Computer-Aided Design 41 (2009) 830–846

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Offsetting operations on non-manifold topological models

Sang Hun Lee *

School of Mechanical and Automotive Engineering, Kookmin University, Jeongneung-Dong, Seongbuk-Gu, Seoul, 136-702, Republic of Korea

ARTICLE INFO

Article history:
Received 5 October 2008
Accepted 8 May 2009

Keywords:
CAD
Geometric modeling
Offset
Non-manifold
Wireframe
Sheet
Solid
Sheet thickening
Solid shelling

ABSTRACT

This paper describes non-manifold offsetting operations that add or remove a uniform thickness from a given non-manifold topological model. The mathematical definitions and properties of the non-manifold offsetting operations are investigated first, and then an offset algorithm based on the definitions is proposed and implemented using the non-manifold Euler operators proposed in this paper. In this algorithm, the offset elements of minimal size for the vertices, edges and faces of a given non-manifold model are generated first. Then, they are united into a single body using the non-manifold Boolean operations. In order to reduce computation time and numerical errors, the intersections between the offset elements are calculated considering the origins of the topological entities during union. Finally, all topological entities that are within the offset distance are detected and removed in turn. In addition to the original offset algorithm based on mathematical definitions, some variant offset algorithms, called sheet thickening and solid shelling, are proposed and implemented for more practical and efficient solid modeling of thin-walled plastic or sheet metal parts. In virtue of the proposed non-manifold offset operation and its variations, different offsetting operations for wireframes, sheets and solids can be integrated into one and applied to a wide range of applications with a great potential usefulness.

1. Introduction

1.1. Background and objectives

Offsetting operations add or remove a uniform thickness from a given object. For a positive offset, one adds to an object X all the points exterior to X that lie within a distance r from the boundary of X. For a negative offset, one subtracts from X all the points of X within a distance r from its boundary. Offsetting operations can be applied not only to solid models but also to sheet or wireframe models. Here, a sheet model is defined as a degenerated solid model with zero thickness, and thus it looks like a generalized surface model that allows an edge to be adjacent to more than two faces. Offsetting operations cover a wide range of potential applications. Wireframe offsetting can be used to generate sheet or solid models for pipelines of plants or ships. Sheet offsetting has been used to generate solid models for plastic or sheet metal parts with thin and uniform thickness, efficiently [1]. Solid offsetting has been used for tolerance analysis, clearance testing, NC tool path generation [2], collision-free path planning for robots [3], rapid prototyping [4], constant-radius rounding and filleting and so on [5].

Since conventional geometric modeling systems usually do not represent all of the wireframe, surface and solid models with a single representation scheme, offsetting operations for each type of model were developed separately and used in their own restricted application areas. In recent years, however, non-manifold geometric modeling systems have been developed and more widely spread in industries, which can manipulate all wireframes, surfaces, solids, and their mixtures with a single unified topological representation. Therefore, if we develop offsetting operations for non-manifold topological (NMT) models, three types of operations can be integrated into one in a single environment to be used in many application areas. For instance, these operations can be used in detailed design to create a solid from an abstract model that is usually generated as a mixture of wireframes and sheets in conceptual design [6]. In addition, if we give some variances to the original offset algorithm, thin-walled solid models for plastic or sheet metal parts can be created easily from the sheet models.

To take great advantages of non-manifold offsetting, in this paper, the mathematical definitions and properties of the non-manifold offsetting operations are studied first, and then the positive and negative offset algorithms are proposed and implemented using the non-manifold Euler and Boolean operations. In addition, some variant offset algorithms called sheet thickening and solid shelling are proposed and implemented for more practical and efficient solid modeling of thin-walled plastic or sheet metal parts. Note that this work is mainly focused on topological operations for offset models, although most of the previous research has

* Tel.: +82 2 910 4835; fax: +82 2 910 4839, 82 2 916 0701.
E-mail address: shlee@kookmin.ac.kr.

0010-4485/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2009.05.001

been concentrated on geometric calculations for developing offset curves and surfaces.

The remainder of this paper is organized as follows. Section 1.2 surveys the related work. Section 2 describes the formal definitions and mathematical properties of non-manifold offsetting. Section 3 presents a set of Euler operators for non-manifold models based on extended Euler Poincare formula. Section 4 describes an offset algorithm for non-manifold models, which is implemented using a set of non-manifold Euler and Boolean operations. Section 5 presents variant offset algorithms for wireframe and sheet models, which include sheet thickening and solid shelling functions to allow users to create more practical offset solids from sheets and solids. Finally, Section 6 presents the conclusions of this paper.

1.2. Related work

A lot of research related with offsetting has been carried out for over three hundred years, and they may be classified into two main categories: offset geometry and offset topology. The area of offset geometry deals with the exact or approximate methods for generating offset curves and surfaces. Pham [7] carried out a literature survey on offset curves and surfaces up to 1992, and Maekawa [8] overviewed the literature after 1992 focusing on five active areas of research on offsets: representing exact offsets in Bézier/B-spline format, approximations, self-intersections, geodesic offsets and general offsets. The area of offset topology deals with the development of topological operations for generating offset solids or converting sheets into solids in geometric modeling systems. This paper belongs to the research category of offset topology because it concentrates on the description of topological operations for offsetting a non-manifold model. Although no publications on non-manifold offsetting was found, much research on offsetting operations on solids and sheets has been made and published. Since offsetting operations on non-manifold objects encompass those on solids, sheets and wireframes, previous works on solid and sheet offsetting will be reviewed as related work instead.

Rossignac and Requicha [5] first attempted to incorporate offsetting operations into solid modelers. They introduced a family of operations called solid offsetting (s-offsetting). The regularized solid offset of a regular set S by a distance r can be seen as the volume swept by a solid sphere of radius r as its center moves throughout the set S. They discussed some properties of s-offsetting operations, and included them in an extended form of CSG called CSGO, in which offsetting operations are represented as non-terminal nodes in CSG trees. This method allows CSG modelers to get around the problem that, if a solid is some Boolean combination of primitives, the offset operator cannot be expressed as the same Boolean combination of the offsets of the primitives.

Farouki [9] proposed an offsetting procedure for simple solid primitives of extrusion and of convex polyhedrals. Offset patches for each face, edge and vertex are generated first, and then they are merged together to compose the boundary of an offset solid. Farouki described exact offset procedures for simple solids of three types: convex planar polyhedra, solids of revolution and solids of linear extrusion with simple profile curves. The simple profile curves are linear and circular arcs. In order to construct the offset to a simple solid, the topology of the solid is first resolved as faces, edges and vertices. For each face, a face offset element is generated by offsetting the face by a vector $r\,\mathbf{n}$, where $\mathbf{n}$ is the outward normal to that face, and r is the offset distance. For each edge, an edge offset element is generated by a translation sweep of an arc along the edge curve. For each vertex, a spherical vertex offset element is generated as a set of triangular spherical patches bounded by circular arcs. The face, edge and vertex offset elements above are guaranteed to match precisely at their boundaries and form a complete, designed offset surface for the simple solids. However, this algorithm cannot be applied to a solid with concave edges or vertices or a solid with complex curves and surfaces.

Saeed et al. [10] also attempted to introduce a class of offsetting operations into solid modeling systems. Their mathematical formulation for offsets is based on the concept of open ball neighborhoods in an n-dimensional space. Their definition of an offset is equivalent to that given by Rossignac and Requicha [5]. By using the neighborhood function, an offset solid can be constructed from the offsets of its boundary sets in lower dimensions. This mathematical formulation provides a coherent framework for synthesizing an offset solid.

Satoh and Chiyokura [11] defined the open set to represent the partial boundary of a solid, and developed an algorithm for Boolean operations on open sets. As an application of the open-set Boolean operations, he also proposed algorithms for offset solid generation and self-intersecting solid correction. Their offset procedure is composed of the following steps. First, open sets with offset surfaces are generated for all faces of a given solid. Next, these sets are united using the open-set Boolean operations. Finally, if there are any gaps between the open sets, new faces are generated to close the gaps. If the resulting offset solid is self-intersecting, the correction procedure is applied. However, they did not describe a detailed method of eliminating gaps that are caused by convex edges or vertices.

Forsyth [12] proposed algorithms for offsetting and shelling operations on B-rep solids, and also implemented them with the modeling capabilities of SolidDesigner. In his offset algorithm, an offset solid is generated as follows. First, offset surfaces for each face are generated. Secondly, offset curves for each edge are generated by intersecting two offset surfaces of the adjacent faces. Thirdly, offset positions for each vertex are also obtained by intersecting offset curves. Fourthly, the offset surfaces, curves and positions are attached to the corresponding faces, edges and vertices, respectively. Finally, if positive offsetting, all convex edges are blended with a radius of a given offset distance r, or, if negative offsetting, all concave edges are blended. The outstanding feature of this algorithm is the substitution of the geometric entities of a given solid with the offset ones and then the blending of the concave or convex edges. However, the offset solid can show, in geometric substitution, topologically irregularity, for which he did not suggest any topology correction procedures.

Recently, Kumar et al. [13] proposed an approach for the automatic offset of a NURBS B-Rep, which can be used for a class of manifold B-Reps. The approach offsets each of the trimmed surfaces (faces) of the B-Rep and then removes the gaps and intersections between offset faces automatically, if any. The offset B-Rep is then created by sewing all the updated offset faces. The approach can generate both positive and negative constant offsets of a B-Rep. The approach works under the assumption that the number of faces in both base and offset B-Reps is the same. It further assumes that the faces are at least G^1 continuous and non-self-intersecting after constant offset. The approach has been applied to composite laminate modeling to make offsets of many lay-up surfaces.

The commercial geometric modeling kernels such as Parasolid and ACIS also provide functionality for solid offsetting. ACIS [14] provides the body offset operation that can be applied not only to the whole body but also to the specific faces of the body. In both cases, the surfaces of the selected faces are replaced with offset surfaces by using the tweak operation. The tweak operation replaces the surface under a face with any other surface supplied by the user, provided that the surface intersects appropriately and that necessary topology changes are supported. However, faces with radial surfaces cannot be offset. If they exist, they are removed and the resulting wound is healed by the surrounding face surfaces. Parasolid [15] also provides the offsetting operations analogous to

ACIS's. If the offset surfaces do not meet, the underlying surfaces are extended to enable the offsets to intersect. Recently, however, a new functionality for the automatic detection and repair of self-intersecting geometry has been included. If the creation of offset surfaces introduces self-intersections, the system removes such self-intersections, and heals any resulting holes either by extending and intersecting neighboring offset surfaces, or by filling them with an approximate surface that joins the neighboring regions within the given offset tolerance.

In addition to solid offsetting, research on sheet offsetting has been done in order to more efficiently generate solid models of thin and constant thickness from a given sheet model. In this method, a sheet model is first created for one side of the part or for a medial surface of the part, and then a thin-walled solid is generated by adding volume to the sheet by a given thickness. This method is usually called sheet thickening, which is very useful in modeling plastic or sheet metal parts. To facilitate the discussion, let us define a sharp edge as an edge that is adjacent to only one face and constitutes the boundary of a sheet, and define a thickness face as a face that is a thin strip-like face connecting the inside and the outside wall. In sheet thickening approaches, the sharp edges in a sheet body are converted to the thickness faces.

Stroud [16] suggested a method to convert a sheet into a solid for a given thickness. In his system, a sheet object is represented as a degenerate B-rep solid model, in which the thickness faces are represented by sharp edges. In the transformation procedure, the sharp edges and their end vertices are first split in order to make topological data for the thickness faces. Then, the geometry for each vertex, edge, and face is calculated and assigned to the corresponding topological entity. However, this method may result in unacceptable, impractical solids, as he did not suggest any verification and correction methods for self-intersections of the converted solid.

Lee and Kwon [17] proposed another sheet modeling and transformation approach, which was revised by Lim and Lee [1]. They adopted the winged edge data structure as a topological framework for representing sheet objects. From a schematic viewpoint, this approach is very similar to Stroud's, as they both adopt solid data structures in order to represent sheet objects, which are regarded as degenerated solids. However, there is a difference between the two approaches with respect to the storage of topological data for the degenerated solids. In Lee and Kwon's work, a sheet model has full topological data for the corresponding solid model, including the thickness faces, whereas in Stroud's work, the thickness faces are degenerated into sharp edges. Therefore, in Lee and Kwon's method, sheet thickening is basically just replacing the geometry of each topological entity with the offset geometry. Of course, however, this method may cause unacceptable, impractical solid models, as the topology of the acceptable solid is not coincident with the topology of the sheet body, and self-intersection of the offset solid is not considered. To find a way to overcome these drawbacks, they investigated failure cases and suggested topological correction methods for unacceptable solids, although such cases are limited, and they did not consider the self-intersection problem. On the other hand, in order to facilitate the development of high-level sheet modeling capabilities, they also proposed a set of sheet Euler operators that are like the macros of the standard Euler operators used for solid modeling. However, as their sheet Euler operators are not a complete set of topological operators for manipulating topological entities of a sheet model, the standard Euler operators must still be used together.

