MM Research Preprints, 89–116
KLMM, AMSS, Academia Sinica
Vol. 24, December 2005

nD Polyhedral Scene Reconstruction from Single 2D Line Drawing by Local Propagation

Hongbo Li
Key Laboratory of Mathematics Mechanization
Academy of Mathematics and Systems Science, CAS
Beijing 100080, P.R.China

Abstract. In this paper, we study the problem of reconstructing the polyhedral structures and geometric positions of a general nD polyhedral scene from a single 2D line drawing. With the idea of local construction and propagation, we propose several powerful techniques for structural reconstruction (i.e. face identification) and geometric reconstruction (i.e. realizability and parametrization). Our structural reconstruction algorithm can handle 3D solids of over 10,000 faces efficiently, outperforming any other existing method. Our geometric reconstruction algorithm can lead to amazing simplification in symbolic manipulation of the geometric data, and can be used to find linear construction sequences for non-spherical polyhedra.

Keywords: Polyhedra, Structural Reconstruction, Geometric Reconstruction, Grassmann-Cayley Algebra, Local Propagation.

1. Introduction

Representing and perceiving an nD object has been a very fascinating problem in both science and art [18]. For $n = 3$, the simplest representation is a line drawing which is the 2D projection of the wireframe of the object, like drafting in geometric design and mathematical diagram. To perceive an nD object one needs to rebuild the nD structure from its 2D projection.

How can the n dimensions be recovered from a representation in which almost all dimensions are lost? To start with, let us analyze how a solid in 3D space is perceived. No one can direct his eyesight to pierce through the solid. The only perceived object is the boundary of the solid, which is a 2D *closed manifold*. It is the closedness that allows us to fill the boundary with solid content to achieve one more dimension. When we watch a line drawing of the wireframe of a solid, which is essentially one dimensional, we extract each cycle of edges, either fill it by a plane or by some other surface to improve its dimension by one. Then we detect if any closed manifold is formed by the planes and surfaces, and if so, gain one more dimension by filling the closed manifold with solid content. The closedness of a manifold and a pattern to fill it are the two essential things in our 3D perception from low dimensional data.

From the topological point of view, a closed manifold is a *closed chain* in homology, and the boundary of a non-closed manifold is a *boundary chain*. While all boundary chains are closed, the converse is not true. For any dimension r, the quotient of the closed r-chains over the boundary r-chains is called the *r-th homology* of the object. Determining the boundary chains from the closed ones is equivalent to determining the homology.

The *wireframe model* of an nD object consists of (1) a set of *edges* connecting a finite set of points, called *vertices* of the object, (2) a subset of closed r-chains for $0 < r < n$, called *boundary r-chains*, which are the boundaries of the $(r+1)$D pieces of the object, (3) a set of filling patterns, each for a boundary r-chain.

Example 1. Line drawings of a tetrahedron and a torus (also a 3D sphere).

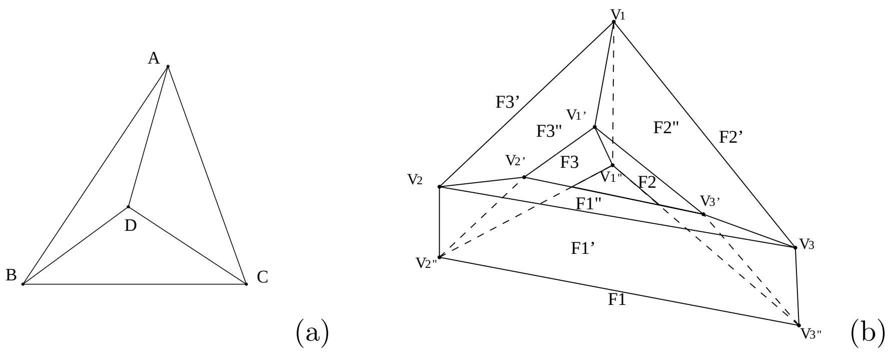

Fig. 1. (a): a tetrahedron; (b): a torus in 3D space (also a 3D sphere in 4D space).

