Algorithmic aspects of alternating sum of volumes. Part 1: Data structure and difference operation

K Tang and T Woo

In terms of basic theory, a unique conversion from a boundary representation to a CSG representation is of importance. In terms of application, the extraction of features by convex decomposition is of interest. The alternating sum of volumes (ASV) technique offers both. However, some algorithmic issues are still unresolved. The paper is the first section of a 2-part paper that addresses specialized set operations and the convergence of the ASV process. In the first part, a fast difference operation for the ASV process and a data structure for pseudopolyhedra are introduced.

A fast difference operation between an object and its convex hull is made possible by the crucial observation that it takes only linear time to distinguish them. However, it takes O(NlogN) time to construct a data structure with the proper tags. The data structure supporting the operation is a pseudopolyhedron, capturing the special relationship between an object and its convex hull. That the data structure is linear in space is also shown.

feature extraction, representation conversion, convex hull, alternating sum, difference operation, manifold data structure

The idea of the alternating sum of volumes (ASV) technique is to represent an object by a series of convex components with alternating signs (for volume addition and subtraction). It is a technique to extract 'features' from the boundary representation of a 3D component[1]. As an example, consider the object shown in Figure 1: a block with a slot and a hole. The ASV series of this object is

$$\mathbf{H}_0 - \mathbf{H}_1 + \mathbf{H}_2 - \mathbf{H}_3$$

where the $\mathbf{H}_i$s are convex.

Formally, the ASV series of an object $\boldsymbol{\Omega}$ is defined as

$$\boldsymbol{\Omega}_0 = \sum (-1)^i \mathbf{H}_i$$

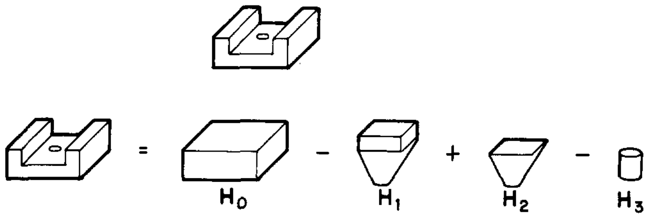


Figure 1. Alternating sum of volumes

where

$\boldsymbol{\Omega}_0 = \boldsymbol{\Omega}$

$\mathbf{H}_i$ is the convex hull of $\boldsymbol{\Omega}_i$, CH($\boldsymbol{\Omega}_i$)

$\boldsymbol{\Omega}_i$ is the deficiency, and is defined as the regularized set difference between $\mathbf{H}_{i-1}$ and $\boldsymbol{\Omega}_{i-1}$

Figure 2 shows how the terms in an ASV are derived for the object in Figure 1. The deficiency $\boldsymbol{\Omega}_i$ is obtained by subtracting $\boldsymbol{\Omega}_{i-1}$ from $\mathbf{H}_{i-1}$, where $i = 1, 2, \ldots,$. As $\boldsymbol{\Omega}_{i+1}$ becomes the null set $\varnothing$, the $\mathbf{H}_i$s are collected,

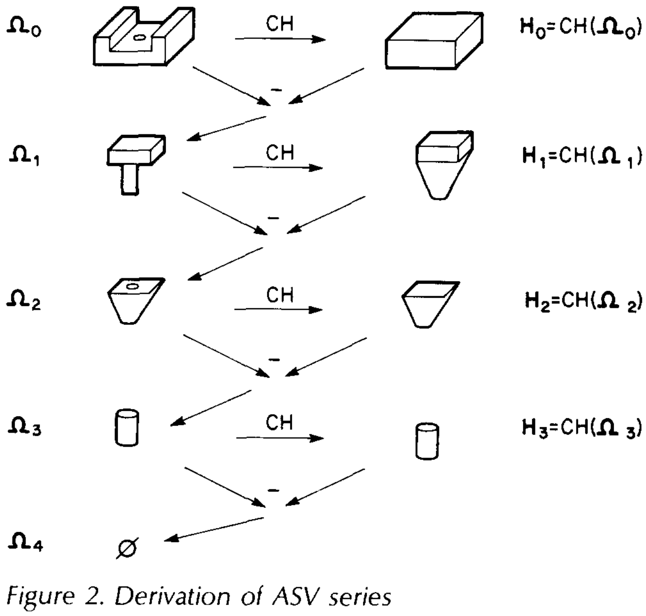


Figure 2. Derivation of ASV series
[CH = convex hull operation, —— = difference operation, $\varnothing$ = null set.]

Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109-2117, USA
Paper received: 13 June 1989. Revised: 9 March 1990

 0010-4485/91/050357-10

starting with $\mathbf{H}_0$. The ASV expression is formed by alternating '$-$' and '$+$' signs, as in $\mathbf{H}_0 - \mathbf{H}_1 + \mathbf{H}_2 -, \ldots,$.

Consider the machining process of the mechanical component in the above example. There are two features: a slot and a hole. They can be extracted by algebraic manipulation of the object's ASV as illustrated in Figure 3. Parenthesizing $\mathbf{H}_1$ and $\mathbf{H}_2$ forces a change in the sign from $+$ to $-$. Subtracting $\mathbf{H}_2$ from $\mathbf{H}_1$ yields a new $\mathbf{H}_1'$ for a disjunctive expression:

$$\mathbf{H}_0 - (\mathbf{H}_1 - \mathbf{H}_2) - \mathbf{H}_3 = \mathbf{H}_0' - \mathbf{H}_1' - \mathbf{H}_2'$$

If $\mathbf{H}_0'$ is the stock, $\mathbf{H}_1'$ and $\mathbf{H}_2'$ can be thought of as volumes to be removed to create the slot and the hole.

As another illustration of the material-joining process, consider Figure 4. The components adjacent to the $-$ sign are parenthesized, yielding a conjunctive expression:

$$\mathbf{H}_0 - \mathbf{H}_1 + \mathbf{H}_2 = (\mathbf{H}_0 - \mathbf{H}_1) + \mathbf{H}_2 = \mathbf{H}_0' + \mathbf{H}_1'$$

Here, $\mathbf{H}_0'$ is the base plate on which a protrusion $\mathbf{H}_1'$ is to be joined.

These two examples illustrate that, through the manipulation of an ASV series, features of a given object can be extracted automatically, which can in turn help the process planners in deciding on suitable manufacturing operations, such as machining or welding.

The ability to 'disassemble' allows conversion from boundary-based solid-modelling systems to those that are CSG-based[2].

As implied in the examples, the terminating condition of an ASV-series expansion is when the deficiency $\mathbf{\Omega}_n$ becomes convex, that is, when $\mathbf{H}_n$ identifies with $\mathbf{\Omega}_n$. This condition, however, is not guaranteed. Figure 5 shows an example of an infinite ASV series. It has been shown[1] that an ASV series is nonterminating if and only if there is an integer i such that $\mathbf{H}_{i+1} = \mathbf{H}_i$. In such a case, the deficiency $\mathbf{\Omega}_i$ is said to be *nonconvergent*. When a nonconvergent deficiency $\mathbf{\Omega}_i$ is encountered, the ASV expansion cannot continue.

When a deficiency $\mathbf{\Omega}_n$ becomes nonconvergent, one solution is to divide it into convex subsets[1]. However, there is a drawback. It is known[3] that there can be $O(n^3)$ convex subsets, where n is the number of concave edges, and each subset requires further polynomial time to determine. An alternative is to decompose the deficiency into subsets that are themselves convergent, so that the ASV series can expand further. For example, the object $\mathbf{\Omega}$ in Figure 6 is nonconvergent. By separating it along the edge e into two parts P_1 and P_2, and performing the ASV expansion on each of them, a finite ASV series of two branches, each of which is a finite ASV series, results. The observation that edges of the type of e in Figure 6 may be a very small subset of the set of all concave edges encourages inquiry.