The methods of Stroud [16] and Lee and Kwon [17] have a common problem in that their topological data structures do not store and provide proper information about the adjacency relationships of the topological entities. This is because sheet objects are described by a solid data structure, even though sheets are fundamentally non-manifold objects. This deficiency makes it difficult to develop algorithms for sheet modeling and thickening capabilities.

The current solid modeling systems usually provide sheet thickening operations which can convert a sheet model to a solid model by offsetting it in one or both normal directions [14]. For example, ACIS provides the *api_sheet_thicken* function, and Parasolid offers the *PK_BODY_thicken_3* function. However, these functions have several limitations currently. First, the sheet must be manifold, single lump, and single shell. Next, the normals of adjacent faces must be consistent. Therefore, it cannot generate an offset for an arbitrary non-manifold body. For example, this T-shape body cannot be offset exactly. In addition, ACIS cannot fix the self-intersecting surfaces either, while Parasolid can do recently. Similar limitations exist in the existing commercial systems.

There is another way to model a thin-walled solid efficiently through a method called solid shelling. The solid shelling operation creates a thin-walled solid model by digging out the inside volume from the solid model of the outside shape. Although this operation is not suitable for modeling sheet metal parts with many bending and hole features, it is very useful for modeling plastic parts in the shape of bowls.

Lee and Lee [18] first developed solid shelling capabilities using a commercial solid modeling kernel, ROMULUS. In their algorithm, a solid object for an outer or inner wall is modeled first. Then, another solid for the other side of the wall is created by copying the original solid and then shrinking the copied solid by a given thickness. Finally, a thin-walled solid is generated by subtracting the offset solid from the original solid using the Boolean operations. In this method, however, the authors did not consider any correction process for illegal topology caused by self-intersection in the shrinking step.

Forsyth [12] also proposed an algorithm for shelling operations on B-rep solids, and implemented the algorithm with the modeling capabilities of SolidDesigner. His algorithm is the same as that of Lee and Lee [18] except that it checks to correct the self-intersection of the shelled body and rounds the convex edges and vertices of the offset solid.

Recently, in rapid prototyping, the shelling method has received much attention because it can reduce the building time and expensive RP material consumption significantly by constructing a hollowed prototype instead of a solid model. For hollowing a solid model, several methods have been developed, and they can be classified into four categories [19]: spatial enumeration techniques such as the octree and the voxel model [20–22]; constructive solid geometry (CSG) offsetting methods [4]; curve offsetting methods [19,23]; and surface offsetting methods [24,25]. In the enumeration methods, the inner wall of a thin-shell solid model is obtained by using a sub-boundary octree [21] or voxel elements [20,22]. These methods are computationally effective, but can cause an internal staircase effect. In the CSG offsetting method [4], a thin-shell solid is obtained by subtracting the original solid from its offset counterpart. However, this method is only applicable to CSG solids. In the curve offsetting method proposed by Ganesan and Fadel [23], by slicing the original part, external cross-sectional curves are obtained first, and then two-dimensional curve offsetting is carried out to find the internal cross-sectional curves. Finally, the interior surface is constructed from the internal curves. Since the curve offsetting method employs two-dimensional curve offsetting instead of three-dimensional surface offsetting, this approach is relatively easy to implement. However, it cannot guarantee uniform wall thickness because the wall thickness depends on the surface normals. To overcome this drawback, Park [19] developed a new method to generate internal contours directly from the external contours. The sum of the circle swept volumes of

external contours represents the offset model. It is possible to compute an internal contour of a layer by slicing the circle swept volumes affecting the layer. Actually, it is not necessary to compute the actual circle swept volumes because we can generate the sliced curves with a simple combination of two-dimensional geometric operations. Since the algorithm is based on well-known two-dimensional geometric algorithms, it is efficient and easy to implement. As for the surface offsetting methods, Koc and Lee [24] proposed a non-uniform vertex offsetting method based on an averaged surface normal method to hollow out a solid model, and Qu and Stucker [25] proposed a vertex offset method, in which topology information is built first and then the vertex offset is calculated using the weighted sum of the normals of the adjoining facets. However, in the solid hollowing method for RP processing, the input geometric model is a polyhedral solid model described in the STL format, and the output is also a polyhedral model offset by a given thickness. Therefore, this method is not adequate for obtaining the exact offset of a general solid model with quadratic and freeform surfaces.

2. Mathematical definitions and properties of non-manifold offsets

2.1. Definitions of non-manifold models

In this paper, the Euclidean cell complex is selected as a proper mathematical model for a non-manifold object. In an n-dimensional Euclidean space, E^n, the n-dimensional cell (n-cell) is defined as a bounded subset of E^n, which is homeomorphic to an n-dimensional open sphere. If a set of a finite number of cells, X, satisfies the three conditions below, X is defined as a Euclidean cell complex [26].

$$X = \cup_{\lambda=\Lambda} e_\lambda, \tag{1}$$

$$[e_\lambda] - e_\lambda \subset \{e_\mu | dim(e_\mu) < dim(e_\lambda), \mu \in \Lambda, \lambda \in \Lambda\}, \tag{2}$$

$$e_\lambda \cap e_\mu = \phi, \quad \lambda \neq \mu, \mu \in \Lambda, \lambda \in \Lambda, \tag{3}$$

where e_λ denotes an n-cell, Λ a universal set of n-cell indices, $dim(e_\lambda)$ the dimension of e_λ, and $[e_\lambda]$ the closure of e_λ that includes an n-cell as well as its boundary. The first condition means that an n-dimensional cell complex is a collection of 0-cells, 1-cells, 2-cells, …, and n-cells. The second condition means that the boundary of each cell consists of lower-dimensional cells. This condition ensures that a cell complex is always closed and does not contain any unclosed topological entities. The third condition means that no topological entities intersect each other.

In this paper, since we deal only with non-manifold objects in E^3, their models are always three-dimensional cell complexes. The non-manifold modeler adopted in this paper supports the modeling capabilities for these cell complexes. Its brief description is given in the following section. When comparing n-cells with topological entities of non-manifold modelers, 0-cells, 1-cells, 2-cells and 3-cells correspond to vertices, edges, faces and regions, respectively. In our non-manifold modeler, all three-dimensional spaces are represented by the region entities, whether or not they are filled with material. The material information is stored as an attribute for a region. In this paper, an empty region is called a *void* region and a material-filled region is a *filled* region.

2.2. Definitions and properties of non-manifold offsets

2.2.1. Definition of non-manifold offsets

If X denotes a non-manifold model defined as a three-dimensional cell complex and $X \oplus r$ denotes the positive offset of X by a positive distance r, then the positive offset of X, X_o, is

$$X_o = X \oplus r = \{\mathbf{p}_o | \exists \mathbf{p} \in X, \|\mathbf{p}_o - \mathbf{p}\| \leq r\}. \tag{4}$$

Note that, if X is empty, X_o is also empty. An equivalent definition is as follows:

$$X \oplus r = \cup_{p \in X} B^*(\mathbf{p}, r) \tag{5}$$

where $B^*(\mathbf{p}, r) = \{\mathbf{p}_o | \|\mathbf{p}_o - \mathbf{p}\| \leq r\}$ and denotes a closed ball of radius r centered at $\mathbf{p}$. This can be understood as the volume swept by a solid sphere of radius r as its center moves throughout X.

The complement of a non-manifold model can be obtained easily by exchanging the *void* and *filled* attributes of the regions with each other. If the negative offset of X by a distance r is denoted by $X \ominus r$ and the complement of X is denoted by X^C, the negative offset is defined as follows:

$$X \ominus r = (X^C \oplus r)^C. \tag{6}$$

Fig. 1(a) shows a cross-section of a simple non-manifold model that is composed of an L-shaped solid and a sheet. Its positive offset is shown in Fig. 1(b) and its negative offset in Fig. 1(c).

2.2.2. Mathematical properties of non-manifold offsets

Various mathematical properties of solid offsets were discussed by Rossignac and Requicha [3]. Most of them also appear in offsets of non-manifold models. The equality and inclusion relations between solid offsets also appear in non-manifold offsets. That is, if $A = B$, then $A \oplus r = B \oplus r$ and $A \ominus r = B \ominus r$. In general, however, neither $A \oplus r = B \oplus r$ nor $A \ominus r = B \ominus r$ implies that $A = B$. If $A \subset B$, then $A \oplus r \subset B \oplus r$ and $A \ominus r \subset B \ominus r$.

In addition, non-manifold models defined as a three-dimensional Euclidean cell complex are algebraically closed under offsetting operations. That is, if X is a three-dimensional Euclidean cell complex, then its offsets $X \oplus r$ and $X \ominus r$ are also three-dimensional Euclidean cell complexes. This implies that one can add offsetting operations to a non-manifold modeler and be sure that the resulting sets are valid models and therefore can be used in the system as inputs for further operations.

However, the positive and negative offsetting operations are not generally commutative, and the two operations should not be thought of as inverses because they have the following properties:

$$(X \ominus r) \oplus r \subset X \subset (X \oplus r) \ominus r. \tag{7}$$

Actually, we can obtain the rounding and filleting effects of non-manifold models by combining the positive and negative offsetting operations. That is, $(X \ominus r) \oplus r$ rounds the convex edges and vertices of the given object, while $(X \oplus r) \ominus r$ fillets the concave edges and vertices [3].

The topological boundary of an expanded model $X \oplus r$ or a shrunk model $X \ominus r$ is included in the set of points that are at a distance r from X. If $d(\mathbf{p}_o, X)$ denotes the distance from a point outside X, $\mathbf{p}_o$, to the closest point on X, this relationship can be written as follows:

$$\partial(X \oplus r) = \{\mathbf{p}_o | d(\mathbf{p}_o, X) = r\} \tag{8}$$

$$\partial(X \ominus r) = \{\mathbf{p}_o | d(\mathbf{p}_o, X^C) = r\}, \tag{9}$$

where

$$d(\mathbf{p}_o, X) = \min \|\mathbf{p}_o - \mathbf{p}\|, \quad \mathbf{p} \in X \tag{10}$$

and

$$d(\mathbf{p}_o, X) = d(\mathbf{p}_o, \partial X). \tag{11}$$

The boundary of the offset model has the following property, as in a solid offset [3]:

$$\partial(X \oplus r) \subset \partial(\partial X \oplus r). \tag{12}$$

This property is useful for constructing a superset of the boundary of the offset model. $\partial X \oplus r$ can be obtained by uniting the offsets of all faces, edges and vertices of a given model X. Since the boundary of the resulting offset model $\partial(X \oplus r)$ is included in $\partial(\partial X \oplus r)$, $\partial(X \oplus r)$ can be obtained by eliminating unnecessary topological entities from $\partial(\partial X \oplus r)$. Based on these mathematical definitions and properties, an offset algorithm for a non-manifold model is suggested in Section 4.

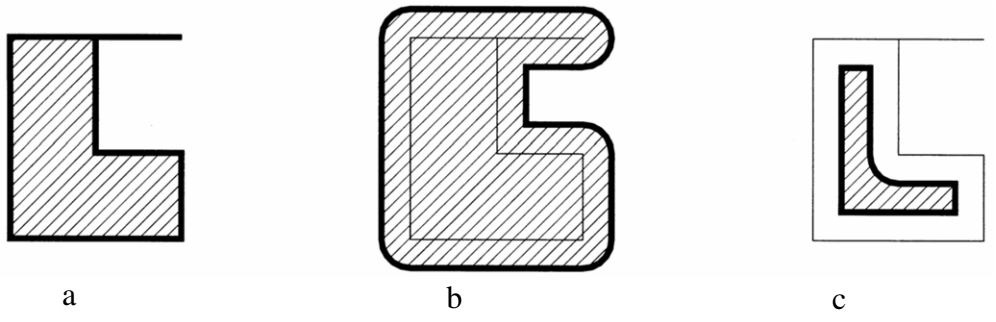


Fig. 1. Non-manifold offsetting operations: (a) a simple non-manifold object composed of a sheet and an L-shaped solid; (b) a positive offset; (c) a negative offset.

