Shelling Algorithm in Solid Modeling

Dong-Ming Yan, Hui Zhang, Jun-Hai Yong,
Yu Peng, and Jia-Guang Sun

Tsinghua University, Beijing 100084, China
yandm@cg.cs.tsinghua.edu.cn

Abstract. Shelling is a powerful modeling operation in current CAD /CAM systems. In this paper, we present a new shelling algorithm for regular B-Rep solids. We first generate an initial offset solid, then correct the self-intersecting offset solid if necessary. We obtain the resultant shelling solid through a regularized Boolean operation finally. The algorithm has been implemented in a solid modeling system, and some examples are given to illustrate it.

1 Introduction

Shelling and offsetting are two related modeling operations widely used in CAD /CAM systems. Offsetting is a procedure for adding or removing a constant thickness from a solid model. Shelling is a procedure for turning a solid model into a thin-walled shell of constant thickness or variable thickness. This "shelled" solid is a hollowed version of the original model.

The research on offsetting has been carried out for a long time. Lee et al. [8] classified it into two main categories: offset geometry and offset topology. In the last decade or so, numerous researchers focused on offsetting curves or surfaces [5]. Generating offset solids belongs to the area of offset topology [1, 2, 3]. Satoh et al. [4] developed a Boolean operation algorithm on open sets. They also presented an algorithm for offset solid generation using their Boolean operation method. Lee et al. [8] proposed an algorithm for non-manifold offsetting operation. Recently, Ravi [10] developed an algorithm for the automatic offset of a NURBS B-Rep solid.

But very few studies focus on shelling a solid. Forsyth [6] presented several algorithms for offsetting and shelling operations on B-Rep solids, but it must be ensured that the offset solid does not self-intersect. Zuo [9] also developed a shelling algorithm in a solid modeling system, but his method has the same drawback as Forsyth's .

We present a solid shelling algorithm in this paper, which is more robust than Zuo's and Forsyth's algorithm. Firstly, we compute an initial offset solid of the original solid, then turn the self-intersecting offset solid into 2-manifold, if necessary. Finally we compute the shelling body through a Boolean operation.

2 Definition

In this paper, we assume that the solid is 2-manifold and in B-Rep form. Given a 2-manifold B-Rep solid $O \subseteq \mathbf{R}^3$ (three-dimensional Euclidean space). The boundary ∂O of the body O is the union of a set of faces, denoted by $F(O)$. Each face in $F(O)$ is a trimmed surface, which includes an underling surface to represent its geometric shape and a set of loops to define its boundary. A face must have a counterclockwise outer loop and some clockwise inner loops. All the loops of a face don't intersect with each other.

Generally, shelling operation needs to specify one or more pierce faces and the shelling direction. The offset distances of pierce faces are zero. Each of the non-pierce faces of the body offsets a user-defined nonzero distance, and the distances can be varied. If the shelling direction is inward, the offset direction is opposite to the normal of the face, otherwise the offset direction is along the normal of the face. The definition of shelling is illustrated in Fig. 1.

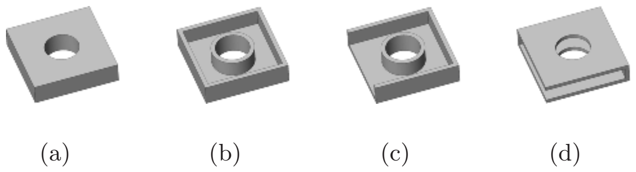

Fig. 1. Illustration of inward shelling operation. (a) The original solid. (b) Only the top face is a pierce face. (c) Both the top face and the left side face are pierce faces. (d) The left side, the front side and the cylindrical face are pierce faces

3 Methodology

At the beginning of the shelling operation, user should select the pierce face(s), set the offset distances of the non-pierce faces(the distances can be varied), and set the direction of shelling, i.e. inward or outward.

3.1 Initial Offset Solid Generation

In this step, we compute an initial offset solid, which may be topologically irregular in geometric substitution. We will solve the issue in the next step.

Firstly, we use Forsyth's method [6] to create an offset face F' of each face F. An offset surface is defined as the locus of points, which are at a constant distance d along the normal from the base surface [10]. Secondly, offset curve for each edge E is obtained by intersecting two offset surfaces of E's two adjacent faces. And then we use offset surfaces of E's two section faces to trim the intersecting curve. Thus the trimmed curve is between the two section faces. We denote edge E's offset edge as E', and attach the trimmed curve to the related offset edge E'. Then the offset edges of each offset face are connected to form offset loops L', which have the same direction as their original corresponding loops L, respectively, and the offset loops are attached to their corresponding offset faces. Finally we sew all the offset faces together to obtain an initial offset solid.

3.2 Facial Error Management

In the following step, we should judge whether the faces of the offset solid is valid. Several cases may occur on a single face. First of all, we judge whether a loop of face is self-intersecting, then we detect whether some loops of a face intersect with each other, and finally we judge whether a face has a clockwise outer loop. The offset solid becomes illegal if one or more cases occur.