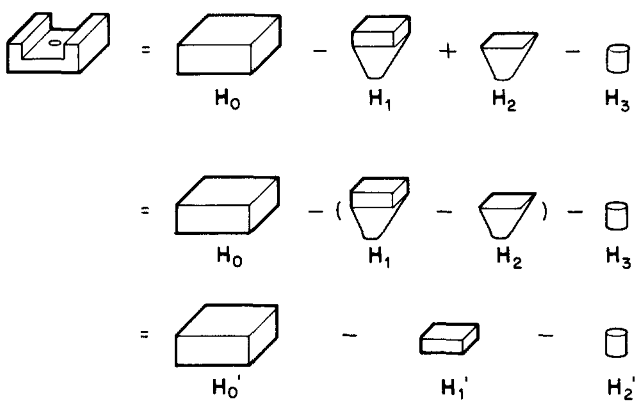


Figure 3. Algebraic manipulation of ASV series into disjunctive form

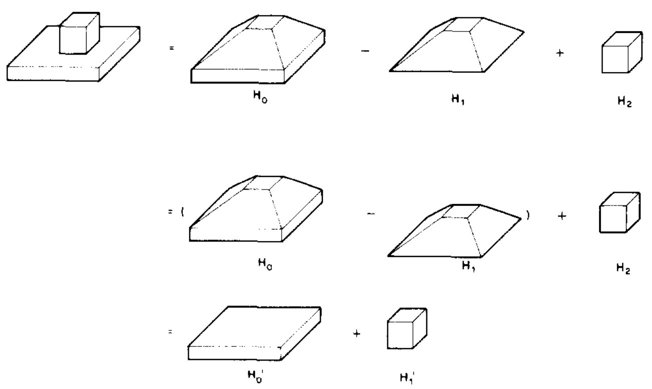


Figure 4. Algebraic manipulation of ASV series into conjunctive form

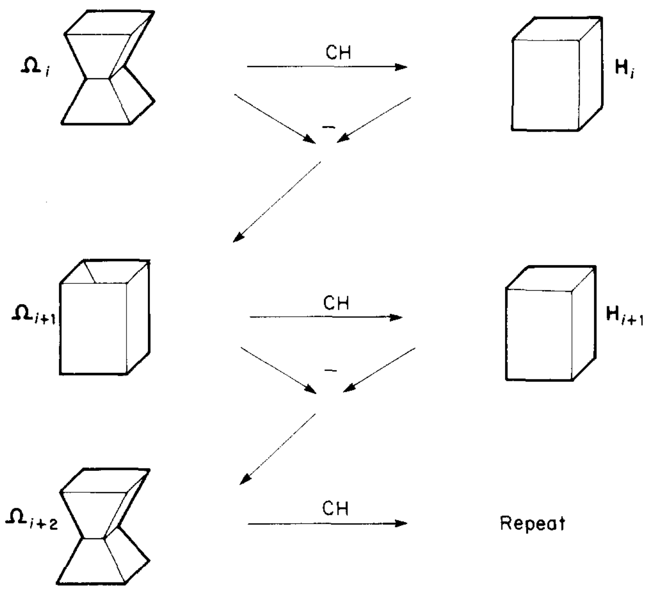


Figure 5. Example of ASV nonconvergence

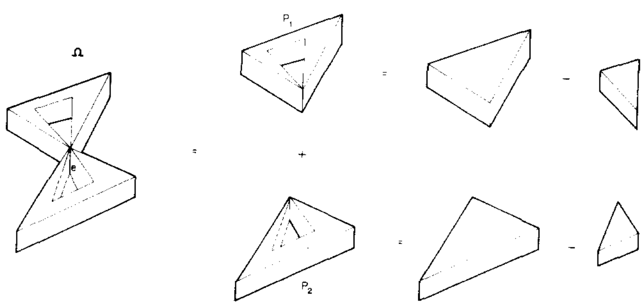


Figure 6. Remedy for nonconvergence

This paper is the first part of a 2-part paper[4]. Part 1 deals with a special difference operation. While a general difference operation[5] may be invoked, the relationship between an object and its convex hull merits investigation. This relationship is made concrete by the notion of a *pseudomanifold* object, shown as $\mathbf{\Omega}$ in Figure 6. This is an entity with characteristics between those of a 2-manifold[6] and a nonmanifold[7] object. With the aid of a data structure for pseudomanifold objects, the use of an $O(n\log n)$ time algorithm for computing a convex deficiency is made possible, where n is the number of faces of the pseudomanifold. The data structure is shown to be $O(n)$ in space, which gives an absolute upper bound to the number of edges of the type e in Figure 6.

The characterization of nonconvergence and its remedy as illustrated by Figure 6 is given in Part 2[4] of this paper.

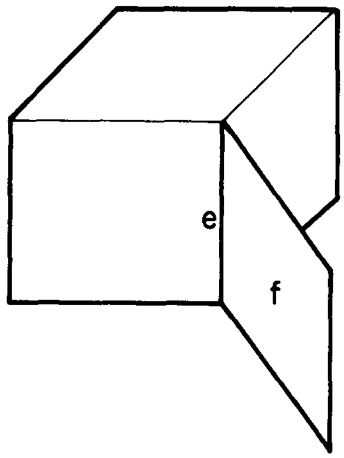


Figure 7. Points in a manifold, a pseudomanifold, and a nonmanifold

DOMAIN AND DATA STRUCTURE

In this section, the domain of objects and a data structure to represent them are given. An object $\mathbf{\Omega}$ is a set of points in 3D Euclidean space, E^3. It must satisfy certain restrictions. Because the ASV process performs operations on the boundary of volumes, each object must be a closed surface that forms the closure of an open set of finite extent in E^3. In other words, an object must be the surface of a volume, and must not have 'dangling' faces and edges[2]. In addition to this restriction of *homogeneous three-dimensionality*, the objects must also be closed under the (regularized) difference operation, i.e. they should have *differential preservability*[2]. To define a domain of objects that will meet both the restrictions, some definitions of the interior and boundary points of a 3-dimensional point set must be clarified:

> *Definition 1:* A point $\mathbf{p}$ of a set $\mathbf{S}$ in E^3 is called an *interior* point of $\mathbf{S}$ if there exists an open 3-dimensional neighbourhood that consists of points in $\mathbf{S}$ only. A point $\mathbf{p}$ is called a *boundary* point of $\mathbf{S}$ if it is not an interior point. The set $\mathbf{B}(\mathbf{S})$ of all boundary points of $\mathbf{S}$, and the set $\mathbf{I}(\mathbf{S})$ of all the interior points of $\mathbf{S}$, are defined as the *boundary* and the *interior* of $\mathbf{S}$ respectively.