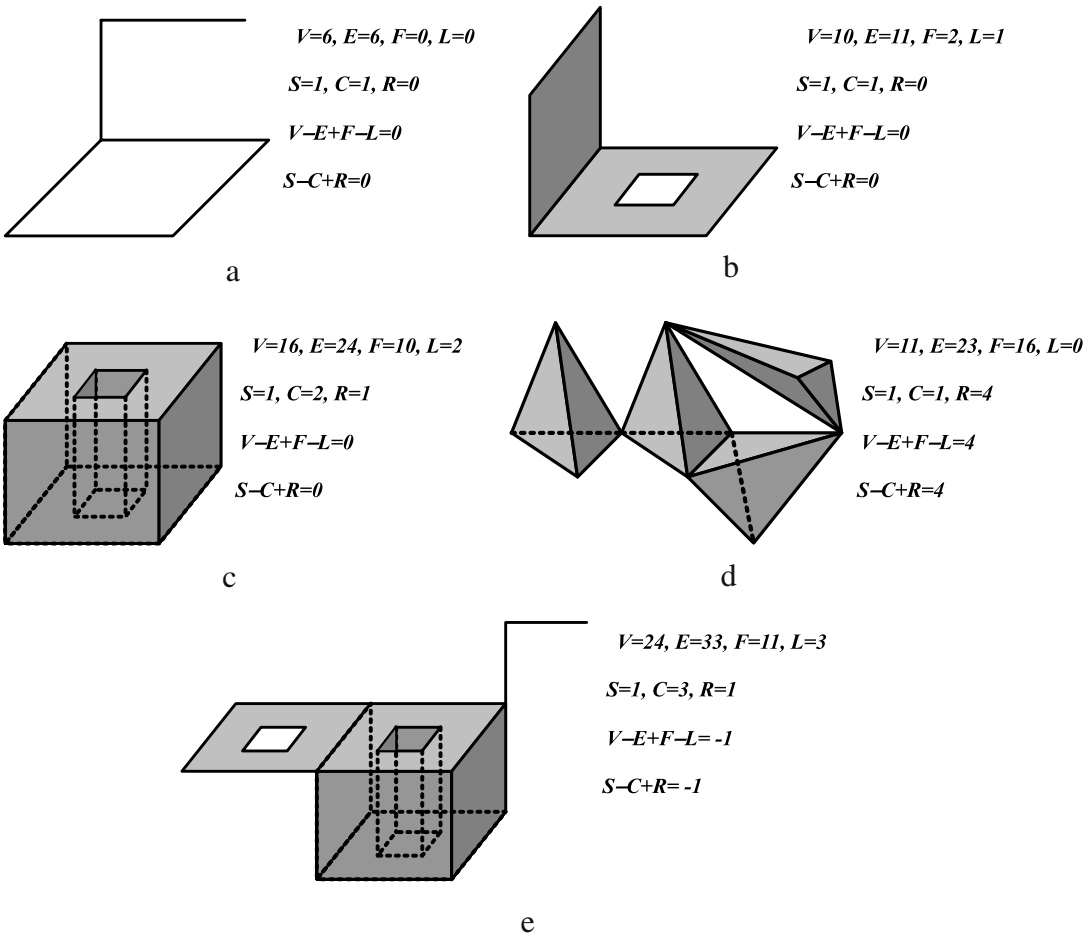


Fig. 2. Examples of the application of the extended Euler–Poincare formula.

3. Non-manifold topological operators

The algorithms for the offsetting operations proposed in this paper are described using a set of the basic non-manifold topological (NMT) operators including the extended Euler operators for non-manifold objects. Euler operators in B-rep solid modeling guarantee the integrity of the model, insulate geometric modeling functionality of higher level from the specifics and complexities of underlying data structures, and allow easy implementation of undo commands, as each operator has its inverse operator [27]. To take advantage of these useful features, considerable effort has also been made to derive an Euler–Poincare formula generalized for three-dimensional non-manifold models, and define a set of Euler operators based on the formula [26,28–32].

This paper proposes a set of NMT operators that includes a set of Euler operators based on the following equation, which is an extended Euler–Poincare formula for three-dimensional non-manifold models proposed by Yamaguchi and Kimura [28].

$$V - E + F - L = S - C + R \tag{13}$$

where V, E, F, L, S, C, and R are the numbers of vertices, edges, faces, hole loops, void shells, non-manifold cycles, and regions, respectively. The non-manifold cycle can be interpreted as an independent cycle that is not converted to a face in a wireframe composed of vertices and edges, or a circle that cannot be retracted to a point on a surface. Note that the infinite region is not counted in this formula. Fig. 2 shows some examples to which Eq. (13) is applied.

Table 1
Non-manifold Euler operators and their base vectors.

Category	Name	Meaning	V	E	F	L	S	C	R
Basic NMT operators	MMR	Make model and region							
	KMR	Kill model and region							
Basic Euler operators	MVS	Make vertex and shell	1	0	0	0	1	0	0
	KVS	Kill vertex and shell	−1	0	0	0	−1	0	0
	MEV	Make edge and vertex	1	1	0	0	0	0	0
	KEV	Kill edge and vertex	−1	−1	0	0	0	0	0
	MEC	Make edge and cycle	0	1	0	0	0	1	0
	KEC	Kill edge and cycle	0	−1	0	0	0	−1	0
	MFKC	Make face, kill cycle	0	0	1	0	0	−1	0
	KFMC	Kill face, make cycle	0	0	−1	0	0	1	0
	MFR	Make face and region	0	0	1	0	0	0	1
	KFR	Kill face and region	0	0	−1	0	0	0	−1
	MVL	Make vertex and loop	1	0	0	1	0	0	0
	KVL	Kill vertex and loop	−1	0	0	−1	0	0	0
Auxiliary Euler operators	SEMV	Split edge, make vertex	1	1	0	0	0	0	0
	JEKV	Join edge, kill vertex	−1	−1	0	0	0	0	0
	MEF	Make edge and face	0	1	1	0	0	0	0
	KEF	Kill edge and face	0	−1	−1	0	0	0	0
	KEML	Kill edge, make loop	0	−1	0	1	0	0	0
	MEKL	Make edge, kill loop	0	1	0	−1	0	0	0
	KEMS	Kill edge, make shell	0	−1	0	0	1	0	0
	MEKS	Make edge, kill shell	0	1	0	0	−1	0	0

Some terms needed for the description of Euler operators are defined as follows. A manifold edge is an edge with two incident faces. A non-manifold edge is an edge that is not a manifold edge. A wire edge is a non-manifold edge that is not associated with any face. A sharp edge or a lamina edge is a non-manifold edge that is associated with only one face. If one of the vertices of an edge is adjacent to no other edge, the edge is called a strut edge. A strut edge can be either a manifold edge or a non-manifold edge. An isthmus edge is a manifold edge that is adjacent to the same face and its vertices are adjacent to other edges. Usually an isthmus edge connects an inner loop to another loop of a face. A single-vertex shell is a shell that consists of only an isolated vertex. A single-vertex loop is a loop that consists of only an isolated vertex.

All of the NMT operators in our system are listed in Table 1. According to the naming convention for Euler operators, the following abbreviations are used to represent each operation and topological entity: M (make), K (kill), S (split), J (join), V (vertex), E (edge), F (face), L (hole loop), S (void shell), C (non-manifold cycle), and R (region). As shown in Table 1, they are categorized into three groups. The first group is a pair of the basic NMT operators for initially creating a model and its infinite region, and finally deleting them. The second group is six pairs of basic Euler operators, which are a minimal set of independent Euler operators derived from the formula. Since there are six independent variables in Eq. (13), at least six independent Euler operators and their six inverse operators are required to manipulate the topological structure of non-manifold models. The third group is four pairs of Euler operators supplementing the six pairs of basic operators and facilitating the implementation of high-level modeling operations. The details of these operators including the specification of the input and output arguments are described with figures in the following subsections.

Our kernel modeler was implemented based on the Partial Entity Structure [33], which is a compact but efficient non-manifold boundary representation proposed by the authors. Since all of the high-level modeling operations, such as sweeping and Boolean operations, have been implemented using this set of Euler operators in order to build our kernel modeler, this set has been verified to be sufficient for the development of a non-manifold modeler. The non-manifold offsetting operations suggested in this paper were also implemented using the Euler operators listed in Table 1.

MMR (Model ***M*, Region ** *R*)
KMR (Model **M*)

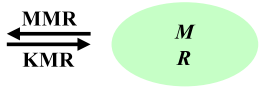


Fig. 3. MMR and KMR operators.

MVS (Region **R*, Vertex ***V*, Shell ****S*, Point **Pt*)
KVS (Shell **S*)

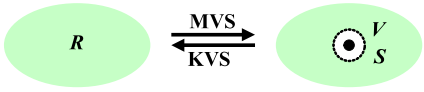


Fig. 4. MVS and KVS operators.

3.1. Basic NMT operators

MMR, KMR

The operator MMR creates, from scratch, an instance of the model, *M*, that has just one infinite region, *R*, as illustrated in Fig. 3. It is the first operator used in any topology construction. The inverse of MMR is called KMR, and it destroys an instance of the model identical to that created by MMR. The input model for KMR should include only one infinite region without any void shell. The C++ style of the function prototypes for MMR and KMR are shown below. A single pointer argument denotes an input whereas a double pointer argument denotes an output.

3.2. Basic Euler operators

MVS, KVS

The operator MVS creates a single-vertex shell, *S*, at a given position, *Pt*, in a given region, *R*, as described in Fig. 4. The isolated vertex created by MVS, *V*, can be used as a starting point for subsequent construction of additional topological features. The inverse of MVS is KVS, which destroys the specific single-vertex shell. The effect of MVS and KVS is depicted in Fig. 4.

MEV, KEV

The operator MEV creates a new vertex, V_2, first, and then creates a new edge, *E*, connecting V_2 with a given vertex, V_1, as

MEV (Vertex *V_1, Entity *$Parent$, Edge **E, Vertex **V_2, Curve *Cv)
KEV (Edge *E, $Vertex$ * V_2)

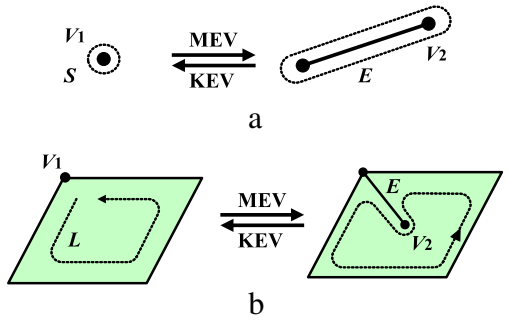


Fig. 5. MEV and KEV operators.

MEC (Shell *S, $Vertex$ *V_1, $Vertex$ *V_2, Edge **E, Curve *Cv)
KEC (Edge *E)

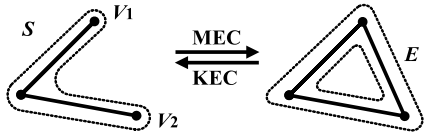


Fig. 6. MEC and KEC operators.

MFKC (Shell *S, EdgeList *E_list, Face **F, Surface *Sf)
KFMC (Face *F)

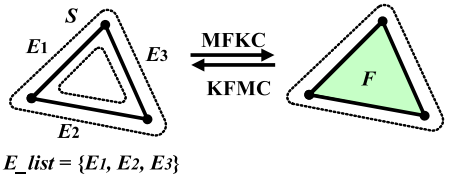


Fig. 7. MFKC and KFMC operators.

illustrated in Fig. 5. The geometry of the edge E is given as an input argument Cv. If the new edge is a wire edge as shown in Fig. 5(a), the shell S containing the edge E is set as the input argument $Parent$. On the contrary, if the edge E is a strut edge as shown in Fig. 5(a), the loop L including the edge E is set as $Parent$. The inverse of MEV is KEV, which destroys the specific edge and vertex. Here, the edge to be deleted should be a strut edge.

MEC, KEC

The operator MEC creates a new wire edge, E, joining two given vertices, vertex V_1 and V_2, to form a non-manifold cycle, as illustrated in Fig. 6. The newly generated edge, E, becomes a part of the specific shell, S. The geometry of the new edge is given as an input argument Cv. The inverse operator of MEC is KEC, which destroys the specific wire edge, E, together with a non-manifold cycle. The edge to be deleted should not be the only pass linking its two end vertices. For example, there are two passes between V_1 and V_2: the one is the edge E, and the other is two remaining edges. If there is only one pass and it is the edge to be deleted, then a new shell is created. The effect of MEC and KEC is depicted in Fig. 6.

