A 3D surface offset method for STL-format models

Xiuzhi Qu and
Brent Stucker

The authors

Xiuzhi Qu is at the Department of Industrial and Manufacturing Engineering, University of Rhode Island, Kingston, RI, USA.
Brent Stucker is an Assistant Professor of Mechanical and Aerospace Engineering at Utah State University, Logan, UT, USA.

Keywords

Rapid prototyping, Computer aided design

Abstract

This paper presents a new 3D offset method for modifying CAD model data in the STL format. In this method, vertices, instead of facets, are offset. The magnitude and direction of each vertex offset is calculated using the weighted sum of the normals of the facets that are connected to each vertex. To facilitate the vertex offset calculation, topological information is generated from the collection of unordered triangular facets making up the STL file. A straightforward algorithm is used to calculate the vertex offset using the adjoining facet normals, as identified from the topological information. This newly developed technique can successfully generate inward or outward offsets for STL models. As with any offset methodology, this technique has benefits and drawbacks, which will be discussed in this paper. Finally, conclusions will be made regarding the applicability of the developed methodology.

Rapid Prototyping Journal
Volume 9 · Number 3 · 2003 · pp. 133–141
© MCB UP Limited · ISSN 1355-2546
DOI 10.1108/13552540310477436

Introduction

Offsets are widely used in many applications. These include tool path generation for 3D NC machining, rapid prototyping, hollowed or shelled model generation, and access space representations in robotics. In the rapid prototyping/manufacturing arena, 3D offsets are particularly important and useful as pre-process modifications to CAD geometry. Offsetting the CAD model allows one to compensate for errors caused by non-uniform shrinkage, inaccuracies in the process and the "stair-case effect" of the layer-by-layer build process. If the surface quality and accuracy of the fabricated parts is unacceptable, some kind of post processing is needed. If surface finishing is to be performed after part fabrication, a 3D offset of the model in the outward direction is required to ensure sufficient material for finishing final dimensions. Since the STL file format is widely utilized, the development of an effective STL-based 3D offset method applicable to rapid prototyping is the goal of this effort.

Offset operations can be applied to curves, surfaces or entire 3D models. By definition, an offset means moving a point, curve or surface of a 3D model by a constant distance "d" in a direction normal to the curve/surface of the model. Offset techniques for curves and surfaces have been extensively studied by Maekawa (1999) and Pham (1992). It will be more complicated to generate offsets for 3D solid models, which involve not only the geometrical issue of offsetting each individual surface in the model but also the topological issue of reconnecting these offset surfaces into a closed 3D model. Generally, offsets of 3D models are achieved by first offsetting all surfaces of the model and then trimming or extending these offset surfaces to reconstruct a closed 3D model (Farouki, 1985; Forsyth, 1995; Satoh and Chiyokura, 1991). To offset a 3D model in the STL format, the most direct method would be to offset each triangular facet with the given offset distance in its corresponding normal direction. However, this will result in intersections or gaps between the offset surfaces of two neighboring triangles.

Received: 31 August 2002
Revised: 10 December 2002
Accepted: 28 February 2003

As shown in Figure 1(a), a gap is formed between two offset surfaces F1 and F2 when the angle between them is convex. Conversely, an intersection or overlap occurs between offset surfaces, as shown in Figure 1(b), when the angle between them is concave. In order to make closed 3D models from these triangular offset surfaces, it is necessary to identify all the intersections, and then trim the surfaces on the line of intersection, and to identify all the gaps and extend the surfaces to fill them. This can be quite complex, since thousands or millions of triangular facets may exist when representing complex 3D models using the STL format.

This problem can be avoided if the vertices, instead of the triangular facets, are offset. As shown in Figure 2, when offsetting the vertices the relationship between facets will remain and there is no need to recalculate the triangle intersections. The challenge when utilizing this method is how to effectively calculate the offset vector for each vertex, taking into account the offset direction and magnitude, from all of its surrounding triangular facets.