The relationship between a boundary point and its neighbouring points of a set $\mathbf{S}$ in E^3 is described by one of three characterizations, namely *manifold, pseudomanifold*, and *nonmanifold*. A point $\mathbf{p}$ in $\mathbf{B}(\mathbf{S})$ is called a *2-manifold* point if it has a 3D neighbourhood such that the subset of the points of $\mathbf{S}$ contained in that neighbourhood is topologically equivalent to a hemisphere[7]. A point $\mathbf{p}$ is a *pseudomanifold* point if every 3D neighbourhood of it contains some points in $\mathbf{I}(\mathbf{S})$. If a point $\mathbf{p}$ has a 3D neighbourhood such that the subset of the points of $\mathbf{S}$ contained in that neighbourhood entirely belong to $\mathbf{B}(\mathbf{S})$, then it is called a *nonmanifold point.* As an example, the boundary surface of the object in Figure 7 consists of the six faces of the cube and a 'dangling' face f. All the boundary points except those on f (including edge e) are 2-manifold points. The boundary points on the six faces of the cube, including edge e, are pseudomanifold points. The nonmanifold points are those on face f but not on edge e.

> *Definition 2:* A point set $\mathbf{S}$ in E^3 is a *2-manifold* set if $\mathbf{I}(\mathbf{S})$ is connected, and every point in $\mathbf{B}(\mathbf{S})$ is a 2-manifold point. $\mathbf{S}$ is a *pseudomanifold* set if every point in $\mathbf{B}(\mathbf{S})$ is a pseudomanifold point. $\mathbf{S}$ becomes a *nonmanifold* set if $\mathbf{B}(\mathbf{S})$ contains some nonmanifold points.

A pseudomanifold point is a relaxation of a 2-manifold point, i.e. it only requires that every neighbourhood of the point contains some interior points of the set, but with no topological constraint on the neighbourhoods. A pseudomanifold set need not be a connected set either. The relationship of these three sets can best be described in Figure 8. Because an object must have homogeneous three dimensionality, nonmanifold sets are immediately excluded from consideration. Although 2-manifold sets satisfy the homogenous 3-dimensionality condition, they are not closed under regularized difference operation[2]. Pseudomanifold sets, while still conforming to homogeneous three dimensionality but also guaranteeing differential preservability[2], prove to be the only clan of objects suitable for ASV representations. Figure 9 shows several examples of 2-manifold, pseudomanifold and nonmanifold objects.

A data structure for the pseudomanifolds is crucial to both the development and analysis of algorithms. A

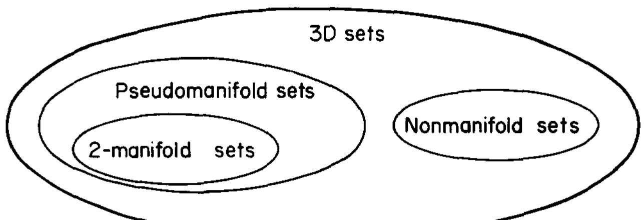


Figure 8. Relationship between 2-manifold sets, pseudomanifold sets and nonmanifold sets

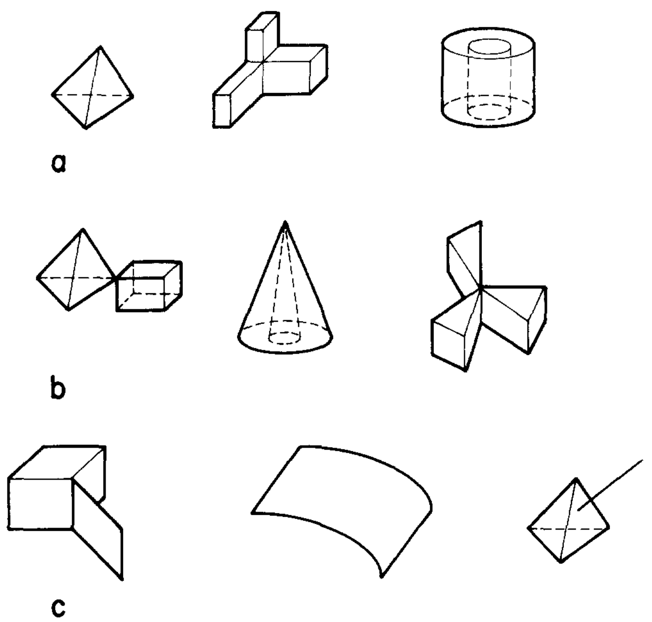


Figure 9. (a) 2-manifold, (b) pseudomanifold and (c) nonmanifold objects

data structure for polyhedra[8] is not suitable because a pseudomanifold could have more than two faces meeting at an edge (see Figure 9*b*). The representation of general nonmanifolds[7] is more than that needed here, because of the difference preservability of pseudomanifolds. The concept of *pseudopolyhedra* is proposed. A pseudopolyhedron is 'almost' a polyhedron, except that it allows edges to have more than two adjacent faces.

> *Definition 3:* A *pseudopolyhedron* P is a finite collection of planar faces such that (a) every edge of P has at least two adjacent faces, and (b) if any two faces meet, they meet at a common edge. Specifically, a pseudopolyhedron with n_v vertices, n_e edges and n_f faces is a quintuple $\langle V, E, F, NORM, E_f \rangle$, which is defined as:

- $V = \{v_1, v_2, \ldots, v_{n_v}\}$ is a list storing the n_v vertices; each v_i is a coordinate triple (x_i, y_i, z_i).
- $E = \{\langle v_{1,1}, v_{1,2}\rangle, \langle v_{2,1}, v_{2,2}\rangle, \ldots, \langle v_{n_e,1}, v_{n_e,2}\rangle\}$ is the edge list. Each entry $\langle v_{i,1}, v_{i,2}\rangle$ stands for an edge, with $v_{i,1}$ and $v_{i,2}$ being indices of the two end points; e.g. $\langle v_{i,1}, v_{i,2}\rangle = \langle 3, 10\rangle$ means that the end points of the *i*th edge are v_3 and v_{10}.
- $F = \{F_1, F_2, \ldots, F_{n_f}\}$ stores face information. Each element F_i is itself an array of the form $\{\langle e_{1,1}, e_{1,2}, \ldots, e_{1,i_1}\rangle, \langle e_{2,1}, e_{2,2}, \ldots, e_{2,i_2}\rangle, \ldots, \langle e_{k,1}, e_{k,2}, \ldots, e_{k,i_k}\rangle\}$, where *k* is the total number of polygons in face F_i. Each $\langle e_{j,1}, e_{j,2}, \ldots, e_{j,i_j}\rangle$ is a simple polygon, and each $e_{j,1}$ is the index of an edge in E. For example, $F_i = \{\langle 2, 4, 1\rangle, \langle 5, 7, 6, 3\rangle\}$ means that face F_i is bounded by two simple polygons; the indices of the edges of the outer polygon are 2, 4, 1, and they are 5, 7, 6, 3 for the inner polygon. The edges are ordered clockwise (for the outer polygon) or counterclockwise (for the inner polygon).
- $NORM = \{N_1, N_2, \ldots, N_{n_f}\}$ stores the outward normals of the n_f faces.
- $E_f = \{\langle f_{1,1}, f_{1,2}, \ldots, f_{1,k_1}\rangle, \langle f_{2,1}, f_{2,2}, \ldots, f_{2,k_2}\rangle, \ldots, \langle f_{n_e,1}, f_{n_e,2}, \ldots, f_{n_e,k_{n_e}}\rangle\}$ is a list that describes the edge–face adjacency relationship. An entry of $\langle f_{i,1}, f_{i,2}, \ldots, f_{i,k_i}\rangle$ says that edge E_i has k_i adjacent faces, and that the indices for them are $f_{i,1}, f_{i,2}, \ldots, f_{i,k_i}$. For example, if $\langle f_{i,1}, f_{i,2}, \ldots, f_{i,k_i}\rangle$ is $\langle 1, 7, 6, 2\rangle$, then edge E_i has four adjacent faces, and their indices in F are 1, 7, 6 and 2. Each k_i is defined as the *face adjacency index* of edge E_i.