MFKC, KFMC

The operator MFKC creates a new face, F, bounded by the specific edges in the list, E_list, as illustrated in Fig. 7. As a result, the non-manifold cycle composed of the edges disappears. The input edges as well as the output face belong to the specific shell, S. The geometry of the new face is given as an input parameter Sf. The inverse operator of MFKC is KFMC, which destroys the specific face, F, and creates a new non-manifold cycle composed of the boundary edges of F. The effect of MEC and KEC is described in Fig. 7. (OK)

MFR, KFR

The operator MFR creates a new face, F, and a new region, R, from the specific edges in the E_list. The region of the specific

MFR (Shell *S, $EdgeList$ *E_list, Face **F, Region **R, Surface *Sf)
KFR (Face *F, $Region$ *R)

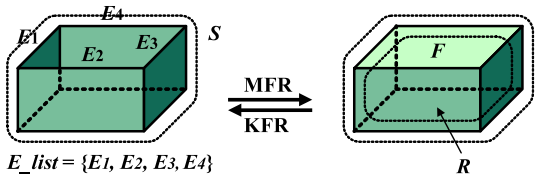


Fig. 8. MFR and KFR operators.

MVL (Face *F, Vertex **V, Loop **L, Point *Pt)
KVL (Loop *L)

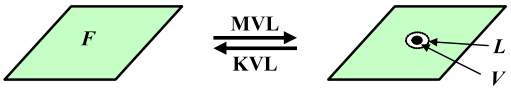


Fig. 9. MVL and KVL operators.

SEMV (Edge *E_1, Edge **E_2, Vertex **V, Point * Pt)
JEKV (Edge *E_2, Vertex *V)

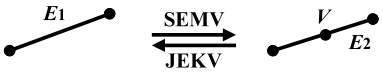


Fig. 10. SEMV and JEKV operators.

shell S is separated into two by adding a new face. Some void shells of the original region may be transferred to the new region according to the in/out test. The inverse operator of MFR is KFR, which merges two adjacent regions by destroying the face between them. The void shells of the deleted region, R, are attached to the other surviving region. The effect of MFR and KFR is depicted in Fig. 8.

MVL, KVL

The operator MVL creates a new vertex and a new loop at the same time. As shown in Fig. 9, the loop L is composed of an isolated vertex V, and belongs to the face F. The location of the new vertex is given with an input argument Pt. The inverse of MVL is KVL, which destroys the specific single-vertex loop.

3.3. Auxiliary Euler operators

SEMV, JEKV

The operator SEMV creates a new vertex, V, and a new edge, E_2, by splitting a given edge, E_1, at a given position, Pt, as shown in Fig. 10. Its inverse operator is JEKV, which merges two edges into one. The specific vertex and edge are deleted after JEKV is executed. The edges may be either manifold or non-manifold edges.

MEF, KEF

The operator MEF creates a new edge, E, and a new face, F, by subdividing a given loop, L, by joining two vertices, V_1 and V_2, as illustrated in Fig. 11. The operator attaches the inner loops of the face to either a new face or the other face according to the in/out test result. The inverse operator of MEF is KEF, which merges two adjacent faces by destroying the edge between them, E, and the specific face, F. The inner loops of the deleted face are automatically attached to the other surviving face.

KEML, MEKL

The operator KEML splits a given loop, L, into two new ones, L_1 and L_2, by removing an isthmus or strut edge, E, as illustrated in Fig. 12. This operator is exactly the same as the corresponding Euler operator for solid models. If the edge E is a strut edge, a single-vertex loop appears as a result. The inverse operator of KEML is MEKL, which merges two loops, L_1 and L_2, by joining two vertices, V_1 and V_2, which belong to two different loops of the same face.

MEF (Loop *L, Vertex *V_1, Vertex *V_2, Edge **E, Face **F, Curve *Cv)
KEF (Edge *E, Face *F)

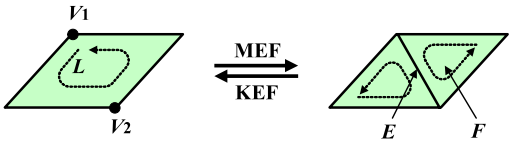


Fig. 11. MEF and KEF operators.

KEML (Edge *E, Loop ** L_1)
MEKL (Loop *L_1, Loop *L_2, Vertex *V_1, Vertex *V_2, Edge **E, Curve *Cv)

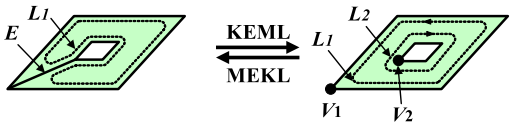


Fig. 12. KEML and MEKL operators.

KEMS, MEKS

The operator KEMS splits a given shell, S_1, into two ones, S_1 and S_2, by removing a wire edge, E, as illustrated in Fig. 13. In this case, the edge E should be the only entity linking its two vertices, V_1 and V_2. Otherwise, the shell S is not separated. If the edge is E, a strut wire edge, a single-vertex shell appears as a result. The inverse operator of KEMS is MEKS, which merges two shells, S_1 and S_2, by joining two vertices, V_1 and V_2, which belong to two different shells of the same region.

4. Non-manifold offset algorithm

This section presents an algorithm for non-manifold offsetting operations, which is based on the mathematical definitions and properties of non-manifold offsets described in Section 2. The algorithm focuses on the positive offset of a given non-manifold model X, because the negative offset is easily obtained by a sequential process of Eq. (6). According to Eq. (6), when a negative offset is desired, the system gets the complement of X by replacing the region attribute *void* with *solid* and vice versa before applying the positive offset algorithm to X^C, and after the positive offsetting, the complement operation is applied to $(X^C \oplus r)$ again. The positive offset algorithm for non-manifold models can be summarized as follows:

(Step 1) Select all the vertices, edges and faces whose offsets will contribute to the construction of a superset of the boundary of the offset model of X.
(Step 2) Generate the offset elements for the vertices, edges and faces selected in Step 1. The offset elements can be a sheet or a solid model.
(Step 3) Unite all the offset elements generated in Step 2 in order to create a superset model X_s whose boundary is $\partial(\partial X \oplus r)$.
(Step 4) Remove all topological entities of X_s that are within the offset distance r from the boundary of the original model X to obtain the exact offset model X_o.

In Step 2, the offset element of a vertex by a distance r is a spherical surface of radius r centered at the vertex position. The offset element of an edge is the envelope of the volume obtained by sweeping a ball along the edge curve as a trajectory. The offset element of a face is the rolling ball surface of radius r when its center moves through all the points on the face. However, it is not necessary to generate the full boundary of the offset element for all the vertices, edges, and faces. For instance, spheres need to be generated only for the vertices of singular points on the boundary of X. In addition, spheres do not need to be whole because, in some cases, even partial segments of the spheres are sufficient for the vertex offset elements. The same can be said of the edge and face offset elements. In order to reduce the computation time and numerical errors, in this paper, we modify the algorithm above: in Step 2, the minimal areas of the offset elements are generated according to the classification of the cases, and in Step 3, the intersection between the cell entities of the offset elements is calculated considering the origin of each topological entity in the offset element. The following sub-sections describe the steps above in more detail. Fig. 14 shows an example non-manifold model for a chair composed of a solid, a sheet, and four wire edges. This model will be used throughout this paper to explore our proposed offset algorithm.

4.1. Selection of topological entities to be offset

The system searches for all the shells of the void regions of X and then selects all the faces, edges and vertices that are adjacent to the shells. The offsets of these entities will participate in the boundary of the offset model of X. In the case of a model shown in Fig. 15, since the shells S_0, S_1 and S_3 are the boundaries of the void regions R_0 and R_2, the collected entities are all the faces, edges and vertices adjacent to S_0, S_1 and S_3. F_1 is excluded because it is adjacent to only S_2, which is a shell of the filled region R_1. In Partial Entity Structure, shells are oriented to the inner space of the regions. Each of the selected entities is offset towards the inner normal directions of its adjacent void shells. In the case of an example model shown in Fig. 14, all the faces, edges, and vertices are selected as they are adjacent to the only void region R_0 of the model.

4.2. Generation of offset elements for the selected entities

4.2.1. Vertex offset elements

Offset elements are generated for the vertices selected in Section 4.1. The positive offset element of a vertex by a distance r is defined as a sphere of radius r centered at the vertex position. However, it is not necessary to generate spheres for all the vertices. Spheres need to be generated only for the vertices of singular points on the boundary of X. In addition, spheres do not need to be whole because, in some cases, only partial segments of the spheres are sufficient for vertex offset elements. In this paper, the vertices are classified into several types as shown in Table 2. In cases 1.1 and 3.1, a whole sphere is generated as a vertex offset element using primitive generation capabilities in the modeler. However, in

KEMS (Edge *E, Shell ** S_1)
MEKS (Shell *S_1, Shell *S_2, Vertex *V_1, Vertex *V_2, Edge **E, Curve *Cv)

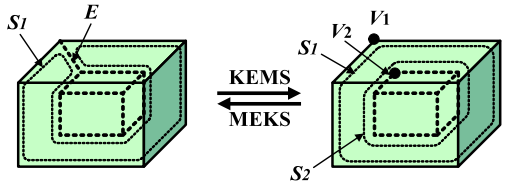
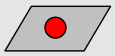
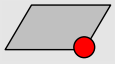
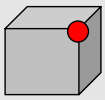


Fig. 13. KEMS and MEKS operators.

Table 2
Positive vertex offset elements.

Classification	Example	Case no.	Condition	Positive vertex offset element	Example of positive vertex offset element
1. Single-vertex shell		1.1	Always	Sphere	
2. Wire-edge vertex		2.1	If two edges join smoothly (g^1 cont.)	X	X
		2.2	For an end vertex	Sphere (hemi)	
		2.3	Otherwise	Sphere (partial)	
3. Single-vertex loop		3.1	If a singular point	Sphere	
		3.2	Otherwise	X	X
4. Sharp-edge vertex		4.1	If two edges join smoothly	X	X
		4.2	Otherwise	Sphere (partial)	
5. Inner vertex		5.1	If smooth	X	X
		5.2	If concave	X	X
		5.3	Otherwise	Sphere (partial)	

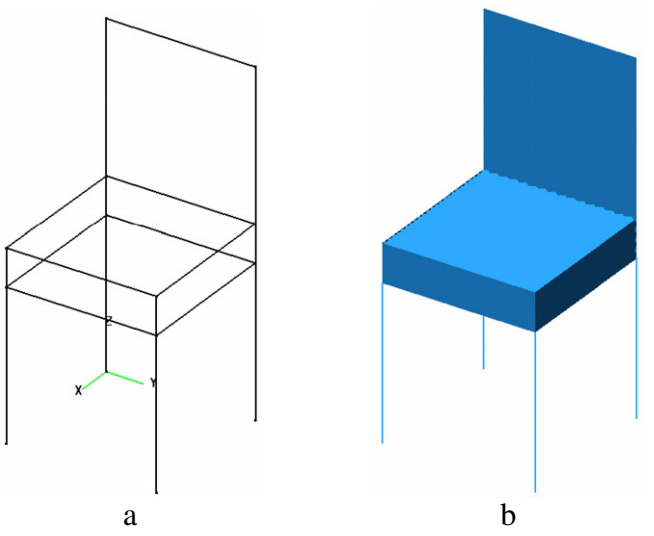


Fig. 14. An example non-manifold model for a simplified chair: (a) displayed in the wireframe mode; (b) displayed in the shaded mode.

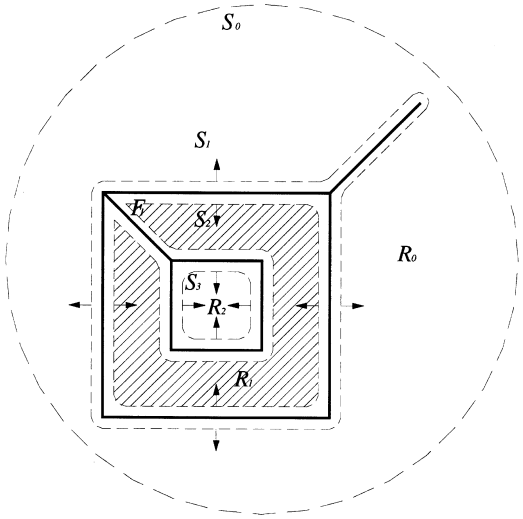


Fig. 15. A simple non-manifold model.