Figure 1 (a) Gap between offset surfaces; (b) intersection between offset surfaces

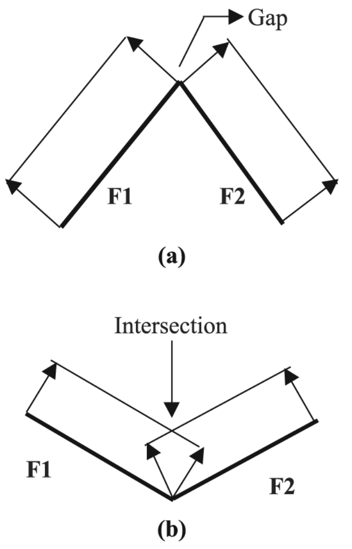
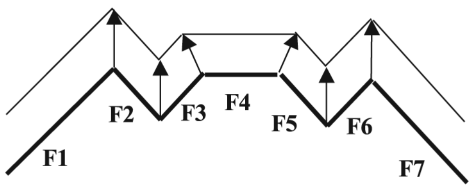


Figure 2 Offsetting vertices



The simplest method for calculating vertex offsets is to calculate an averaged vector of the normal vectors of the triangles that are connected to this vertex. For example, the offset vector $\vec{V}_{\text{Offset}}$ at vertex V_i, where there are n triangles connected at vertex V_i, can be calculated using the following equation,

$$\vec{V}_{\text{Offset}} = \frac{\sum_{j=1}^{n} \vec{N}_{i,j}}{\left| \sum_{j=1}^{n} \vec{N}_{i,j} \right|} \tag{1}$$

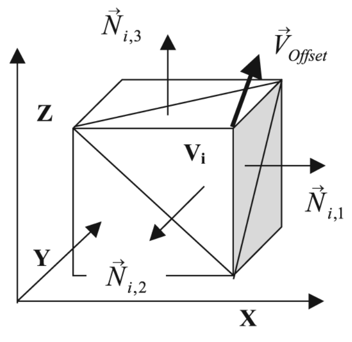
where $\vec{N}_{i,j}$ are the normal vectors of the triangles that are connected to vertex V_i.

For a vertex located on a relatively smooth surface, this method works very well. However, for a vertex located on a sharp edge – the boundary curve of two or more surfaces with large normal vector differences between them – it will not work well and may lead to unacceptable errors. As shown in Figure 3, if the offset vector $\vec{V}_{\text{Offset}}$ of vertex V_i is calculated by averaging the normals $\vec{N}_{i,1}(1,0,0)$, $\vec{N}_{i,2}(0,-1,0)$ and $\vec{N}_{i,3}(0,0,1)$, the result will be $(1/\sqrt{3}, -1/\sqrt{3}, 1/\sqrt{3})$. But the correct offset vector should be $(1,-1,1)$ for vertex V_i, which can be easily derived by calculating the intersection of the offset facets of this cube.

The same problem will occur for all the remaining vertices of this cube, and the final offset will be $d/\sqrt{3}$ when using an offset distance d. This large error is unacceptable. In order to overcome this problem, this paper presents a new method for calculating the offset vector for a vertex by using the weighted sum of the normal vectors of the triangular facets connected to the vertex as follows:

$$\vec{V}_{\text{Offset}} = \sum_{j=1}^{n} W_j \cdot \vec{N}_{i,j} \tag{2}$$

Figure 3 Offsetting vertex using averaged normal vectors



where W_j are the weighting coefficients associated with each triangular facet. A detailed method for calculating W_j and $\vec{V}_{\text{Offset}}$ will be discussed.

Recently, Koc and Lee (2002) described a vertex-based solid offset method similar to the method described in this paper. However, the vertex normal vector is calculated using the averaged vector of the normal vectors of connecting triangles in their method. As previously demonstrated, this works well only for vertices on relatively smooth surfaces but will cause large errors for vertices on sharp edges.

Since a STL file is simply a collection of unordered triangular facets of a model, topological information, particularly the identification of each unique vertex location and its neighboring triangles, needs to be generated. The remainder of this paper deals with techniques for generating this topological information and a more detailed discussion of the algorithms used for calculating vertex offsets. Some examples of application of these algorithms to various geometries will be discussed and conclusions about the developed method will be given.