The data structure given above, although quite simple, completely describes one family of pseudomanifolds: planar pseudomanifolds. (Pseudomanifolds with nonplanar boundary surfaces are not considered in this paper.) A detailed proof is given in the Appendix that the space requirement of a pseudopolyhedron is linear in the number of its faces. It should be noted that, although a pseudopolyhedron completely describes the boundary of a pseudomanifold, it carries no set-theoretic information itself. It is the pseudomanifold, which a pseudopolyhedron *represents*, that possesses the set in E^3.

DIFFERENCE OPERATION

To study the difference operation between a pseudomanifold and its convex hull, the following notations are used: $\mathbf{\Omega}$ denotes a pseudomanifold object, $\mathbf{\Omega}_h$ its convex hull, and $\mathbf{\Omega}_d$ the deficiency $\mathbf{\Omega}_h - \mathbf{\Omega}$. The same notations are used to represent their defining pseudopolyhedra unless noted otherwise.

Consider the convex hull $\mathbf{\Omega}_h$ of an object $\mathbf{\Omega}$. The set $\mathbf{\Omega}_h$ can be divided into four disjoint subsets. They are:

ξ_i: $\{\mathbf{I}(\mathbf{\Omega})\}$, the interior points of $\mathbf{\Omega}$,

ξ_h: $\{\mathbf{B}(\mathbf{\Omega}_h)\}$, the boundary (hull) points of $\mathbf{\Omega}_h$,

ξ_p: $\{\mathbf{B}(\mathbf{\Omega}) \cap \mathbf{I}(\mathbf{\Omega}_h)\}$, the boundary points of $\mathbf{\Omega}$ excluding those that are also in ξ_h,

ξ_d: $\{\mathbf{I}(\mathbf{\Omega}_h) - \mathbf{\Omega}\}$, interior (deficiency) points of $\mathbf{\Omega}_h$ excluding $\mathbf{\Omega}$.

Figure 10 shows these subsets in two dimensions.

> *Definition 4:* A point $\mathbf{p} \in \mathbf{\Omega}_h$ is a *preserved point* if, for any real number $\varepsilon > 0$, no matter how small it is, the open neighbourhood sphere that centres at $\mathbf{p}$ with radius ε always has a point in ξ_d. A point $\mathbf{p}$ is a *lost point* if it is not a preserved point.

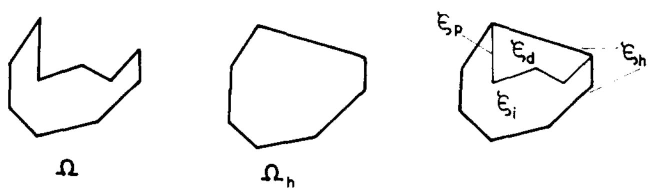


Figure 10. Illustrations of ξ_i, ξ_h, ξ_p and ξ_d

The difference operation between $\mathbf{\Omega}_h$ and $\mathbf{\Omega}$ is defined as follows:

Definition 5: The *deficiency* $\mathbf{\Omega}_d$ of an object $\mathbf{\Omega}$ is a set in E^3 consisting of all the preserved points of $\mathbf{\Omega}_h$.

The definition of deficiency here correlates with the intuitive view. The purpose of categorizing the points of $\mathbf{\Omega}_h$ is to relate the interior and the boundary of the deficiency $\mathbf{\Omega}_d$ to those of $\mathbf{\Omega}$ and its convex hull $\mathbf{\Omega}_h$. It is easy to see that, because both $\mathbf{I}(\mathbf{\Omega})$ and $\mathbf{I}(\mathbf{\Omega}_h)$ are open sets, so must the sets ξ_i and ξ_d be. By definition (of the deficiency), all the points of ξ_i are lost. As ξ_d is an open set, every point in it has a neighbourhood sphere of points in ξ_d only and is thus interior to $\mathbf{\Omega}_d$. This means that ξ_d must be a subset of the interior of deficiency $\mathbf{\Omega}_d$. The neighbourhood of any point in ξ_p or ξ_h contains some points in either ξ_i or $\{E^3 - \mathbf{\Omega}_h\}$, i.e. it contains points not belonging to $\mathbf{\Omega}_d$, and hence, by definition, they cannot be in the interior of $\mathbf{\Omega}_d$. These observations are summarized in the following lemma.

Lemma 1: The deficiency $\mathbf{\Omega}_d$ of a pseudomanifold $\mathbf{\Omega}$ is also a pseudomanifold, whose interior set $\mathbf{I}(\mathbf{\Omega}_d)$ is the ξ_d set of $\mathbf{\Omega}_h$, and whose boundary set $\mathbf{B}(\mathbf{\Omega}_d)$ is a subset of $\{\xi_h \cup \xi_p\}$.

As noted in Lemma 1, the boundary surface $\mathbf{B}(\mathbf{\Omega}_d)$ of the deficiency $\mathbf{\Omega}_d$ is a subset of $\{\xi_h \cup \xi_p\}$ of $\mathbf{\Omega}_h$. Because of the planarity of the faces, $\mathbf{B}(\mathbf{\Omega}_d)$ must be a set of some faces of $\mathbf{\Omega}$ and $\mathbf{\Omega}_h$. The key to the algorithm for finding the deficiency is to find these faces and the adjacency relationship between them, so that the result is a pseudopolyhedron representation of $\mathbf{\Omega}_d$.

Definition 6: A face of a pseudomanifold $\mathbf{\Omega}$ is a *hull face* if it is in ξ_h; otherwise it is called an *internal face*.

Lemma 2: A hull face f of $\mathbf{\Omega}$ will not exist in the boundary surface $\mathbf{B}(\mathbf{\Omega}_d)$.

Proof: For any point in the interior set $\mathbf{I}$(f) of f, say $\mathbf{p}$, there must exist an open neighbourhood ε of $\mathbf{p}$ that belongs to $\mathbf{I}$(f) and hence in ξ_h only. As $\mathbf{\Omega}$ is a pseudomanifold, every point in ε, including $\mathbf{p}$ itself, must have a neighbourhood sphere that consists of points in $\{\xi_h \cup \xi_i \cup \{E^3 - \mathbf{\Omega}_h\}\}$ only. Because of the way that a preserved point is defined, $\mathbf{p}$ can only be a lost point. Therefore, every point in set $\mathbf{I}$(f) is a lost point.