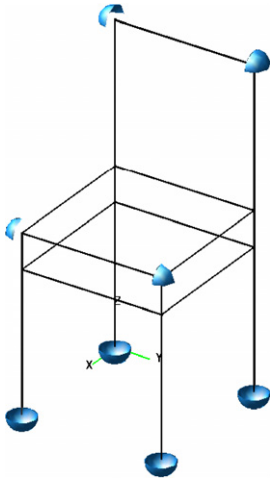


Fig. 16. Offset elements of the selected vertices for the example model shown in Fig. 14.

other cases, the vertex offset elements are only spherical segments or do not need to be generated. Fig. 16 shows the offset elements of the selected vertices for the example model shown in Fig. 14. The detailed implementation for Case 2.3 is illustrated below, although the detailed modeling procedure for each case is omitted here because it is a detail of implementation.

Cases 2.3 and 4.2: Singular vertices adjacent to only two edges

If a vertex is adjacent to only two edges of a wireframe or a sheet and does not satisfy G^1 continuity as shown in Fig. 17, a spherical segment, not a whole sphere, is enough for the positive vertex offset element. In this case, a sheet model of the spherical segment is generated by creating a semicircle as the profile and performing a rotational sweep about the axis $\mathbf{n}_1$ at an angle θ. Here, θ denotes the angle between two edge tangents $\mathbf{t}_1$ and $\mathbf{t}_2$ at the vertex position, and $\mathbf{n}_1$ is a unit normal obtained by $\mathbf{n}_1 = \mathbf{t}_1 \times \mathbf{t}_2$. The curve equation of the semicircle C_1 for E_1 is as follows:

$$C_1(t) = \mathbf{p}_v + r(\cos(t)\mathbf{n}_1 + \sin(t)\mathbf{b}_1), \quad 0 \le t \le \theta \tag{14}$$

where $\mathbf{p}_v$ is the vertex position and $\mathbf{b}_1$ is a unit vector obtained by $\mathbf{b}_1 = -\mathbf{t}_2$.

The algorithm for generating the vertex offset element for a singular vertex adjacent to only two edges can be described as follows:

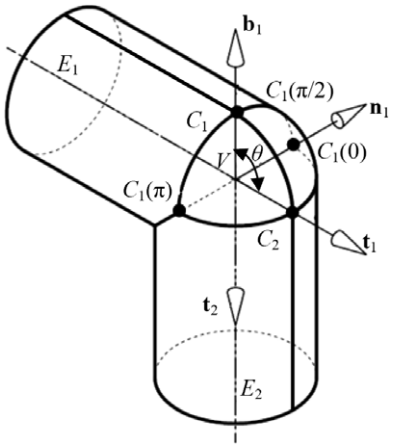


Fig. 17. Creating a spherical patch for a vertex offset element.

1. Calculate the start, end and bisectional points of C_1, i.e., $C_1(0)$, $C_1(\pi/2)$ and $C_1(\pi)$, using Eq. (14).
2. Generate a wireframe model for the semicircle C_1. In many cases, a semicircle is constructed by two edges of quadrant for easy calculation of geometric properties. A sequence of Euler operators is applied as follows:
   MMR $\rightarrow$ MVS at $C_1(0)$ $\rightarrow$ MEV from $C_1(0)$ to $C_1(\pi/2)$ $\rightarrow$ MEV from $C_1(\pi/2)$ to $C_1(\pi)$.
3. Generate a sheet model through the rotational sweeping of the semicircle at an angle θ.

4.2.2. Edge offset elements

The boundary of the positive edge offset element is a set of points that are at a distance r from the edge. It also can be viewed as a rolling ball surface of radius r when its center moves along the edge curve as a trajectory. As shown in Fig. 18, the rolling ball surface can be divided into three segments: two spherical segments S_1 and S_2 centered at the two ending points, and a tubular surface S_3 along the edge curve. Since two spherical segments were already generated in the vertex offset procedure, only a tubular segment is generated with a sheet model for the edge offset element.

The tubular surface can be viewed as a surface swept by a circle of radius r as its center moves along the edge curve. As illustrated in Fig. 19, if the edge curve is denoted by $E(u)$, the tubular surface S is as follows:

$$S(u, v) = E(u) + r(\cos(v)\mathbf{n}(u) + \sin(v)\mathbf{b}(u)) \quad (15)$$

where $\mathbf{t}(u)$ is a unit tangent of $E(u)$, $\mathbf{n}(u)$ is a unit normal, and $\mathbf{b}(u)$ is a unit binormal that is calculated by $\mathbf{b}(u) = \mathbf{t}(u) \times \mathbf{n}(u)$.

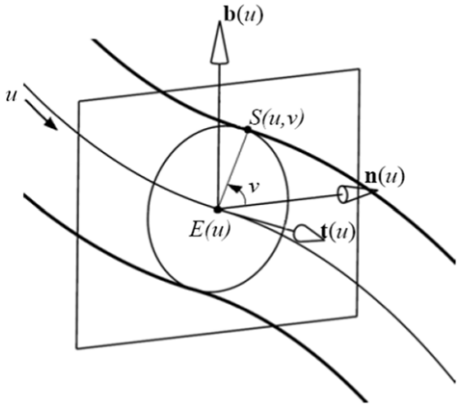


Fig. 19. Creating a tubular surface for an edge offset.

A tubular surface can be self-intersecting. However, the surface does not self-intersect, if it has the following property:

$$r < \min |1/\kappa(u)|, \quad (16)$$

where $\kappa(u)$ means the curvature of $E(u)$. Manipulating self-intersecting surfaces is very difficult and is one of the main research issues in the offset geometry area. However, since this paper is focused on the topology construction of offset models, it is assumed that all of the offset surfaces satisfy the property of Eq. (16), and it remains as future work to extend this algorithm to encompass self-intersecting offset surfaces.

Note that not all the edge offset elements may appear in the resulting offset model and, even if they do appear, only partial segments may do so. If this fact is considered in developing the algorithm, offsetting operations can work more efficiently. The classification of the edges and their positive offset results are illustrated in Table 3.

The positive offset element of a wire edge is the sheet model of a full tubular surface, while the offset element of a sharp edge or an inner edge is the sheet model of a partial tubular surface or nothing. Note that an inner edge can be adjacent to more than two faces in non-manifold models. In this case, the convexity of the edge cannot be defined. Therefore, in this paper, a new type of object, called 'wedge', is introduced. A wedge represents a corner space bounded by an edge and its two adjacent faces. Actually, a wedge class in the system contains three pointers to an edge and its adjacent two partial faces. Each of the wedges can be classified into convex, concave and smooth. If an edge has a convex wedge, a tubular sheet is generated for that wedge in the edge offset procedure.

Edge offset algorithm

1. Generate a new model using MMR.

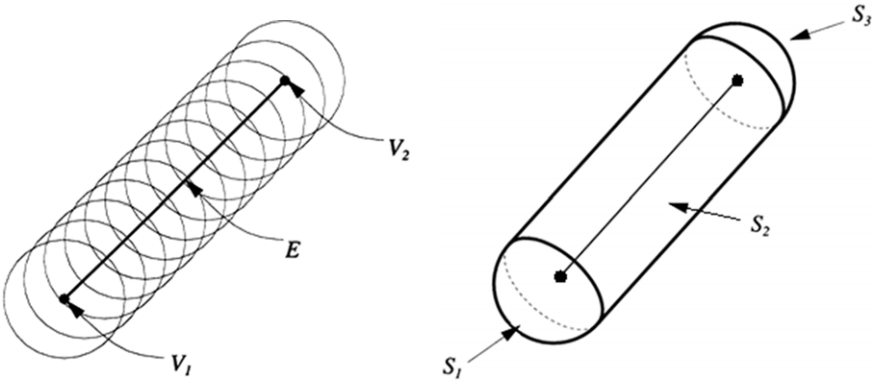
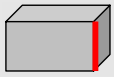


Fig. 18. Offsetting an edge by rolling a ball along the edge.

Table 3
Positive edge offset elements.

Classification	Example	Condition	Positive edge offset element	Example of positive edge offset element
1. Wire edge		Always	Tubular surface	
2. Sharp edge		Always	Half tubular surface	
3. Inner edge		For convex edge	Partial tubular surface	
		For concave edge	X	
		For smooth edge (G^1 continuity)	X	

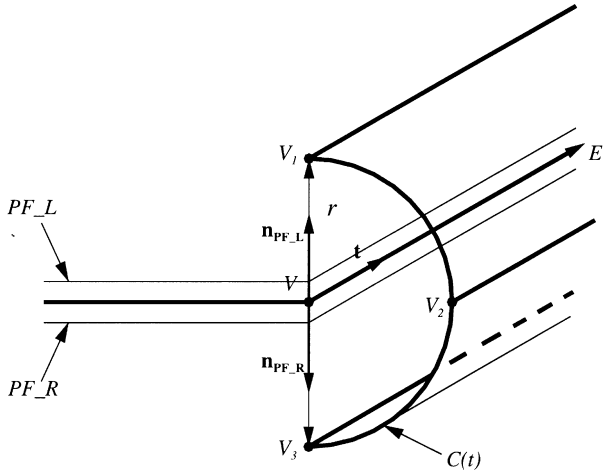


Fig. 20. An edge offset element for a sharp edge.

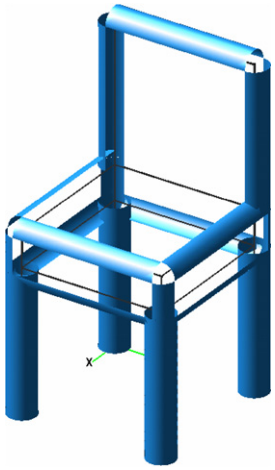


Fig. 21. Offset elements of the selected edges for the example model shown in Fig. 14.

2. Create a wireframe model for the profile of a circular arc. The arc can be defined as follows:

$$C(t) = E(0) + r(\cos(t)\mathbf{n} + \sin(t)\mathbf{b}), \quad 0 \le t \le \theta \tag{17}$$

where $E(0)$ is the start position of the edge, $\mathbf{n}$ is the unit normal at $E(0)$, and $\mathbf{b}$ is the binormal calculated by $\mathbf{b} = \mathbf{t} \times \mathbf{n}$ if $\mathbf{t}$ is the unit tangent at $E(0)$. For the case 1, θ is 2π. For the case 2, as shown in Fig. 20, $\mathbf{n}$ is the unit normal of the left partial face in the wedge $\mathbf{n}_{PF\_L}$, and θ is π. For the case 3, $\mathbf{n}$ is $\mathbf{n}_{PF\_L}$, and θ is the angle from $\mathbf{n}_{PF\_L}$ to $\mathbf{n}_{PF\_R}$. When creating a circular wireframe model, an edge is generated for each $\pi/2$ for easy manipulation of topological data in our system.
3. Create a tubular sheet model by sweeping the profile wireframe along the edge curve.

Fig. 21 shows the offset elements of the selected edges for the example model shown in Fig. 14.

4.2.3. Face offset elements

The boundary of the positive face offset element is a set of points that are at a distance r from the face. It can also be viewed as a rolling ball surface of radius r when its center moves through all the points on the face. In this paper, normal offsetting is defined as offsetting a face by a given distance r along the face normal. As illustrated in Fig. 22, the rolling ball surface can be divided into three types of segments: spherical surfaces originated from vertices (for example, F_{V1}), tubular surfaces from edges (for example, F_{E1}) and two normal offset surfaces from the face (for example, F_{F1}). Since the spherical and tubular surfaces were already generated as the vertex and edge offset elements in Sections 4.2.1 and 4.2.2, only the normal face offset surfaces need to be generated for the face offset elements.

According to the number of *void* regions adjacent to a face, the faces are classified into two groups, i.e., laminar faces and normal faces, and different numbers of the normal offsets are generated, as illustrated in Table 4. Laminar faces are adjacent to two *void* regions, and two offset sheet models are generated in both normal directions of the face. Normal faces are adjacent to one *void* and one *filled* regions, and only one offset sheet model is generated in the normal direction to the void region.