Topological information generation

In order to facilitate the offset operation, topological information is derived from the unordered triangular facet list of the STL file. For a proper solid model, each triangle must have at least three neighboring triangles. The first task is to find those neighboring triangles and record their sequence numbers for each triangle in the 3D model. This can be done by finding matching edges for each edge of every triangle within the model. The implementation of this algorithm is simple but may be very time consuming if done as a one-by-one edge match operation. For example, if there are n triangles in the model, the number of edges will be $3n$. Then the total number of match operations will be

$$C_{3n}^2 = \frac{3n(3n-1)}{2},$$

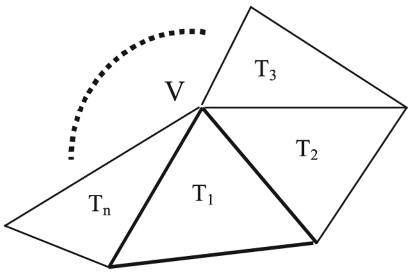
which becomes very large and will slow down the operation dramatically when offsetting a complex, high-accuracy 3D model. Thus, the hash technique has been used to speed up the matching process (Skala and Kuchar, 2000).

After identifying the neighboring relationships among triangles, each vertex is identified and a reference to its connecting triangles is recorded. The algorithm begins with an initializing operation to mark the vertices of all the triangles as unchecked. Then all the vertices are checked one-by-one and the corresponding operations are done based on the checking results as follows.

- If the vertex is unchecked, it means that it is a newly identified vertex. It is added to the unique vertex list, and its status is marked as checked. The current triangle T_1 is registered as the first connected triangle, and neighboring information is used to find all other triangles that are connected to this vertex. As shown in Figure 4, the triangle T_1 has two neighboring triangles T_2 and T_n that also share the current vertex *V*. The one on the right side (T_2) is taken as the next connected triangle, whose sequence number is recorded and the corresponding vertex (which is in the identical location to the original vertex and is not a unique vertex) is marked as checked. The right-side neighboring triangle of T_2 is then taken as the next connected triangle, and the process continues until the right-side neighbor is the first triangle, T_1.
- If the vertex is marked as checked, this vertex is a duplicate vertex and the relationship between the vertex and adjacent triangles has already been established. This vertex is skipped and not added to the unique vertex list.

As a result of these operations, an unique set of vertices for the model is generated, along with a corresponding triangle sequence number list, which represents

Figure 4 Neighboring relationship among triangles connected to vertex *V*



the connecting triangle information for each unique vertex.

In the above operation, a good STL model must be used to achieve a correct result. If there are gaps or missing triangles in the 3D model, a problem will occur, in that some triangles may have no right-side neighboring triangle. Thus the process of finding right-side neighboring triangles will not continue to completion. In this case, the process of finding left-side neighboring triangles, starting from triangle T_1, will also be performed so that as many connecting triangles of vertex V as possible can be found. The results of this operation, however, may not be acceptable as the missing triangle information may significantly affect the offset results. There are numerous software packages available commercially which find and repair gaps and missing triangles in the STL models. It is advantageous to repair a STL model using one of these packages prior to performing other operations on the STL model.

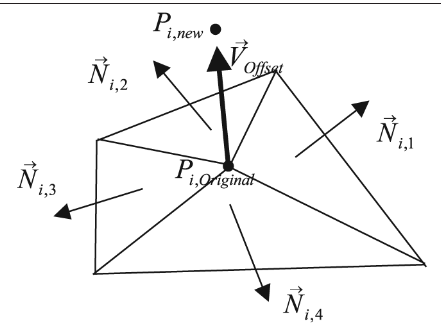
Vertex offset calculation

Equation (2) shows that the offset vector for each unique vertex can be calculated by using the weighted sum of the normal vectors of its connecting triangles. Before presenting the method for calculating the coefficients W_j, the relationship between $\vec{V}_{\text{Offset}}$ and $\vec{N}_{i,j}$ will be explored. As shown in Figure 5, $P_{i,\text{original}}$ is the vertex position before offset, and $P_{i,\text{new}}$ is the position after offset. Given the offset vector $\vec{V}_{\text{Offset}}$ and offset distance d_{Offset}, the following equation can be derived,

$$P_{i,\text{new}} = P_{i,\text{original}} + \vec{V}_{\text{Offset}} \cdot d_{\text{Offset}} \qquad (3)$$

From geometrical information, it is known that the perpendicular distance from $P_{i,\text{new}}$ to