QED

Lemma 2 asserts that the boundary surface $\mathbf{B}(\mathbf{\Omega}_d)$ consists of only the internal faces of $\mathbf{\Omega}$ and the faces

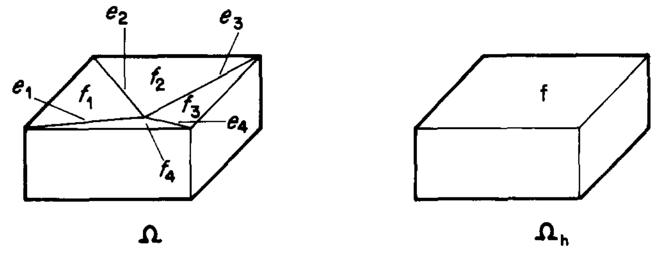


Figure 11. Functionality of procedure HULLP

[F_H: $F_H(f) = 0$; all other F_Hs are 1.
E_H: All E_Hs are nonzero.
F_I: $F_I(f_1) = F_I(f_2) = F_I(f_3) = F_I(f_4) = 1$; all other F_Is are 0.
E_I: $E_I(e_1) = E_I(e_2) = E_I(e_3) = E_I(e_4) = 1$; all other E_Is are 0.]

of $\mathbf{\Omega}_h$ but not the hull faces of $\mathbf{\Omega}$. This observation leads to the development of the desired difference algorithm. It accepts as input (V, E, F, E_f, $NORM_f$), which is the pseudopolyhedron representation of a pseudomanifold $\mathbf{\Omega}$, and it outputs the pseudopolyhedron representation of the deficiency $\mathbf{\Omega}_d$. Suppose that there is a procedure HULLP that takes as input the pseudopolyhedron (V, E, F, E_f, $NORM_f$) of a pseudomanifold $\mathbf{\Omega}$. Its outputs are two: one is the pseudopolyhedron of the resultant convex hull $\mathbf{\Omega}_h$, and the other is two arrays F_H and E_H, called *hull tag arrays*, that distinguish those faces and edges of $\mathbf{\Omega}_h$ that do not belong to $\mathbf{\Omega}$. Specifically, $F_H(i) = 1$ means that face i of $\mathbf{\Omega}_h$ is a face of $\mathbf{\Omega}$. When $F_H(i) = 0$, the meaning is reversed. $E_H(i) = j$ means that edge i of $\mathbf{\Omega}_h$ is edge j of $\mathbf{\Omega}$, whereas $E_H(i) = 0$ means that edge i is not an edge of $\mathbf{\Omega}$. As most available 3D convex-hull algorithms[9,10] support data structures that embody our pseudopolyhedra, the feasibility of the output of HULLP is justified. For convenience of computation, it is also assumed that the vertex array V is unchanged through HULLP, although redundant vertices in the V array of $\mathbf{\Omega}_h$ are implied. Also, there are two additional arrays, F_I and E_I. They are the *internality tag arrays*, which identify internal faces and internal edges of $\mathbf{\Omega}$. Specifically, $F_I(i) = 1$ means that face i of $\mathbf{\Omega}$ is an internal face, and, similarly, $E_I(i) = 1$ means that edge i of $\mathbf{\Omega}$ is an internal edge. Similarly, when $F_I(i)$ or $E_I(i)$ is equal to 0, the meaning is reversed. These two arrays are $O(n \log n)$ derivable from $\mathbf{\Omega}$, because the internality of any face f (or edge e) can be identified by checking the internality of an arbitrary point of $\mathbf{I}$(f) (or $\mathbf{I}$(e)). Figure 11 demonstrates the functionality of procedure HULLP on a pseudopolyhedron.

The first algorithm MERGE given below adds those edges and faces of $\mathbf{\Omega}_h$ that do not belong to $\mathbf{\Omega}$ to the description arrays E and F of $\mathbf{\Omega}$, and updates E_f correspondingly. A constant time function named INSERT_E_f(E_f, i, j) is used. It either sets $E_f(i)$ to 'j' if $E_f(i)$ is not previously defined, or appends 'j' to $E_f(i)$.

MERGE algorithm

Procedure MERGE (n_v, n_e, n_f, V, E, F, E_f, $NORM_f$, F_I, E_I,
n'_v, n'_e, n'_f, V', E', F', E'_f, $NORM'_f$, F_H, E_H)

/*purpose: updates the pseudopolyhedron representation of a pseudomanifold $\mathbf{\Omega}$ by adding the newly generated hull faces and hull edges of its convex hull to it.

```
input:     (n_v, n_e, n_f, V, E, F, E_f, NORM_f, F_I, E_I)------pseudopolyhedron Ω; F_I and E_I
           are the internality tag arrays of its faces and edges.
           (n'_v, n'_e, n'_f, V', E', F', E'_f, NORM'_f, F_H, E_H)------pseudopolyhedron
           representation of the convex hull Ω_h of Ω; F_H and E_H are the hull tag arrays of its faces and edges.
output:    (n_v, n_e, n_f, V, E, F, E_f, NORM_f, F_I, E_I)------updated pseudopolyhedron of Ω
           with the newly generated hull faces and hull edges of Ω_h added; F_I and E_I
           are updated with the following convention:
             F_I(i) = 0  face i is a hull face of Ω
                    = 1  face i is an internal face
                    = 2  face i is a face of Ω_h, but not a face of Ω
             E_I(i) = 0  edge i is a hull edge of Ω
                    = 1  edge i is an internal edge
                    = 2  edge i is an edge of Ω_h, but not an edge of Ω

*/
begin
 (1)       ETOP = n_e
           FTOP = n_f
           for i = 1, n'_e do
             EMAP(i) = 0
           end do {step 1}
 (2)       for i = 1, n'_f do
             if F_H(i) = 0 then
 (3)           FTOP ← FTOP + 1
 (4)           (e_1, e_2, ..., e_l) ← F'(i)
 (5)           for j = 1, l do
                 if E_H(e_j) = 0 then
                   if EMAP(e_j) = 0 then
 (6)                 ETOP ← ETOP + 1
 (7)                 EMAP(e_j) ← ETOP
 (8)                 E(ETOP) ← E'(e_j)
 (9)                 E_I(ETOP) ← 2
                   end if
                   else
 (6')                EMAP(e_j) ← E_H(e_j)
                   end if
 (10)              call INSERT_E_f(E_f, EMAP(e_j), FTOP)
                 end do {step 5}
 (11)            F(FTOP) ← (1, (EMAP(e_1), EMAP(e_2), ..., EMAP(e_l)))
 (12)            NORM_f(FTOP) ← NORM'_f(i)
 (13)            F_I(FTOP) ← 2
             end if
           end do {step 2}
 (14)      n_f = FTOP
           n_e = ETOP
end {MERGE}
```

Comments on MERGE

Step 1 initializes two stack pointers FTOP and ETOP, which stand for the numbers of current faces and edges in $\mathbf{\Omega}$, respectively. Array EMAP is the index mapping between E' and E, e.g. EMAP(i) = j means that edge i of $\mathbf{\Omega}_h$ is edge j of (current) $\mathbf{\Omega}$. Steps 3–13 are performed once for each face of $\mathbf{\Omega}_h$ that is not a face of $\mathbf{\Omega}$ ($F_H = 0$). For each edge of a selected face, whether it is also an edge of $\mathbf{\Omega}$ is first checked. If it is not (when its $E_H = 0$), and it has not been previously added to $\mathbf{\Omega}$, it is then added to E with its E_I set to 2 and its EMAP set to a unique number ETOP (see steps 6–9). Otherwise, its EMAP is assigned with its E_H, which is the index of this edge in the original $\mathbf{\Omega}$ (step 6'). At step 10, the face-adjacency relationship of this edge in $\mathbf{\Omega}$ is updated, as reflected by the insertion of this selected face. Steps 11 and 12 append the selected face and its normal to F and NORM_f of $\mathbf{\Omega}$. (Note that, because $\mathbf{\Omega}_h$ is convex, every face of it has only one bounding polygon.) Step 13 assigns 2 to the F_I of this face that indicates that the added face is not a face of the original $\mathbf{\Omega}$. To analyse the time requirement of MERGE, note that each edge of $\mathbf{\Omega}_h$ has exactly two adjacent faces in $\mathbf{\Omega}_h$. At most, an edge of $\mathbf{\Omega}_h$ will be checked, retrieved