If $\mathbf{n}$ denotes the face normal to the void region and F denotes the face, the normal face offset F_o is as follows:

$$F_o = \{\mathbf{p}_o | \exists \mathbf{p} \in F, \mathbf{p}_o = \mathbf{p} + r\mathbf{n}\}. \tag{18}$$

If the face surface S is defined by a function of two parameters, u and v, its normal offset surface S_o is as follows:

$$S_o(u, v) = S(u, v) + r\mathbf{n}(u, v), \tag{19}$$

where

$$\mathbf{n}(u, v) = (S_u \times S_v)/\|S_u \times S_v\|. \tag{20}$$

If $C(t)$ denotes the edge curve that constitutes the face boundary, its normal offset curve $C_o(t)$ is as follows:

$$C_o(t) = C(t) + r\mathbf{n}(t), \tag{21}$$

where

$$C(t) = S(u(t), v(t)) \tag{22}$$

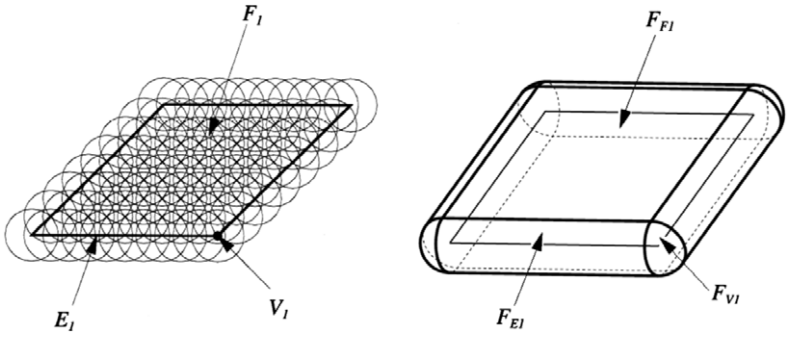
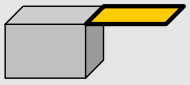
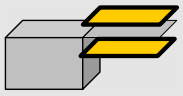
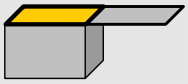
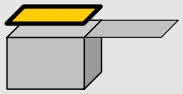


Fig. 22. A positive face offset.

Table 4
Positive face offset elements.

Classification	Example	Condition	Positive face offset result	Example of positive face offset element
1. Laminar face		Always	Two offset faces	
2. Normal face		Always	One offset face	

and

$$\mathbf{n}(t) = (S_u(u(t), v(t)) \times S_v(u(t), v(t)))/ \\ \|S_u(u(t), v(t)) \times S_v(u(t), v(t))\|. \quad (23)$$

If $\mathbf{p}_V$ denotes the position of a vertex on the face boundary, its normal offset $\mathbf{p}_{Vo}$ is as follows:

$$\mathbf{p}_{Vo} = S_o(u, v) = \mathbf{p}_V + r\mathbf{n}(u, v), \quad (24)$$

where

$$\mathbf{p}_V = S(u, v). \quad (25)$$

Note that a face offset element can be degenerated to a vertex if the face has a spherical surface whose radius is the same as the offset distance. In addition, a face offset element can be degenerated to a wire edge if the face has a tubular surface whose radius of curvature is the same as the offset distance. Moreover, a face offset element can self-intersect [34]. In this case, new edges and vertices should be generated. In this paper, however, self-intersection is not considered in this algorithm as mentioned above and remains as future work.

Face offset algorithm

1. Generate a new model using MMR.
2. Check whether or not the face has a spherical surface whose radius is the same as the offset distance. If so, generate a vertex at the center position of the spherical surface using MVS, and then quit this procedure.
3. Check whether or not the face has a cylindrical surface whose radius is the same as the offset distance. If so, generate a wire edge using MVS and MEV, and then quit this procedure.
4. Otherwise, a sheet model for the face offset element is generated as follows:
   - 4.1 Collect the edges and vertices that constitute the face boundary. Then, generate the offset curves for the edges with Eq. (21) and the offset points for the vertices with Eq. (24).
   - 4.2 Select a vertex on the peripheral loop L_1 of the face. Then, generate a vertex using MVS at the offset position of the selected vertex.
   - 4.3 If the loop L_1 has n edges, generate a wire loop of n edges with their offset curves by calling $(n-1)$ MEVs and one MEC.
   - 4.4 Generate a face to close the wire loop using MFKC. The offset surface is attached to this face.
   - 4.5 If the original face has hole loops, carry out the following procedure for each hole loop; otherwise, quit this algorithm.
     - 4.5.1 Generate a bridge edge to connect a vertex on the peripheral loop L_1 with a vertex on the hole loop L_2. Note that the bridge edge should be made not to intersect with any other edges of the hole loop L_2.
     - 4.5.2 Generate edges for the hole loop with the offset curves using MEVs and MEFs.
     - 4.5.3 If any new face created in Step 4.5.2 is an actual hole in the original face, remove it using KFMC to generate a hole.
     - 4.5.4 Remove the bridge edge using KEML.

Fig. 23 shows the offset elements of the selected faces for the example model shown in Fig. 14.

4.3. Union of the vertex, edge and face offset elements

Once the offset elements for the vertices, edges and faces are generated, they are united into a single non-manifold offset model using the non-manifold Boolean operations. Most algorithms for non-manifold Boolean operations are based on the concept of *Merging and Selection* [26,35,36]. The merging and selection algorithm consists of two steps. In the first merging step, a non-manifold model, called a merged-set, is generated by uniting all primitives in the CSG tree of the Boolean operations. Then, in the next selection step, the topological entities that constitute the resulting body of the CSG tree are selected for their display or application. However, for our sheet thickening operations, the selection step is not necessary, as the union result is always what we want. The union operation of two non-manifold models, *A* and *B*, is composed of the following three steps.

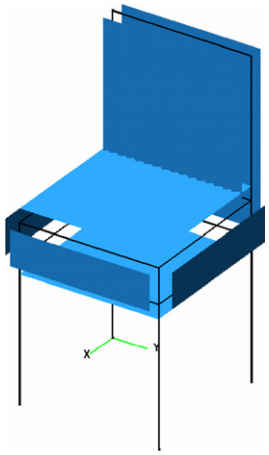


Fig. 23. Offset elements of the selected faces for the example model shown in Fig. 14.

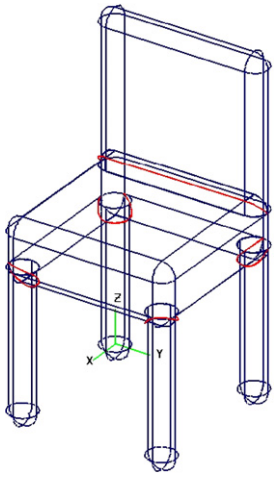


Fig. 25. Uniting all the offset elements of the selected topological entities for the example model shown in Fig. 14.

1. Calculate the intersection points, curves, and surfaces between two models, *A* and *B*. This step is well known to be the most time consuming and error prone. To reduce the computation time and numerical errors, in our algorithm, the topological information of the original model is used for the union of the offset elements made in Section 4.2. Each topological entity of an offset element has a pointer to its original entity in the sheet model. Before the calculation of the intersections of two entities, their original entities are checked to see if they are identical. If so, the overlapping curves can be obtained easily by some additional checking of the geometric properties of the two entities. Otherwise, the routine process for calculating intersections is used. Fig. 24 shows the relationship between the original entities and their offset elements. Each of the faces, edges, and vertices of an offset element stores its parent original entity in its attributes. In Fig. 24(a), for example, all of F_1, E_4, E_5, E_6, V_2, V_3, V_4 are originated from V_1. In Fig. 24(b), E_2 and E_4 are originated from E_1; E_3 and E_5 are from V_1 and V_2, respectively; V_3 and V_6 are from V_1; and V_4 and V_5 are from V_2. In Fig. 24(c), F_2 is originated from F_1; E_5, E_6, E_7, E_8 are from E_1, E_2, E_3, E_4, respectively; and V_5, V_6, V_7, V_8 are from V_1, V_2, V_3, V_4, respectively.
2. Generate new topological entities on the models *A* and *B* using the intersection points and curves. First, vertices are created at the intersection points and the end points of the intersection curves. Next, edges are created with the intersection curves. According to the intersection condition, one of the Euler operators is selected among MVL, MEV, SEMV, MEKL, and MEF. When new topological entities are created, the system also stores not only the partnership data, namely the pairs of topological entities of *A* and *B* that have the same geometry, but also the history records of the topological entities, from which all the parent entities can be traced.
3. Copy all of the topological entities of *B* to *A*. The system finds *B*'s topological entities that do not have any partner in *A*, and then copies them into *A* using Euler operators. The copy operations are carried out for the entities of low dimension to higher ones, i.e., in the order of the vertex, edge, and face. This method reduces the total number of Euler operators to be applied, as the boundary entities of higher-dimensional entities are created in advance. The partnership data are referred to in order to find the bounding vertices of an edge to be created, and the bounding edges of a face to be created.

Fig. 25 shows the union result of all the offset elements of the selected vertices, edges, and faces of the example model shown in Figs. 16, 21 and 23, respectively.

4.4. Removal of unnecessary topological entities

The united offset model can have topological entities that are within the offset distance r from the original model X. They should be removed to complete the boundary of the non-manifold offset model. If the united offset model is denoted by X_o, the removal algorithm is as follows:

1. Search for the faces, edges and vertices of X_o to be removed. The system calculates the closest distance between each entity of X_o and X. If the distance is less than r, the entity of X_o is marked to be deleted.

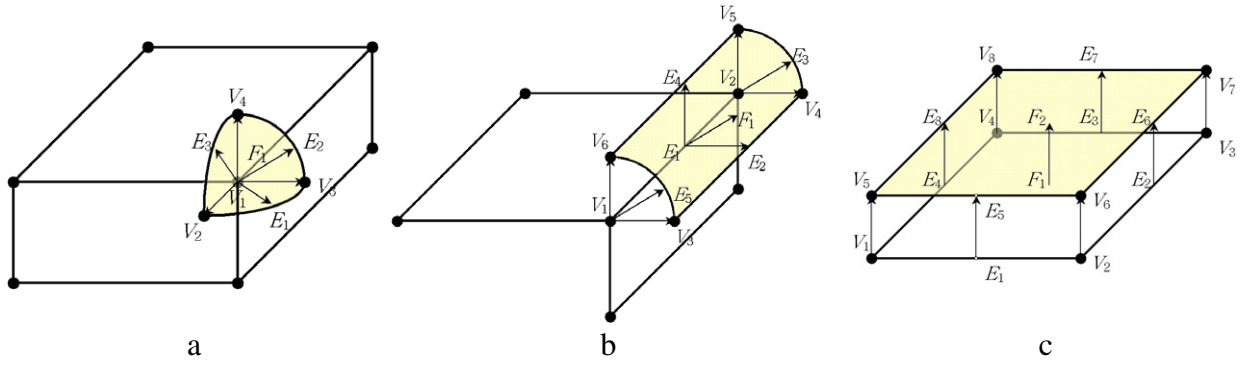


Fig. 24. Topological entities of the offset element and their original entities: (a) a vertex and its child entities; (b) an edge and its child entities; (c) a partial face or face-use and its child entities.

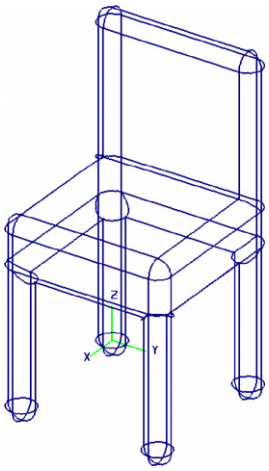


Fig. 26. Removing all the unnecessary topological entities from the model shown in Fig. 25.

2. Delete the marked faces using KFMC or KFR. If the face is adjacent to the same region on both sides of the face, KFMC is called. Otherwise, KFR is called to delete not only the face but also one of its adjacent regions to merge two regions into one.
3. Delete the marked edges using KEC, KEV or KEMS. KEC is used when the edge is a part of a wire loop. KEV is used when the edge is a strut edge with a free end. KEMS is used when a new isolated vertex is generated by deleting the edge. Note that some marked vertices can be deleted when KEVs are called.
4. Delete the marked vertices, which survived Step 3, using KVSs.

Fig. 26 shows the result of removing all the unnecessary topological entities from the united model shown in Fig. 25.

The negative offset of a given non-manifold model X is easily obtained by a sequential process of Eq. (6): $X \ominus r = (X^C \oplus r)^C$. Fig. 27 shows the negative offset process for the example model shown in Fig. 14. Fig. 27(a) shows the complement of the model and the selected topological entities to be offset. Fig. 27(b) shows the offset elements for the selected topological entities. Fig. 27(c) shows the united model of all the offset elements. Fig. 27(d) shows the final negative offset result obtained by removing the unnecessary entities and executing the complementing operation.