Figure 5 Illustration of the relationship between $\vec{V}_{\text{Offset}}$ and $\vec{N}_{i,j}$



any original connecting triangular surface should be exactly the offset distance d_{Offset}, since $P_{i,\text{new}}$ is located on the intersection point of the offset surfaces of all the connecting triangular facets. This means that the following relational equation should be met.

$$(P_{i,\text{new}} - P_{i,\text{original}}) \cdot \vec{N}_{i,j} = d_{\text{Offset}} \qquad (4)$$

From equations (3) and (4), the following equation is derived,

$$\vec{V}_{\text{Offset}} \cdot \vec{N}_{i,j} = 1 \qquad (5)$$

Substituting for $\vec{V}_{\text{Offset}}$ results in the following equation,

$$\left(\sum_{j=1}^{n} W_j \cdot \vec{N}_{i,j} \right) \cdot \vec{N}_{i,k} = 1 \quad (k = 1 \ldots n) \qquad (6)$$

Expanding and rearranging the equations in a matrix form yields,

$$\begin{bmatrix} \vec{N}_{i,1} \cdot \vec{N}_{i,1} & \vec{N}_{i,2} \cdot \vec{N}_{i,1} & \cdots & \vec{N}_{i,n} \cdot \vec{N}_{i,1} \\ \vec{N}_{i,1} \cdot \vec{N}_{i,2} & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ \vec{N}_{i,1} \cdot \vec{N}_{i,n} & \cdots & \vec{N}_{i,n-1} \cdot \vec{N}_{i,n} & \vec{N}_{i,n} \cdot \vec{N}_{i,n} \end{bmatrix} \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \qquad (7)$$

These equations can then be used to find W_j ($j = 1 \ldots n$). However, when directly using the normals $\vec{N}_{i,j}$ of all the connecting triangular facets to calculate those equations, there may be no solution. For example, if two of these normals are identical, which is quite common in STL models where one flat surface is often broken down into several small triangles, then two of the equations will be exactly the same, and there will be no solution for W_j. Thus, it is necessary to eliminate duplicated normals for each vertex before proceeding with the calculation.
The equations will be solved using a computer numerical method. Because normals with similar vectors cause problems, due to computational accuracy, two normal vectors

are treated as identical if the dot product between them is between a specified value and 1 – where the specified value is slightly smaller than 1. Under this situation, a vector, which is the average of the two original vectors, is substituted in place of the original normal vectors. If the dot product of two normals is or is close to -1, then it means their corresponding triangles face in opposite directions. This is not possible with a proper STL model since any two triangles connected to the same vertex should never face in opposite directions. In this situation, the STL file should be repaired or recreated. However, for this implementation of the algorithms, the two normals having opposite directions are eliminated, and they make no contribution toward the vertex offset calculation.

Further analysis shows that the equations may have no solution if the number of normal vectors n is larger than 3 after eliminating duplicated normals. From the viewpoint of offsetting operations, this would mean that there is no common intersection point among these triangular facets after offsetting each individual facet at the given distance d_{Offset}. In order to solve this problem, an approximation method is presented when $n > 3$. In this method, the n normals are used to generate n subgroups of normal vectors $(\vec{N}_{i,1}, \vec{N}_{i,2}, \vec{N}_{i,3})$, $(\vec{N}_{i,2}, \vec{N}_{i,3}, \vec{N}_{i,4}) \ldots (\vec{N}_{i,n}, \vec{N}_{i,1}, \vec{N}_{i,2})$. Each subgroup includes exactly three normal vectors, and can be solved to find its corresponding $\vec{V}_{\text{Offset}}$. The final $\vec{V}_{\text{Offset}}$ for the vertex will be the arithmetic mean of these subgroup $\vec{V}_{\text{Offset}}$.

If there are more than three connecting triangles with different unit normal vectors for one vertex, a common intersection point among their corresponding offset facets may or may not exist, depending upon the geometrical relationship between these triangles. If offset facets intersect at one individual point, the $\vec{V}_{\text{Offset}}$ will be exactly the same for all n generated subgroups, and the arithmetic mean of them will be equal to this common vector. For this case, an exact vertex offset vector, instead of an approximate one, will be generated. Otherwise, there is no common intersection point among these offset facets, and the resulting $\vec{V}_{\text{Offset}}$ for each of the n subgroups will be different. For this case, the approximation method of using the arithmetic mean $\vec{V}_{\text{Offset}}$ of the subgroups will be acceptable if the differences among these subgroup $\vec{V}_{\text{Offset}}$ are not very large and the offset distance is small. But for the situation of big differences among the subgroup $\vec{V}_{\text{Offset}}$, serious distortion of the new offset model will occur. To solve this problem, a new approximation method is being developed, where the n subgroup $\vec{V}_{\text{Offset}}$ are used individually to offset the original vertex into n points, and then a new surface is constructed by linking these points, and finally rearranging the triangles.