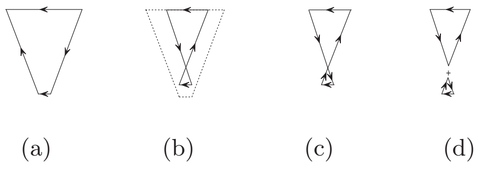

Fig. 2. Illustration of loop self-intersection

Fig. 2 illustrates the loop self-intersection problem in 2D. Fig. 2(a) is the original loop. The offset loop in Fig. 2(b) is self-intersecting(The dashed denotes the original loop). It is detected by intersecting every two edges of the loop. If the intersection point is not the endpoint of an edge, the loop is self-intersecting. We split the intersected edges at the intersection points, as shown in Fig. 2(c), and partition the loop into several sub-loops, as shown in Fig. 2(d). If the original loop is counterclockwise, we create a new face whose direction is reversed from the original face for each clockwise sub-loop. Otherwise, the original loop is clockwise, and we create a new reversed face for each counterclockwise sub-loop.

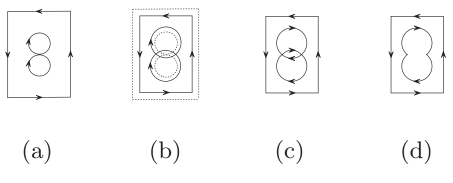

Fig. 3. Illustration of inner-inner loop intersection

Two or more loops of an offset face may intersect with each other if the face has any inner loops. Fig. 3 and 4 illustrate two cases of loop intersection. One is that several inner loops intersect with each other, and the other is that one or more inter loops intersect with the outer loop.

Firstly, we compute the intersection points among all inner loops, and then spilt the intersected edges with those points, as shown in Fig. 3 and 4. At last, we merge the intersected inner loops into one inner loop(for example, the inner loops in Fig. 3 and a 3D example in Fig. 6).

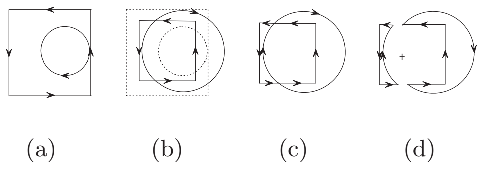

Fig. 4. Illustration of inner-outer loop intersection

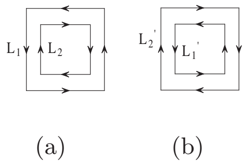

Fig. 5. Illustration of loop direction error

After merging all the intersected inner loops, we compute the intersection between the outer loop and inner loops, and then split edges similar to the above case. We partition the intersected loops into several sub-loops, which don't intersect with each other. Then create a new face for the clockwise loop with the reversed direction of the original face. Fig. 7 gives a 3D example.

Each face must contain a counterclockwise outer loop and some clockwise inner loops. The loops don't intersect with each other. Fig. 5(a) shows an original face with an outer loop L_1 and an inner loop L_2. Fig. 5(b) shows the offset face of Fig. 5(a), where L_1' and L_2' are the offset loops of L_1 and L_2, respectively. The offset face is topologically incorrect. We deal with this problem by simply reversing the direction of the face. Fig. 8 gives an example for this case in 3D.

3.3 Solid Correction and Shelling

The cases discussed above only deal with the single face of the offset solid. Moreover we should judge whether the faces of the offset solid intersect with each other. Therefore, we apply a face-face intersection procedure in this step which partitions the intersection faces by intersection curves.

The offset solid may be non-manifold now. We should eliminate the illegal parts to obtain a 2-manifold solid. For each face of offset solid, we get an arbitrary reference point $\mathbf{p}$ on the face, which does not lie on the boundary of the face. The normal of the face at the point $\mathbf{p}$ is denoted by $\mathbf{N}$. We create a ray line by using $\mathbf{p}$ as start point and $\mathbf{N}$ as its direction, and then calculate the intersection points between the ray line and all other faces. We mark the current face as undesired if the number of intersection points is odd, otherwise mark the face as reserved. We delete the undesired faces and the open faces (one or more edges of this face have only one adjacent face). The remaining faces will be sewed together to form the final offset solid. The offset solid will be $NULL$ if no face is left.