4.5. Case study

Fig. 28 shows a non-manifold model for a snow sled created with a sheet and wire edges, and its positive offset result. Fig. 29 shows a solid model and its negative offset result. Fig. 30 shows the rounding and filleting effects of non-manifold models by combining positive and negative offsetting operations. $(X \ominus r) \oplus r$ rounds the convex edges and vertices of the given object as shown in Fig. 30(b), while $(X \oplus r) \ominus r$ fillets the concave edges and vertices as shown in Fig. 30(c).

5. Variations of non-manifold offset algorithm

The offset algorithm based on the mathematical definition needs to be modified to obtain more practical offset results for various application areas. In this section, some variant offset algorithms for a wireframe are discussed.

5.1. Wireframe offsetting for pipelines

If a user wants to obtain solid models for pipelines, the user may create wireframe models and then convert them into solids using the offsetting operations proposed in this paper. However, the shapes of the resulting solids are far from those of actual products because they have spherical surfaces at the ends of pipelines, as shown in Fig. 31(a). Therefore, the offset algorithm needs to be adapted for more practical applications, as shown in Fig. 31(b). Since the spherical surfaces are originated from the offset elements of the vertices at the ends of the wireframe, those vertex offset elements should be replaced with sheet models for flat discs. On the other hand, if the user wants to obtain sheet models for pipelines, the end-vertex offset elements should not be generated.

5.2. Sheet thickening for thin-walled parts

The non-manifold offset algorithm described above can be applied for converting sheets to solids to facilitate solid modeling for thin plastic or sheet metal parts. However, if the non-manifold offsetting operation is applied to an L-shaped sheet model shown in Fig. 32(a), the offset solid of the sheet would have thickness faces with tubular surfaces that satisfy the mathematical definition of offsets, as shown in Fig. 32(b). However, tubular thickness faces

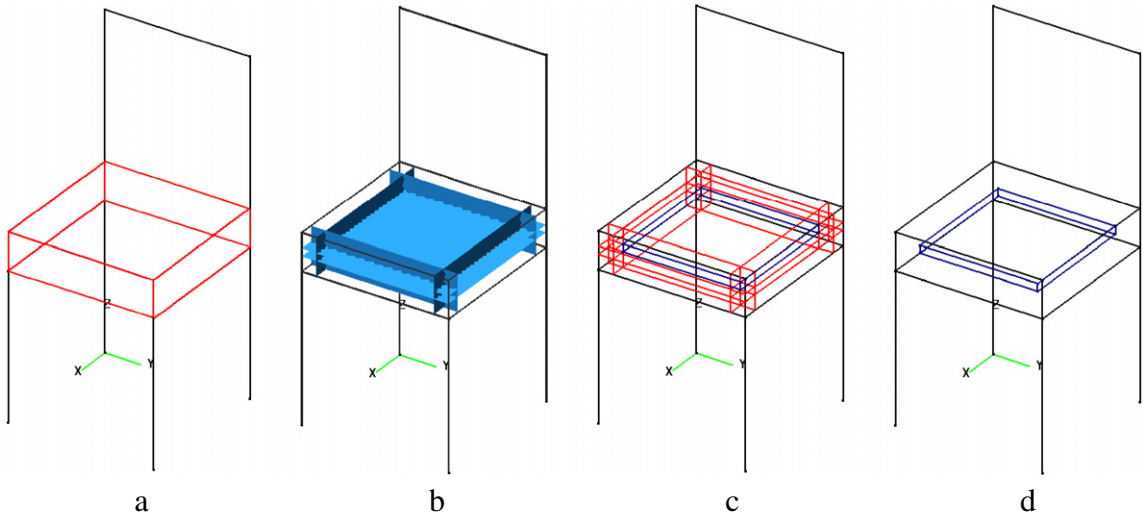


Fig. 27. Negative offset process: (a) executing the complementing operation and selecting the entities to be offset; (b) generating the offset elements for the selected entities; (c) uniting all the offset elements; (d) purging unnecessary entities and executing the complementing operation.

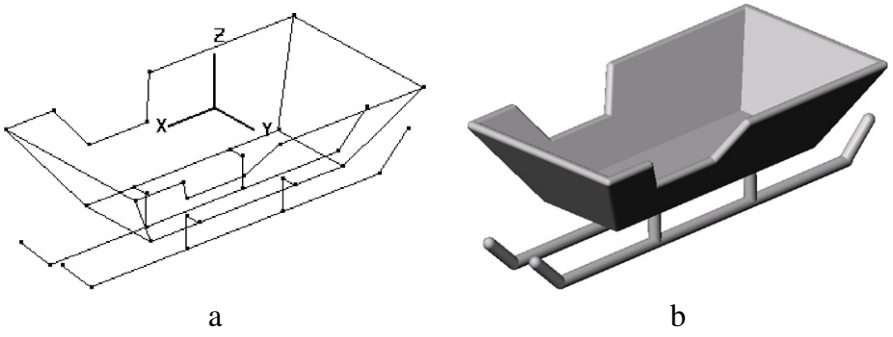


Fig. 28. Offsetting a non-manifold model in the positive direction: (a) a simple non-manifold model for a snow sled; (b) a positive offset model.

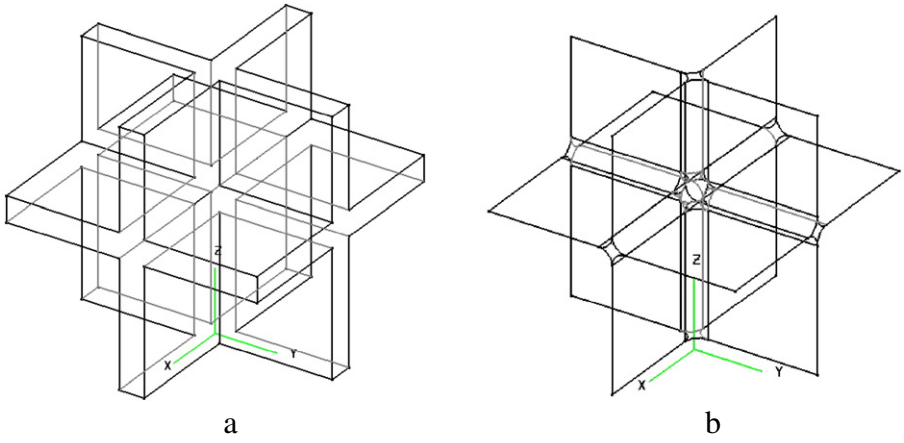


Fig. 29. Offsetting a solid model in the negative direction: (a) a simple solid object; (b) a negative offset model.

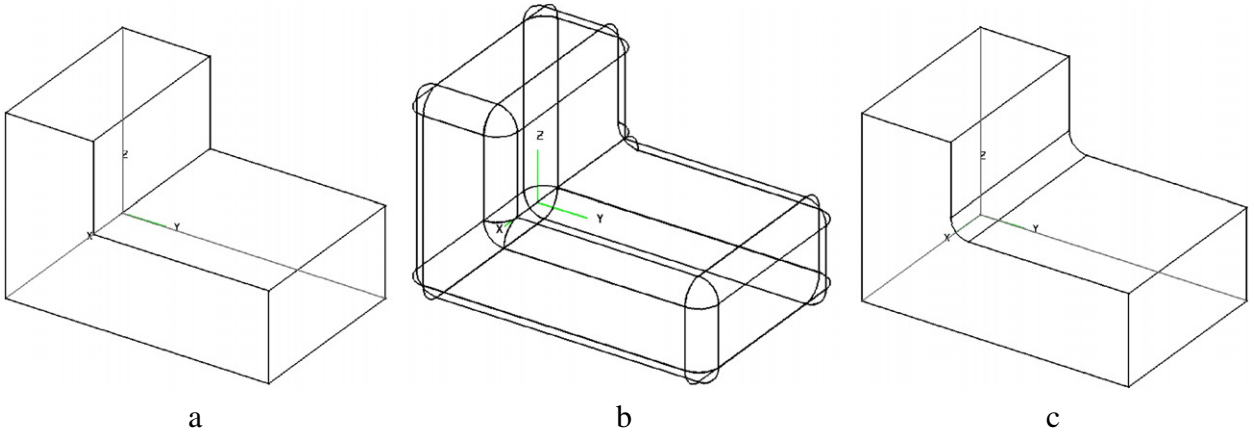


Fig. 30. Rounding and filleting results using the offset operations: (a) rounding the convex edges and vertices by $(\mathbf{X} \ominus \mathbf{r}) \oplus \mathbf{r}$; (b) filleting the concave edges and vertices by $(\mathbf{X} \oplus \mathbf{r}) \ominus \mathbf{r}$.

are rarely shown in actual plastic or sheet metal parts. Moreover, in many cases, a user prefers to offset a sheet model for the outer or inner shape of the part in one normal direction rather than to offset in both normal directions. Therefore, we need to modify the proposed offset algorithm to generate practical, thin-walled solids, as shown in Fig. 32(c).

The tubular surfaces are generated by sweeping an arc along the sharp edges at the edge offsetting step in Section 4.2.2. Therefore, if a line is adopted as the sweeping profile instead of an arc, they are converted to the ruled surfaces immediately. The end points $\mathbf{p}_{v1}$ and $\mathbf{p}_{v2}$ of the line are calculated as follows:

$$\mathbf{p}_{v1} = E(0) + r\mathbf{n}_{PF\_L} \tag{26}$$

$$\mathbf{p}_{v2} = E(0) + r\mathbf{n}_{PF\_R}. \tag{27}$$

The spherical surfaces originate from the offsets of the singular vertices of the sharp edges. One probable algorithm is not to generate vertex offsets for these vertices. Then, some holes may remain on the thickness faces when the offset models for the faces, edges and vertices are united in a single model. In that case, the system detects and closes these holes with new faces. If the boundary of a hole is on a plane, the plane is created and attached to the face. Otherwise, a freeform surface that interpolates the hole boundary is attached to the face. If there are n holes, MFKCs are called $n-1$ times and then one MFR is called finally to close n holes. If the user wants to clean up the resulting offset solid, he can use the tidy operations that merge two neighbor faces or edges with the same geometry and delete isolated vertices and edges.

When a user creates a thin-walled solid model, in many cases, he/she prefers to offset a sheet model for the outer or inner shape

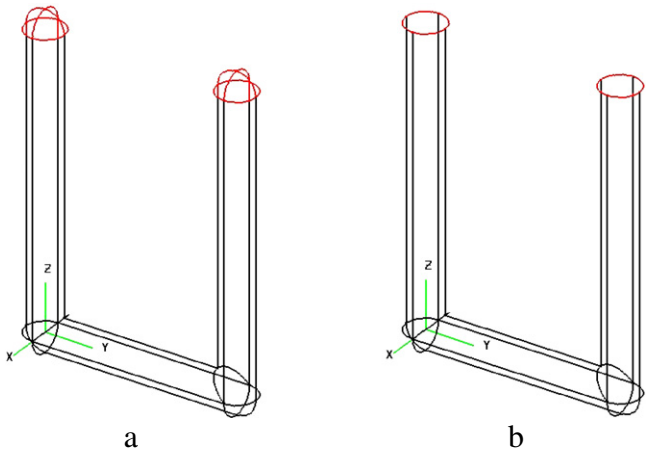


Fig. 31. Wireframe offsets for solid modeling of pipelines: (a) applying the algorithm based on the mathematical definition; (b) applying the adapted algorithm for more practical pipeline modeling.

of the part in one normal direction rather than to offset in both normal directions. To meet this requirement, one of the end points $\mathbf{p}_{v1}$ and $\mathbf{p}_{v2}$ in Eqs. (26) and (27) should be $E(0)$ depending on inner or outer offsetting. Fig. 33(a) shows a sheet model for the inside wall of the cover of a mouse device. By applying the sheet thickening operation in the outside direction, a solid model shown in Fig. 33(b) is obtained. In this case, although four holes occur on the thickness faces, they are detected and removed automatically in the hole-filling step.

This modified algorithm is more logical and easier than those of Stroud [16], Lee and Kwon [17], or Lee and Lim [1] that are based on solid data structures and modeling functions. However, the algorithm should be designed more precisely considering exceptional cases of thickness faces. This remains as future work.

5.3. Solid shelling for thin-walled parts

The non-manifold offset algorithm proposed in this paper also can be used for development of shelling operations on solid bodies. The shelling operation generates a thin-walled solid with constant wall thickness by removing the inside volume from the solid model of the outside shape [12]. Shelling operations are classified into two categories: closed shelling and open shelling. If the closed shelling operation is applied to a solid, it results in a thin-walled solid with an interior void. In open shelling operations, users can specify the faces to be opened. If the open shelling operation is applied to a solid, a thin-walled solid body with openings in the specified faces is generated. Open shelling operations are very useful to create solid models for bowl-like plastic parts.