After the offset vector $\vec{V}_{\text{Offset}}$ for each unique vertex is found, the new position of each vertex can be calculated using equation (3). Depending on whether an outward or inward offset is desired, d_{Offset} can be either positive or negative, respectively. All of the new position values for each unique vertex are then used to construct a new set of triangular facets which represent the desired offset model in the STL format.

Implementation

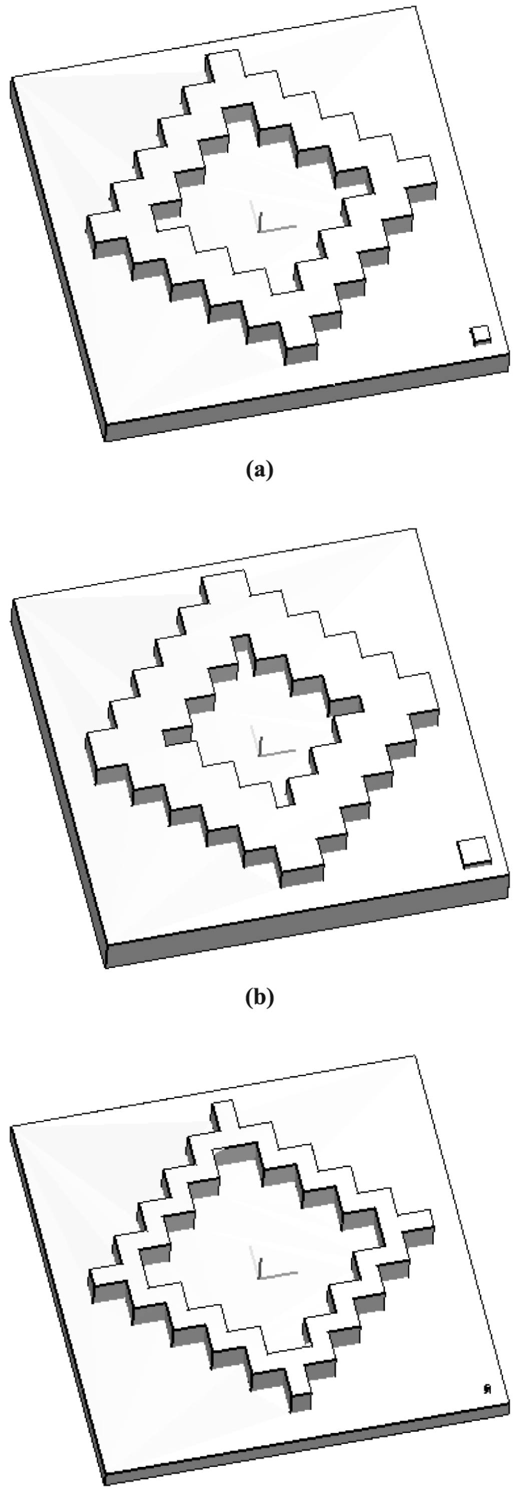
The developed technique has been implemented using Visual C++ as a module in an integrated software package for finish machining of rapid manufactured parts. To investigate the performance of this offset method, some example parts were tested.

Figure 6(a) shows a benchmark part from 3D Systems Corporation. It is a simple STL format benchmark model for machine accuracy studies. The developed software successfully generated both outward and inward offset models for using an offset distance of 0.1 in., as shown in Figure 6(b) and (c).

The second example part was designed using Pro/Engineer, and then converted into the STL file format. This part includes a variety of features, such as different types of surfaces, complicated sharp edges and holes in different orientations. The model was offset 0.03 in. outward and inward, and the results are shown as Figure 7.

The third example part, shown in Figure 8(a), is an injection molding tool cavity design provided by LaserFare, Inc. An offset value of 0.02 in. was applied, and the results are shown in Figure 8(b) and (c).