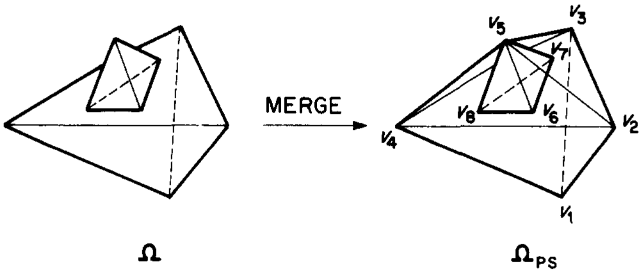


Figure 12. Example of PS description $\boldsymbol{\Omega}_{PS}$

[$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$
$E = \{\langle v_1, v_2\rangle, \langle v_1, v_3\rangle, \langle v_1, v_4\rangle, \langle v_5, v_6\rangle, \langle v_5, v_7\rangle, \langle v_5, v_8\rangle, \langle v_5, v_2\rangle, \langle v_5, v_3\rangle, \langle v_5, v_4\rangle, \langle v_2, v_3\rangle, \langle v_3, v_4\rangle, \langle v_4, v_2\rangle, \langle v_6, v_7\rangle, \langle v_7, v_8\rangle, \langle v_8, v_6\rangle\}$
$F = \{\langle 1, 10, 2\rangle, \langle 1, 12, 3\rangle, \langle 11, 2, 3\rangle, \langle\langle 10, 11, 12\rangle, \langle 13, 15, 14\rangle\rangle, \langle 13, 5, 4\rangle, \langle 5, 14, 6\rangle, \langle 4, 6, 15\rangle, \langle 7, 9, 12\rangle, \langle 7, 10, 8\rangle, \langle 8, 11, 9\rangle\}$
$E_f = \{\langle 1, 2\rangle, \langle 1, 3\rangle, \langle 2, 3\rangle, \langle 5, 7\rangle, \langle 5, 6\rangle, \langle 6, 7\rangle, \langle 8, 9\rangle, \langle 9, 10\rangle, \langle 10, 8\rangle, \langle 1, 4, 9\rangle, \langle 3, 4, 10\rangle, \langle 2, 4, 8\rangle, \langle 5, 4\rangle, \langle 4, 6\rangle, \langle 4, 7\rangle\}$]

and stored twice. Steps 12 and 13 take constant time. As a result, the loop from step 2 to step 13 is $O(n'_e + n'_f)$.

The output of MERGE, called the *PS description* of $\boldsymbol{\Omega}$, and denoted as $\boldsymbol{\Omega}_{PS}$, is a pseudopolyhedron. Figure 11 lists the V, E, F, E_f entries of the P description of a pseudomanifold. $\boldsymbol{\Omega}_{PS}$ itself, however, no longer represents a legitimate pseudomanifold, as it contains all the intermediate data for obtaining the deficiency $\boldsymbol{\Omega}_d$. For the pseudomanifold $\boldsymbol{\Omega}$ in Figure 12, the boundary of its deficiency $\boldsymbol{\Omega}_d$ consists of the faces f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, as defined in the F entry of $\boldsymbol{\Omega}_{PS}$. The vertices of $\boldsymbol{\Omega}_d$ are v_2, v_3, v_4, v_5, v_6, v_7, v_8 of $\boldsymbol{\Omega}_{PS}$, and the edges of $\boldsymbol{\Omega}_d$ are e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15} of $\boldsymbol{\Omega}_{PS}$. The procedure DIFFBUILD given below will generate $\boldsymbol{\Omega}_d$ using $\boldsymbol{\Omega}_{PS}$. A constant time-routine called INSERT(L, i) will be used in the algorithm which appends an integer i into an integer list L.

DIFFBUILD procedure

```
Procedure   DIFFBUILD (n_v, n_e, n_f, V, E, F, E_f, NORM_f, F_l, E_l,
                       n_dv, n_de, n_df, V_d, E_d, F_d, E_df, NORM_df)
/*purpose:   finds the deficiency Ω_d of a pseudomanifold Ω, and outputs the pseudopolyhedron
             representation of Ω_d to the external.
input:       (n_v, n_e, n_f, V, E, F, E_f, NORM_f, F_l, E_l)------the PS description of Ω.
output:      (n_dv, n_de, n_df, V_d, E_d, F_d, E_df, NORM_df)------the pseudopolyhedron
             representation of deficiency Ω_d.
*/
begin
(1)          n_df ← 0
             n_de ← 0
             n_dv ← 0
(2)          for i = 1, n_f, do
               if F_l(i) ≠ 1 then
(3.1)            n_df ← n_df + 1
(3.2)            F_d(n_df) ← F(i)
(3.3)            FMAP(i) ← n_df
                 if F_l(i) = 0 then
                   NORM_df(n_df) ← NEG(NORM(i))
                 else
(3.4')             NORM_df(n_df) ← NORM(i)
                 end if
               end if
             end do {step 2}

(4)          for i = 1, n_v, do
               VMAP(i) ← 0
             end do {step 4}
(5)          for i = 1, n_e, do
(5.1)          ⟨f_1, f_2, ..., f_k⟩ ← E_f(i)
(5.2)          NewE_fi ← nil
(5.3)          for j = 1, k, do
                 if F_l(f_j) ≠ 1 then
(5.4)              call INSERT(NewE_fi, FMAP(f_j))
                 end if
               continue {step 5.3}
               if NewE_fi ≠ nil then
(5.5)            n_de ← n_de + 1
(5.6)            E_df(n_de) ← NewE_fi
(5.7)            EMAP(i) ← n_de
```

```
(5.8)         ⟨v_1, v_2⟩ ← E(i)
              if VMAP(v_1) = 0 then
(5.9)            n_dv ← n_dv + 1
                 V_d(n_dv) ← V(v_1)
                 VMAP(v_1) ← n_dv
              end if
              if VMAP(v_2) = 0 then
(5.10)           n_dv ← n_dv + 1
                 V_d(n_dv) ← V(v_2)
                 VMAP(v_2) ← n_dv
              end if
(5.11)        E_d(n_de) ← (VMAP(v_1), VMAP(v_2))
           end if
        end do {step 5}

(6)     for i = 1, n_df, do
           ⟨e_1, e_2, ..., e_k⟩ ← F_d(i)
           F_d(i) ← ⟨EMAP(e_1), EMAP(e_2), ..., EMAP(e_k)⟩
        end do {step 6}

end DIFFBUILD
```