The algorithms for closed shelling operations can be devised easily by applying the non-manifold offsetting operation and the difference Boolean operation successively. If the original solid is M and its offset is N, then the closed shelling operation on M results in the solid S formed by the difference [12]:

$$\text{If } d < 0, \quad S = M - N \tag{28}$$

$$\text{If } d > 0, \quad S = N - M \tag{29}$$

where d is the offset distance. If the offset distance d is negative, the offset body will form an interior void and the original body will remain as the exterior of the shelled body. If d is positive, the original body will form an interior void and the offset body will form the exterior of the shelled solid.

The algorithm for the open shelling operation can be implemented very easily using the algorithm for sheet thickening. First, a user specifies the faces to be opened and gives the offset distance d. Then, the system first removes the selected faces from the original body to create a sheet body. Thickness faces appear on the opened faces after the execution of the first step. Secondly, it thickens the sheet body in an inside direction by a given offset distance d, if d

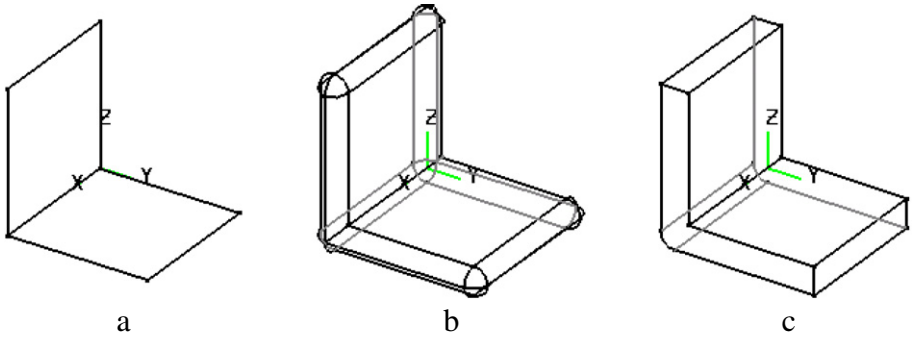


Fig. 32. Different types of offset models for a sheet: (a) a simple L-shaped sheet model; (b) a solid model constructed by the offset algorithm based on the mathematical definition of offsets; (c) an offset solid model as desired for practical part design.

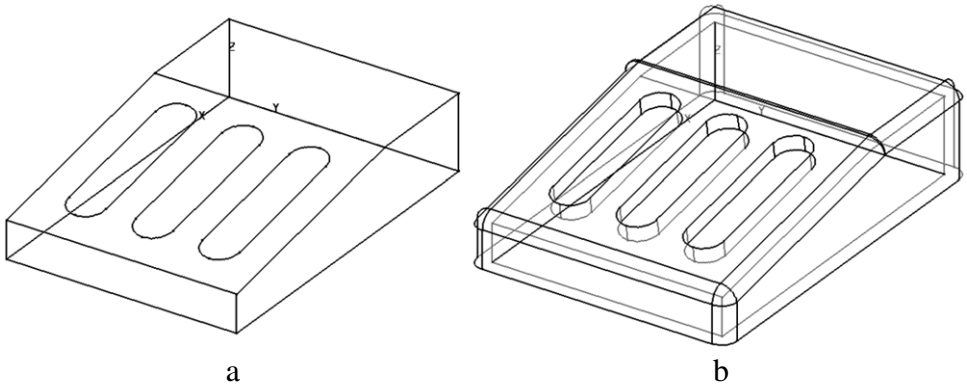


Fig. 33. Modeling of a mouse device: (a) a sheet model for the inner wall of a mouse device; (b) thin-wall solid model obtained by offsetting in the outside direction.

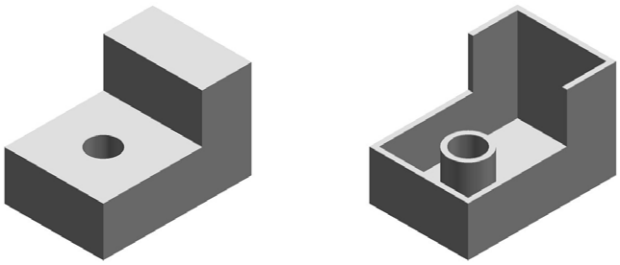


Fig. 34. Modeling of a plastic part: (a) a solid model for the outer wall; (b) a thin-walled solid model obtained by an open shelling operation.

is negative, and if d is positive, thicken the sheet body in an outside direction. Fig. 34(a) shows a solid model for the outside wall. By applying the open shelling operation in an inside direction, a thin-walled solid model as shown in Fig. 34(b) is obtained.

This approach can solve global intersection problem in easy way by taking advantage of the non-manifold offset algorithm. However, it has a limitation on the shape of the thickness faces. The ruled surfaces for the thickness faces can be different from the surface of the removed face. It may be undesirable in some cases. To overcome this shortage, it is necessary to extend the existing algorithm to manipulate this case upon the user's request. This remains as future work.

6. Conclusion

This paper introduced offsetting operations in non-manifold geometric modeling, which can be applied not only to wireframes, sheets and solids but also to their mixtures in one integrated modeling environment. The mathematical definitions and properties of non-manifold offsetting operations were discussed first, and then the offset algorithm using the non-manifold Euler operators were described in detail and implemented in a non-manifold geometric modeler called NGM, which was developed by the author based on the Partial Entity Structure [33]. In particular, in order to reduce the computation time and numerical errors, the minimal areas of the offset elements for the vertices, edges and faces are generated according to the classification of the cases, and the intersection between the offset elements is calculated considering the origins of the topological entities of the offset elements. Since non-manifold geometric modeling systems represent all of the wireframe, surface and solid models with a single unified representation scheme, offsetting operations for each type of model can be integrated into one in a single environment and have great potential to be utilized in many application areas including conceptual design. For instance, for efficient solid modeling of thin-walled plastic or sheet metal parts, sheet thickening or solid shelling operations can be easily implemented based on a non-manifold topological representation and the non-manifold offset algorithm.

However, there still remains future work to complete the implementation of the original and variant offset algorithms for non-manifold models. First, although the algorithm can be applied to a non-manifold model with any type of surface and curve, I implemented the non-manifold modeling system supporting only planar and quadratic surfaces. Therefore, more implementation is necessary to enlarge the surface types such as freeform surfaces. In addition, topological irregularity of an offset face caused by self-intersection has not been considered yet. Therefore, a more rigorous algorithm that can manage the self-intersection problem needs to be devised in the future [34].

Acknowledgement

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (KRF-2008-313-D00066).

References

[1] Lee K, Lim SH. Efficient solid modelling via sheet modelling. Computer-Aided Design 1995;27(4):255–62.
[2] Chen YJ, Ravani B. Offset surface generation and contouring in computer-aided design. Journal of Mechanisms, Transmissions and Automation in Design: ASME Transactions 1987;109(3):133–42.
[3] Lozano-Perez T, Wesley MA. An algorithm for planning collision free paths amongst polyhedral obstacles. Communications of the ACM 1979;25(9):560–70.
[4] Li CL, Yu KM, Lam TW. Implementation and evaluation of thin-shell rapid prototype. Computers in Industry 1998;35:185–93.
[5] Rossignac JR, Requicha AAG. Offsetting operations in solid modeling. Computer Aided Geometric Design 1986;3:129–48.
[6] Kim SH. Development of form feature based non-manifold CAD system supporting top-down assembly modeling. Ph.D. thesis. Korea: Seoul National University; 1994 [in Korean].
[7] Pham B. Offset curves and surfaces: A brief survey. Computer-Aided Design 1992;24(4):223–9.
[8] Takashi M. An overview of offset curves and surfaces. Computer-Aided Design 1999;31(3):165–73.
[9] Farouki RT. Exact offset procedures for simple solids. Computer Aided Geometric Design 1998;2:257–79.
[10] Saeed SEO, Pennington A, Dodsworth JR. Offsetting in geometric modeling. Computer-Aided Design 1998;20(2):50–62.
[11] Satoh T, Chiyokura H. Boolean operations on sets using surface data. In: Rossignac R, Turner J, editors. Proceedings of the 1st ACM symposium on solid modeling foundations and CAD/CAM applications. Austin (Texas, USA): ACM; 1991. p. 119–27.
[12] Forsyth M. Shelling and offsetting bodies. In: Hoffmann C, Rossignac R, editors. Proceedings of the 3rd ACM symposium on solid modeling and applications. Salt Lake City (Utah, USA): ACM; 1995. p. 373–81.
[13] Ravi Kumar GVV, Shastry KG, Prakash BG. Computing constant offsets of a NURBS B-rep. Computer-Aided Design 2003;35(10):935–44.
[14] Spacial Co. 3D ACIS Modeler R.19; 2008. doc.spatial.com.
[15] UGS. Parasolid V16.1 – Functional description. 2004.
[16] Stroud I. Modeling with degenerate objects. Computer-Aided Design 1990;22(6):344–51.
[17] Lee K, Kwon BW. Efficient modeling method of sheet objects. In: Proceedings of ASME computers in engineering conference. vol. 1. 1992. p. 437–46.
[18] Lee SH, Lee K. An integrated CAD system for mold design in injection molding process. In: Proceedings of the winter annual meeting of the ASME. PED-vol. 32. 1988. p. 257–71.
[19] Park SC. Hollowing objects with uniform wall thickness. Computer-Aided Design 2005;37:451–60.
[20] Chiu WK, Tan ST. Using dexels to make hollow models for rapid prototyping. Computer-Aided Design 1998;30(7):539–47.
[21] Lam TW, Yu KM, Cheung KM, Li CL. Octree reinforced thin-shell rapid prototyping. Materials Processing Technology 1997;63:784–7.
[22] Alexander P, Dutta D. Layered manufacturing of surfaces with open contours using localized wall thickening. Computer-Aided Design 2000;32:175–89.
[23] Ganesan M, Fadel GM. Hollowing rapid prototyping parts using offsetting techniques. In: Proceedings of the 5th international conference on rapid prototyping. 1994. p. 241–51.
[24] Koc B, Lee YS. Non-uniform offsetting and hollowing objects by using biarcs fitting for rapid prototyping processes. Computers in Industry 2002;47:1–23.
[25] Xiuzhi Q, Brent S. A 3D surface offset method for STL-format models. Rapid Prototyping Journal 2003;9(3):133–41.
[26] Masuda H. Topological operators and Boolean operations for complex-based non-manifold geometric models. Computer-Aided Design 1993;25(2):119–29.
[27] Mäntylä M. An introduction to solid modeling. New York: Computer Science Press; 1987.
[28] Yamaguchi Y, Kimura F. Non-manifold topology based on coupling entities. IEEE Computer Graphics and Applications 1995;15(1):42–50.
[29] Heisserman JA. A generalized Euler-Poincare equation. In: Rossignac R, Turner J, editors. Proceedings of the 1st ACM symposium on solid modeling foundations and CAD/CAM applications. Austin (Texas, USA): ACM; 1991. p. 533.
[30] Higashi M. High-quality solid-modelling system with free-form surfaces. Computer-Aided Design 1993;25(3):172–83.
[31] Luo Y.. Generalized Euler operators for non-manifold boundary solid modeling. In: Geometric modelling studies, 1990/3, MTA SZTAKI, Hungary. 1993. p. 19–34.
[32] Luo Y, Gabor L. A boundary representation for form features and non-manifold solid objects. In: Rossignac R, Turner J, editors. Proceedings of the 1st ACM symposium on solid modeling foundations and CAD/CAM applications. Austin (Texas, USA): ACM; 1991. p. 45–60.
[33] Lee SH, Lee K. Partial entity structure: A compact boundary representation for non-manifold geometric modeling. Journal of Computing and Information Science in Engineering: ASME Transactions 2001;1(4):356–65.
[34] Seong JK, Elber G, Kim M-S. Trimming local and global self-intersections in offset curves/surfaces using distance maps. Computer-Aided Design 2006;38(3):183–93.
[35] Crocker GA, Reinke WF. An editable non-manifold boundary representation. IEEE Computer Graphics & Applications 1991;11(2):39–51.
[36] Gursoz EL, Choi Y, Prinz FR. Boolean set operations on non-manifold boundary representation objects. Computer-Aided Design 1991;23(1):33–9.