Figure 6 Illustration of offsetting. (a) Original model, (b) outward offset, and (c) inward offset



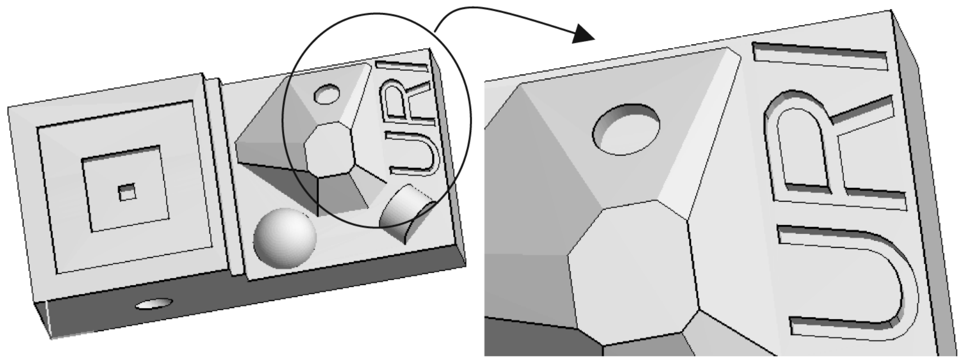
Discussion

Since the offset vector for each vertex is calculated using a weighted sum of the normal vectors of those triangular facets connected to each vertex, in some situations, as shown in Figure 9(a), the length of the offset vector is much larger than one, and the vertex offset value becomes very large compared to the offset distance d_{Offset}. This is obviously undesirable. In order to solve this problem, a vertex splitting technique can be used. For this method, the original vertex is first divided into two vertices by adding a zero area triangular facet, then two vertices which include two different groups of triangular facets are offset individually using the above-mentioned method. Figure 9(b) shows the offset results by applying this technique to the geometry shown in Figure 9(a).

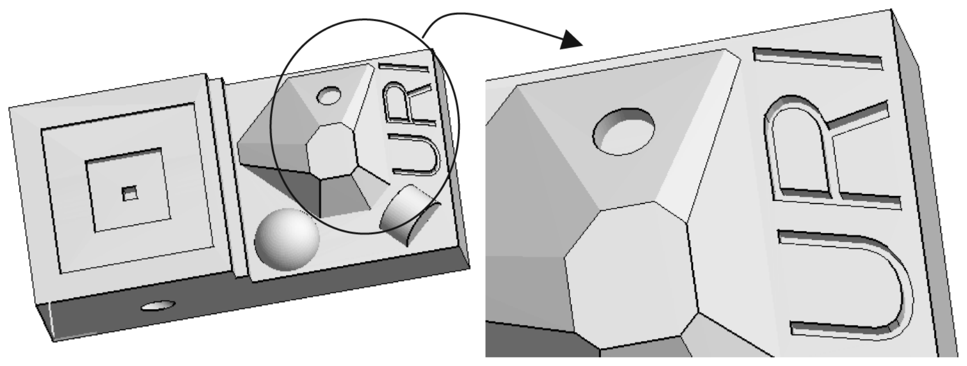
The methodology presented in this paper has focused on just solving the problem of offsetting all individual vertices of an STL model. It works well for small offset values, where local and global self-intersections normally do not happen or is not a main issue. To handle the self-intersection problem while offsetting the model with relatively large values, some post processing needs to be done.

When there are no self-intersections, the topology of the offset triangular facets should be the same as the topology of the original triangular facets. However, when self-intersections exist, the outside surface of the model will be distorted. Under this circumstance, if each newly generated triangular facet is checked, some of the facets will have experienced a dramatic change in their normal vector direction, perhaps even reversing directions, and overlaps and undesirable intersections among those newly generated triangular facets will exist. By checking for unusual facet normal variations and facet intersections, these problems can be fixed. There are two possible ways to remove these self-intersections. One is to remove the self-intersection in 3D space. The possible methods could include first finding all closed volumes by using the facet intersection information, then removing those closed volumes containing unusual facets. This is a complicated process. A different way is to try and remove self-intersections in 2D space on a slice-by-slice basis, which is a suitable method for a layer based manufacturing process. The method works by first slicing the model and then removing the self-intersections from the 2D cross-sectional curves. This 2D method is relatively easy to implement. A significant amount of work has been done in the area of removing self-intersections from 2D offset