Comments on DIFFBUILD

Variables n_{df}, n_{de} and n_{dv} are the numbers of faces, edges and vertices of $\boldsymbol{\Omega}_d$ that have been found. Three arrays FMAP, EMAP and VMAP are the mappings from the preserved faces, edges and vertices of $\boldsymbol{\Omega}_{PS}$ to those of $\boldsymbol{\Omega}_d$. For example, FMAP(5) = 2 means face 5 of $\boldsymbol{\Omega}_{PS}$ is face 2 of $\boldsymbol{\Omega}_d$. At step 1, the total number of faces, edges and vertices in $\boldsymbol{\Omega}_d$ is 0. The loop at step 2 generates the NORM_f set NORM_{df} of $\boldsymbol{\Omega}_d$; if a face of $\boldsymbol{\Omega}_{PS}$ is an internal face of $\boldsymbol{\Omega}$, it is preserved on $\boldsymbol{\Omega}_d$, and its normal should be negated (step 3.4). Otherwise, it is also preserved, but its normal should be the same as the original (step 3.4′). Step 3.2 retrieves the current preserved face of $\boldsymbol{\Omega}_{PS}$ into the F_d set of $\boldsymbol{\Omega}_d$, while step 3.3 establishes the index mapping FMAP between them. The edge indices of the faces in F_d are still the originals from E, and they will be mapped to E_d once EMAP is established. The mapping VMAP is initialized at step 4. The entire loop of step 5 generates the V, E and E_f arrays of $\boldsymbol{\Omega}_d$, V_d, E_d and E_{df}, as well as establishing the mappings VMAP and EMAP. Step 5.1 retrieves all the faces of $\boldsymbol{\Omega}_{PS}$ that are adjacent at an edge of $\boldsymbol{\Omega}_{PS}$. By means of checking their F_is, those unpreserved faces (steps 5.3–5.4) are filtered out. If the remainder is not empty, this edge as well as its end points must be preserved on $\boldsymbol{\Omega}_d$. This is done as follows: step 5.6 inserts the ($\boldsymbol{\Omega}_d$) face-adjacency relationship of the edge into the E_f array E_{df} of $\boldsymbol{\Omega}_d$; step 5.7 assigns the mapping EMAP of the edge. Steps 5.9 and 5.10 establish the mapping VMAP of the two end points of the edge, and store their coordinates from V of $\boldsymbol{\Omega}_{PS}$ into the V array V_d of $\boldsymbol{\Omega}_d$. At step 5.11, this edge, with its new end points indices of V_d, is stored into the E array E_d of $\boldsymbol{\Omega}_d$. Finally, at step 6, the edge indices in F_d are replaced with their mappings in E_d.

Theorem 1: The deficiency $\boldsymbol{\Omega}_d$ of a pseudomanifold $\boldsymbol{\Omega}$ can be obtained in $O(N\log N + k)$ time, where N is $\max\{n_e, n_f, n_v\}$ of $\boldsymbol{\Omega}$, and k is the sum of the face-adjacency indices in $\boldsymbol{\Omega}$.

Proof: As the $\boldsymbol{\Omega}_{PS}$ of $\boldsymbol{\Omega}$ is $O(N\log N)$-derivable from $\boldsymbol{\Omega}$ (see the comments on the MERGE procedure), it is only necessary to analyse the procedure DIFFBUILD. The overall time taken from step 1 to step 4 is $O(N)$. The total time for the loop at step 5 plus the inner loop at step 5.3 is $O(k)$. Analogously, the loop at step 6 is $O(k)$ as well.

QED

The occurrence of k can be somewhat unpleasant, because of its seemingly nondeterministic relationship with N. Fortunately, k is shown to be $O(n_f)$ where n_f is the total number of the faces of the pseudomanifold (see Appendix). Therefore, the deficiency of a pseudomanifold can be obtained in $O(N\log N)$ time.

SUMMARY

An ASV expression of an object $\boldsymbol{\Omega}$ is based on two operations, convex-hull and difference. It is known that the convex-hull operations take $O(N\log N)$ time[9], where N is the number of vertices in $\boldsymbol{\Omega}$. In this paper, it is shown that the difference between an object $\boldsymbol{\Omega}$ and its convex hull $\text{CH}(\boldsymbol{\Omega})$ is also $O(N\log N)$ in time.

Although the time to find the deficiency of a given $\boldsymbol{\Omega}$ is only $O(N)$, it takes $O(N\log N)$ time to construct its pseudopolyhedral representation $\boldsymbol{\Omega}_{PS}$. To support the data structure and the computation, manifolds, pseudomanifolds, and nonmanifolds are distinguished.

REFERENCES

1 **Woo, T C** 'Feature extraction by volume decomposition' *Proc. Conf. CAD/CAM Technology in*

Mechanical Engineering Cambridge, Massachusetts, USA (1982)

2 **Requicha, A A G** 'Representation for rigid solids: theory, methods and systems' *ACM Comput. Surv.* Vol 12 No 4 (1980) pp 437–464

3 **Chazelle, B M** 'Convex decomposition of polyhedra' *ACM Symp. Theory of Computing* Milwaukee, USA (1981) pp 70–79

4 **Woo, T and Tang, K** 'Algorithmic aspects of alternating sum of volumes. Part 2: Nonconvergence and its remedy' *Comput.-Aided Des.* Vol 23 (1991) *to be published*

5 **Requicha, A A G and Voelcher, H B** 'Constructive solid geometry' *TM-25* Production Automation Project, University of Rochester, USA (Nov 1977)

6 **Bollobas, C** *Graph Theory* Springer Verlag (1979)

7 **Weiler, K** 'Topological structures for geometric modeling' *PhD Dissertation* Rensselaer Polytechnic Institute, USA (Aug 1986)

8 **Woo, T C** 'A combinatorial analysis of boundary data structure schemata' *IEEE Comput. Graph. Applic.* Vol 5 No 3 (1985) pp 19–27

9 **Preparata, F P and Shamos, M L** *Computational Geometry* Springer Verlag (1985)

10 **Preparata, F P and Hong, S J** 'Convex hull of finite sets of points in two and three dimensions' *Commun. ACM* Vol 2 No 20 (1977) pp 87–93

BIBLIOGRAPHY

Baer, A, Eastman, C and Henrion, M 'Geometric modeling: a survey' *Comput.-Aided Des.* Vol 11 No 5 (1979) pp 253–272

Cary, A 'Build users' guide' *Document No 102* CAD Group, University of Cambridge, UK (Nov 1979)

Chiyokura, H and Kimura, F 'A method of representing the solid design process' *IEEE Comput. Graph. Applic.* Vol 5 No 4 (1985) pp 32–41

Requicha, A A G and Voelcker, H B 'Solid modeling: current status and research directions' *IEEE Comput. Graph. Applic.* Vol 3 No 7 (1983) pp 25–37

APPENDIX

Space linearity of a pseudopolyhedron

In this appendix, the space required by a pseudopolyhedron is shown to be linear in the number of its faces. Referring to Definition 3, let P = ⟨V, E, F, NORM, E_f⟩ be a pseudopolyhedron with n_v vertices, n_e edges and n_f faces. The numbers n_v and n_e are shown to be both equal to $O(n_f)$ (for the items V and E). In addition, the sum of the face-adjacency indices of all the edges is shown to be linear in n_f (for F and E_f). The face-adjacency index of an edge is the number of the faces that meet at that edge. First, two definitions are introduced:

- *Well adjacency of edges:* An edge of a pseudopolyhedron is called a *well adjacent edge* if its face-adjacency index is 2; otherwise, it is an *ill adjacent edge*.
- *Well orientation of vertices:* A vertex v of a pseudopolyhedron is said to be *well oriented* if the faces that are incident at v have an order $f_1, f_2, \ldots, f_k$ such that f_1 is adjacent to f_2, f_2 is adjacent to f_3, $\ldots$, f_{k-1} is adjacent to f_k, and f_k is adjacent to f_1; otherwise, v is an *ill oriented vertex*.

Figure 13 shows an example of well adjacency and well orientation. Based on these two characterizations of edges and vertices, two operations are defined on the ill oriented vertices and ill adjacent edges:

- *Vertex homogenization:* A *vertex-homogenizing operation* (VHO) on an ill oriented vertex v is a replacement by a set of new vertices $(v_1, v_2, \ldots, v_m)$ such that all the v_is are well oriented, and have the same coordinates as v.
- *Edge homogenization:* An *edge-homogenizing operation* (EHO) on an ill adjacent edge e is a replacement by a set of new edges $(e_1, e_2, \ldots, e_m)$ such that all the e_is are well adjacent, and have the same coordinates as e.