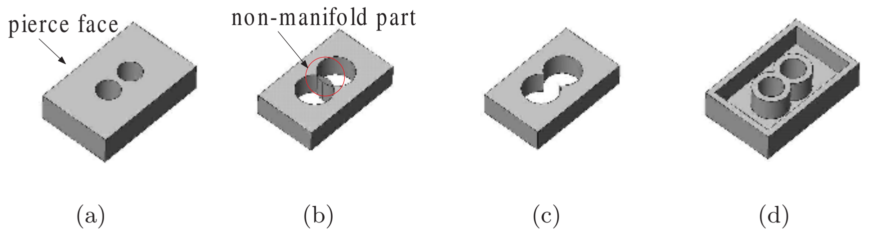

Fig. 6. Example of Inner-Inner loop intersection

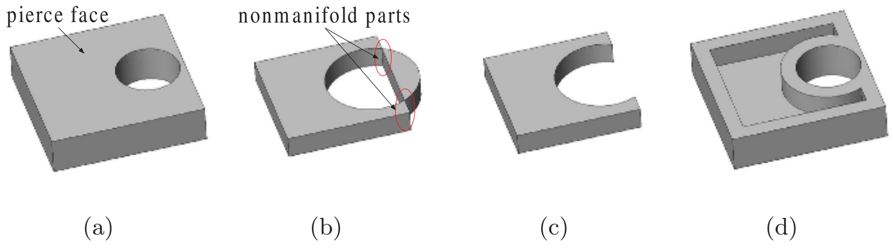

Fig. 7. Example of Outer-Inner loop intersection

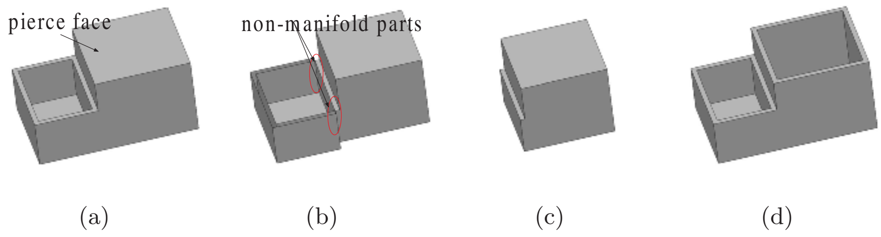

Fig. 8. Example of facial direction error

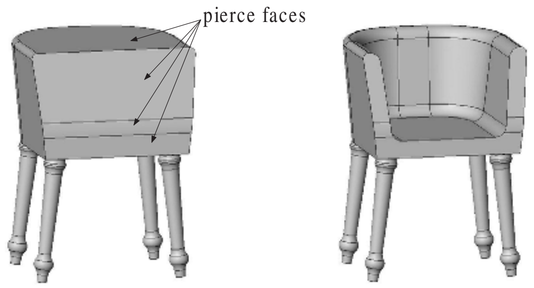

Fig. 9. Chair example

We denote the offset body after correction by O'. The shelling result of solid O is formed by the following regularized Boolean operation rules:

$$Shell(O) = \begin{cases} O, & \text{if } O' = NULL, \\ O - O', & \text{if } \textit{inward shelling}, \\ O' - O, & \text{if } \textit{outward shelling}. \end{cases}$$

4 Results and Conclusion

Fig. 6 - 8 give some examples of our algorithm. In each example, (a) shows the original solid, (b) shows the initial offset solid after loop splitting operations, (c) shows the offset solid after correction, and (d) shows the final shelling result. Fig. 9 gives a chair example using our shelling algorithm.

We present a new shelling algorithm for B-Rep solids. The loop operation used in our algorithm is similar to Gardan's algorithm [7], which reduces the complexity of the algorithm from 3D to 2D. Our algorithm has been implemented in a solid modeling system TiGems.

Acknowledgements

The research was supported by Chinese 973 Program (2002CB312106) and Chinese 863 Program (2003AA4Z3110).

References

1. Farouki, R.T.: Exact Offset Procedures for Simple Solids, Computer Aided Geometric Design. **2** (1985) 257-279
2. Rossignac, J. R., Requicha, A. A. E.: Offsetting Operations in Solid Modelling, Computer Aided Geometric Design. **3** (1986) 129-148
3. Saeed, S. E. O., de Pennington, A. and Dodsworth, J. R.: Offsetting in geometric modeling, Computer-Aided Design, **20** (1988) 50-62
4. Satoh, T., Chiyokura, H.: Boolean Operations on Sets Using Surface Data, ACM SIGGRAPH: Symposium on Solid Modeling Foundations and CAD/CAM Applications. (1991) 119-127
5. Pham, B.: Offset Curves and Surfaces: a Brief Survey, Computer-Aided Design. **24** (1992) 223-229
6. Forsyth, M.: Shelling and offsetting bodies, Proceedings of the third ACM symposium on Solid modeling and applications. (1995) 373-381
7. Gardan, Y., Perrin, E.: An algorithm reducing 3D Boolean operations to a 2D problem: concepts and results. Computer Aided Design, **28** (1996) 277-287
8. Lee, S. H.: Offsetting operations on non-manifold boundary representation models with simple geometry. Proceedings of the fifth ACM symposium on Solid modeling and applications, (1999) 42-53
9. Zuo, Z., Hu, S. M. and Sun, J. G.: A shell algorithm for solid on boundary representation. Journal of Software. **10** (1999) 761-765 (In Chinese)
10. Ravi Kumar, G. V. V., Shastry, K. G. and Prakash, B. G.: Computing constant offsets of a NURBS B-Rep. Computer-Aided Design. **35** (2003) 935-944