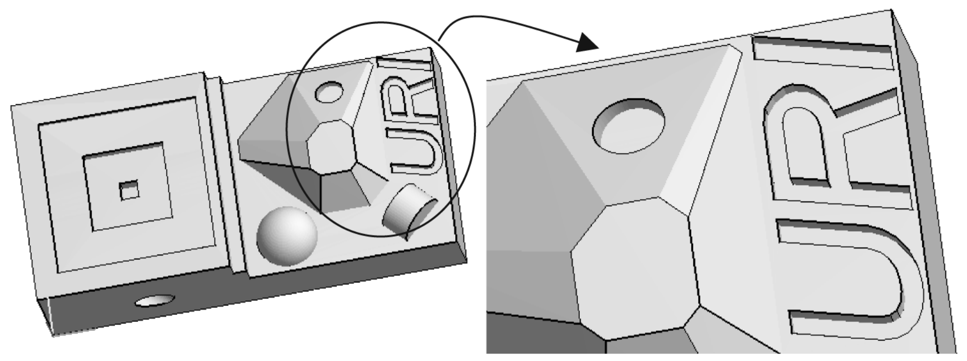
Figure 7 Second example part. (a) Original model, (b) outward offset, and (c) inward offset



(a)



(b)



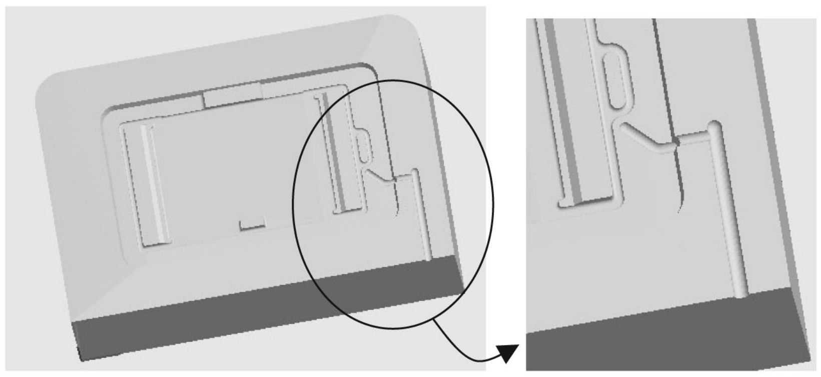
(c)

curves (Bala and Chang, 1991; Rohmfeld, 1998; Tiller and Hanson, 1984), which can be effectively used.

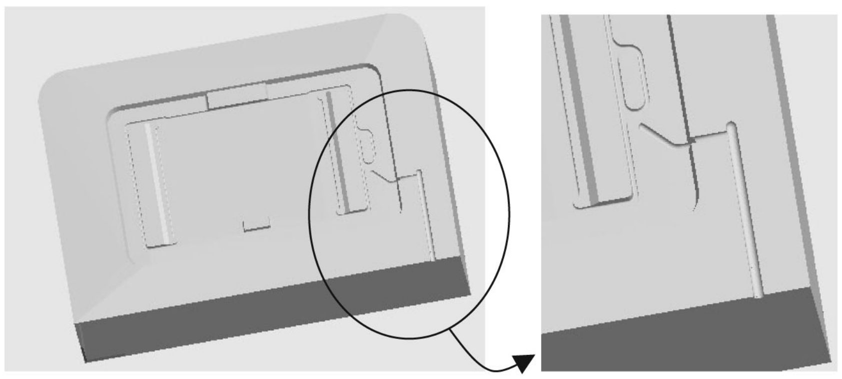
Summary

In this paper, a new offset method is presented for the offsetting of 3D models in the STL file format. This method uses a technique for offsetting each individual vertex of the model, instead of offsetting triangular facets. This eliminates the complicated and time-consuming trimming and extending operations necessary for the offsetting of facets. Through analysis it was found that a method of calculating vertex offset vectors using the average of the normal vectors of triangles connected to the vertex is not accurate enough for vertices on sharp edges. Thus, a method was developed to calculate offset vectors for vertices using the weighted sum of the normal vectors of the connecting triangles. Detailed equations and algorithms were given to calculate the weighting coefficients and offset vectors. Computer implementation showed that the developed techniques can successfully generate inward or outward offsets for STL 3D models. Some of the limitations of this methodology were presented along with some possible solution methods.

