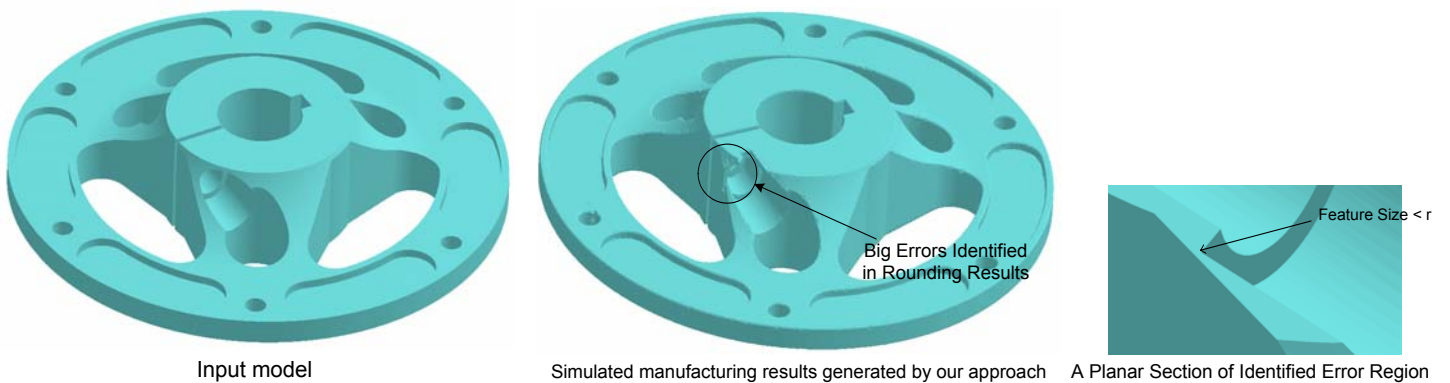


Figure 11. A Test Case for Simulation of Additive Manufacturing Process ($r = 0.01$).

6 CONCLUSION

We propose a hybrid point-based method for rounding and filleting solid models. The approach is general, efficient, and straightforward to implement. It samples the boundary, generates surfels in the vicinity of the model, uses a spatial grid to eliminate candidate samples that are too close to the original surface and restores the offset boundary through interpolation. We provide examples and discuss how to select the sampling rate in order to achieve the desired accuracy.

Different geometry representations (voxels, points and surfaces) have their unique advantages. Linking them together into a hybrid representation provides a promising direction to solve difficult geometrical problems. There are many areas for future work. We would like to investigate and further improve the performance of our algorithms. We would also like to use some grid-less techniques for surface reconstructing from points. Finally, we would like to extend our method to local operations and explore the usage of our method in other applications.

REFERENCES

- Adams, B. and P. Dutre. *Interactive Boolean Operations on Surfel-Bounded Solids*. in *Proceedings of ACM SIGGRAPH*. 2003. San Diego, CA.
- Bernardini, F., J. Mittleman, H. Rushmeier, C. Silva, G. Taubin (1999). "The Ball-Pivoting Algorithm for Surface Reconstruction." *IEEE Transactions on Visualization and Computer Graphics* **5**(4): 349 - 359.
- Breen, D.E. and S. Mauch. *Generating Shaded Offset Surfaces with Distance, Closest-Point and Color Volumes*. in *Proceedings of the International Workshop on Volume Graphics*. 1999.
- Chen, Y., H. Wang, D. Rosen, J. Rossignac (2005). "A Point-based Offsetting Method of Polygonal Meshes." (in preparing).
- Du, Q., M. D. Gunzburger, L. Ju (2003). "Voronoi-based Finite Volume Methods, Optimal Voronoi Meshes, and PDEs on the Sphere." *Computer Methods in Applied Mechanics and Engineering* **192**: 3933-3957.
- Elber, G. and E. Cohen (1997). *Filleting and Rounding Using Truncated Tensor Product Surfaces*. Proceedings of the fourth ACM symposium on Solid modeling and applications, Atlanta, GA, ACM Press.
- Farouki, R.T. *Exact Offset Procedures for Simple Solids*. Computer-Aided Design, 1985. **2**(4): p. 257-279.
- Friskien, S.F., R. Perry, A. Rockwood, J. Thouis. *Adaptively Sample Distance Fields: A General Representation of Shape for Computer Graphics*. in *Proc. of ACM SIGGRAPH*. 2000. New Orleans, LA.
- Frosyth, M. *Shelling and Offsetting Bodies*. in *Proceedings of Third Symposium on Solid Modeling and Applications*. 1995. Salt Lake City, Utah.
- Garland, M. (1999). *Quadric-Based Polygonal Surface Simplification*. Ph.D. Dissertation. *Computer Science*. Pittsburgh, PA, Carnegie-Mellon University.
- Gibson, S.F., *Calculating the Distance Map for Binary Sampled Data*. 1999, Mitsubishi Electric Research Laboratory: Cambridge, MA.
- Hartquist, E. E., J. P. Menon, K. Suresh, H. Voelcker, J. Zagajac (1999). "A Computing Strategy for Applications Involving Offsets, Sweeps, and Minkowski Operations." *Computer-Aided Design* **31**: 175-183.
- Ju, T., F. Losasso, S. Schaefer, J. Warren (2002). *Dual Contouring of Hermite Data*. Proceedings of ACM-SIGGRAPH 2002, San Antonio, TX, ACM Press.
- Kaufman, A., D. Cohen, and R. Yagel, *Volume Graphics*. IEEE Computer, 1993. **26**(7): p. 51-64.
- Kim, Y. J., G. Varadhan, M.C. Lin, D. Manocha (2003). *Fast Sweep Volume Approximation of Complex Polyhedral Models*. ACM Symposium on Solid Modeling and Applications.
- Levin, D. (1998). "The Approximation Power of Moving Least-squares." *Mathematics of Computation* **67**(224): 1517-1531.
- Lorensen, W. E. and H. E. Cline (1987). "Marching Cubes: A High Resolution 3D Surface Construction Algorithm." *Computer Graphics* **21**(4): 163-169.
- Nadler, S.B.J., *Hyperspaces of Sets*. 1978, New York: Marcel Dekker.
- Pauly, M., R. Keiser, L. Kobbelt, M. Gross. *Shape Modeling with Point-Sampled Geometry*. in *Proceeding of ACM Siggraph*. 2003. San Diego, CA.
- Pfister, H., M. Zwicker, J. van Baar, M. Gross. *Surfels: Surface Elements as Rendering Primitives*. in *ACM-SIGGRAPH*. 2000. New Orleans, Louisiana.
- Rossignac, J.R. and Aristides A. Requicha, *Offsetting Operations in Solid Modelling*. Computer Aided Geometric Design, 1986. **3**: p. 129-148.
- Satoh, T. and H. Chiyokura. *Boolean Operations on Sets Using Surface Data*. in *ACM SIGGRAPH: Symposium on Solid Modeling Foundations and CAD/CAM Applications*. 1991. Austin, TX.