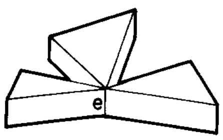
Figures 14 and 15 show VHO and EHO operations, respectively.

An operation on a pseudopolyhedron is defined below using these two microoperations VHO and EHO.

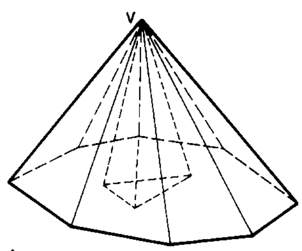
Polyhedron homogenization: A *polyhedron-homogenizing operation* (PHO) on a pseudopolyhedron P is a series of VHOs and EHOs such that the resultant pseudopolyhedron P′ has well oriented vertices and well adjacent edges only (see Figure 16).

Lemma A: The resultant pseudopolyhedron P′ of a PHO on a pseudopolyhedron P is either a single polyhedron or a set of polyhedra.

Proof: Note that a polyhedron is a special case of pseudopolyhedra such that (a) all the faces of it are connected, and (b) all its vertices are well oriented



a



b

Figure 13. Well adjacency and well orientation. (a) All the edges except e are well adjacent, (b) all the vertices except v are well oriented

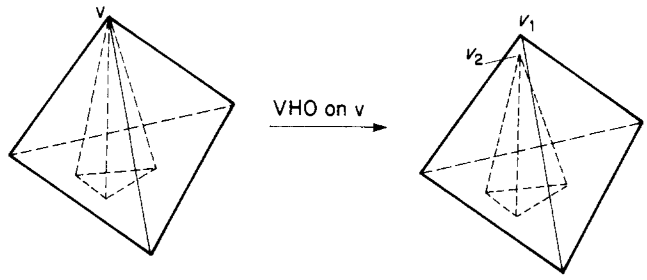


Figure 14. Vertex-homogenizing operation
[Vertices v_1 and v_2 have the same coordinates as v.]

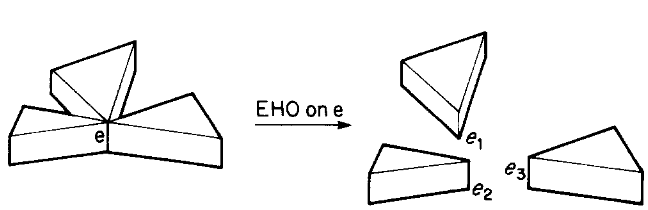


Figure 15. Edge-homogenizing operation
[Edges e_1, e_2, and e_3 have the same coordinates as e.]

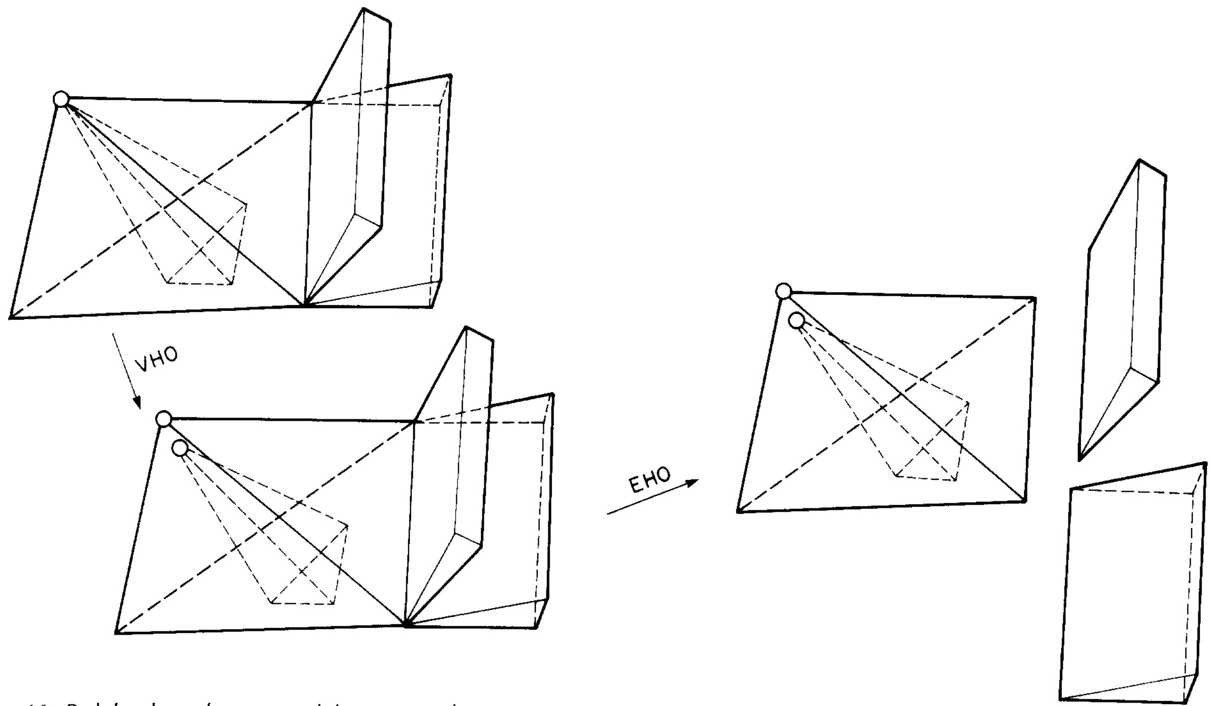


Figure 16. Polyhedron homogenizing operation

and all its edges are well adjacent. By definition, a PHO has the property (b) but not always (a).

QED

The following theorem is induced by Lemma A.

Lemma B: Let $P(n_v, n_e, n_f)$ be a pseudopolyhedron with n_v vertices, n_e edges and n_f faces. The following are then true:

n_v is $O(n_f)$,

n_e is $O(n_f)$,

$K = \Sigma k_i$ is $O(n_f)$

where k_i is the face-adjacency index of edge e_i ($i = 1, 2, \ldots, n_e$).

Proof: Let $P_1, P_2, \ldots, P_m$ be the polyhedra of $P'(n'_v, n'_e, n_f)$, which is the resultant pseudopolyhedron of a PHO on P; each of them has V_i vertices, E_i edges and F_i faces ($i = 1, \ldots, m$). By the Euler formula[6], $V_i \leqslant 2F_i - 4$ and $E_i \leqslant 3F_i - 6$ ($i = 1, \ldots, m$). Summing both sides of the inequality yields $n'_v = V_1 + V_2 + \ldots + V_m \leqslant 2(F_1 + F_2 + \ldots + F_m) - 4m = 2n_f - 4m$, and $n'_e = E_1 + E_2 + \ldots + E_m \leqslant 3(F_1 + F_2 + \ldots + F_m) - 6m = 3n_f - 6m$. As $n_v \leqslant n'_v$ and $n_e \leqslant n'_e$, $n_v \leqslant 2n_f - 4m$ and $n_e \leqslant 3n_f - 6m$. To prove (c), let L be the total number of ill adjacent edges of P. Each EHO operation replaces an ill adjacent edge of P with a number of well adjacent edges. For an ill adjacent edge e_i, exactly $(k_i/2) - 1$ new edges will be generated. This implies, however, that the sum $K' = \Sigma k_i$ over all the L ill adjacent edges of P is $2(n'_e - n_e) + 2L$. The sum $K'' = \Sigma k_i$ over the rest of the $n_e - L$ well adjacent edges of P is certainly $2(n_e - L)$. Therefore, $K = K' + K'' = 2n'_e \leqslant 6n_f - 12m$.

QED