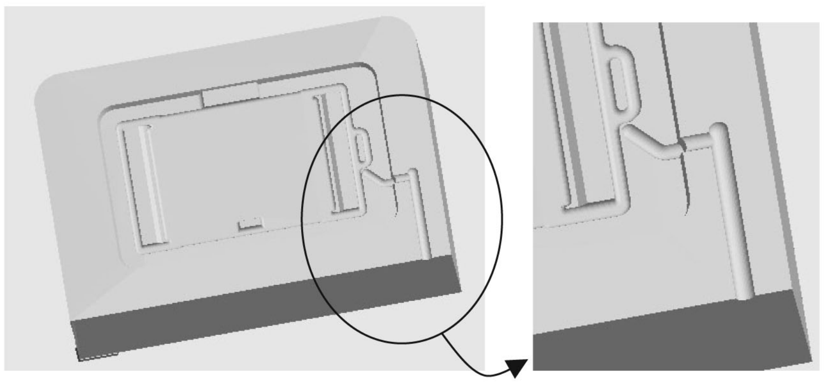
Figure 8 Third example part. (a) Original model, (b) outward offset, and (c) inward offset



(a)

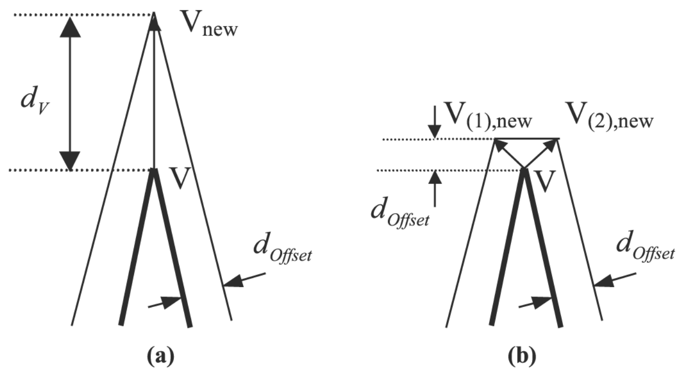


(b)



(c)

Figure 9 Illustration of vertex splitting method. (a) Normal offset, and (b) offset with vertex splitting



References

Bala, M. and Chang, T.C. (1991), "Automatic cutter selection and optimal cutter path generation for prismatic parts", *International Journal of Production Research*, Vol. 29 No. 11, pp. 2163-76.

Farouki, R.T. (1985), "Exact offset procedures for simple solids", *Computer-aided Design*, Vol. 2 No. 4, pp. 257-79.

Forsyth, M. (1995), "Shelling and offsetting bodies", *Proceedings of Third Symposium on Solid Modeling and Applications*, Salt Lake City, Utah, USA, pp. 373-81.

Koc, B. and Lee, Y.S. (2002), "Non-uniform offsetting and hollowing objects by using biarcs fitting for rapid prototyping processes", *Computers in Industry*, Vol. 47 No. 1, pp. 1-23.

Maekawa, T. (1999), "An overview of offset curves and surfaces", *Computer-aided Design*, Vol. 31 No. 3, pp. 165-73.

Pham, B. (1992), "Offset curves and surfaces: a brief survey", *Computer-aided Design*, Vol. 24 No. 4, pp. 223-9.

Rohmfeld, R.F. (1998), "IGB-offset for plane curves – loop removal by scanning of interval sequences", *Computer Aided Geometric Design*, Vol. 15 No. 4, pp. 339-75.

Satoh, T. and Chiyokura, H. (1991), "Boolean operations on sets using surface data", *ACM SIGGRAPH: Symposium on solid modelling foundations and CAD/CAM applications*, Austin, Texas, USA, pp. 119-27.

Skala, V. and Kuchar, M. (2000), "Hash function for geometry reconstruction in rapid prototyping", *Proceedings of ALGORITMY 2000,Conference on Scientific Computing*, Vysoke Tatry, Podbanske, pp. 379-87.

Tiller, W. and Hanson, E.G. (1984), "Offset of two-dimensional profiles", *IEEE Computer Graphics and Applications*, Vol. 4, pp. 36-46.