Figure 1(a) shows a triangle being decomposed into three smaller ones. When all the four triangles are interpreted as polyhedral faces in 3D, the four faces form the boundary of a tetrahedron.

Figure 1(b) has 6 triangular cycles and 9 square cycles of edges. If all the 15 cycles are interpreted as polyhedral faces, then the 2D faces form 6 cycles of faces which are the boundaries of 6 triangular columns. If the 6 cycles of faces are interpreted as triangular columns then they form a cycle of 3D faces, which is the boundary of a 4D ball, i.e., the cycle of 3D faces forms a 3D sphere. If the object is required to be a 2D manifold, then the line drawing has a unique interpretation: it represents a torus in which the 6 triangular cycles are holes instead of boundaries of filled faces.

For several decades, the study of wireframe representation and reconstruction has been focused on the real-world case $n = 3$. The reconstruction consists of two steps: structural reconstruction (i.e., face identification) and geometric reconstruction (i.e., realizability and parametrization). Structural reconstruction is to find those closed cycles which are boundaries of faces of a prescribed filling pattern. Geometric reconstruction is to determine whether a 2D line drawing of an nD scene is realizable in the space, and if so, give a parametrization of the space of all possible realizations.

The reconstruction can be based on either one or several projections of the same object. The projection can be either perspective or parallel. The filling patterns can be either transparent or opaque, either curved or polyhedral. In this paper, we focus on reconstructing an nD polyhedral wireframe in an mD ($m \geq n$) affine space from a single 2D line drawing, by assuming that the fillings be *completely transparent*, so that all the edges and vertices are visible, and that no two vertices are projected onto the same spot.

The structural reconstruction of an 3D object has been an active research topic in computer-aided design [1], [6], computer graphics [3], [7], [27] and computer vision [9], [17], but the case is completely open for $n > 3$. In the literature, all algorithms for 3D reconstruction consist of two steps: searching for all the cycles in the wireframe which are face candidates, and then identifying faces from the candidates. In searching for cycles, all the algorithms are *global* in that the searching is within the whole wireframe. A consequence is that the number of face candidates found is usually much larger than the number of real faces. Since identifying faces from the candidates is an NP-complete problem, the fastest algorithm in the literature can only handle wireframes of about 30 faces [15].

In the first part of this paper, we extend the study of polyhedral structural reconstruction from 3D to nD, under the most general assumption that neither the dimension n of the object nor the dimension m of its surrounding space is given, and whether or not the object is a manifold is unknown. This study is at least valuable in scientific visualization and high dimensional animation in entertainment industry: scientists and artists may be very much excited to find that their conceptual and spiritual nD object can be readily embodied in and perceivable from a single 2D line drawing.

We then propose and implement a very efficient structural reconstruction algorithm, which is valid for any n and m, no matter if they are known or not. In the classical case $m = n = 3$, our algorithm outperforms all other algorithms for face identification in both speed and range of application. For all the examples in [14], [15], [21], our algorithm can generate all the solutions for ambiguous wireframes, and do not produce or produce much fewer redundant cycles which are not real faces. Surprisingly, our algorithm can handle complicated 3D objects of over 10,000 faces [13].

There are several key ideas in our algorithm:

(1) For a general object in an unknown environment, although any compatible face identification is a solution to the reconstruction problem, the goal should be to find the most plausible solution that would be identified by a human observer. A human tends to choose a face identification in which there are as many edges as possible participating in as many faces as possible, which is the guideline for the algorithms in [14] and [21]. For nD face identification, the most important goal should be to find the highest dimension n, and for this purpose the above guideline may not prove to be helpful. In this paper, we propose a *new guideline* catering to this goal.

(2) To improve the speed for finding the first acceptable solution, it is very important to arrange the face candidates in such an order that the most plausible ones come first. We classify the cycles according to their *rigidity* so that they have different *levels of priority* in face identification.

(3) Initially to speed up the finding of the highest dimension, we propose to search for the cycles *locally* in the wireframe, then propagate the local wireframe to construct more cycles. This localization technique proves to be very efficient also in reducing the number of redundant cycles.

(4) While all other methods do not allow neighboring faces to be coplanar, in this paper we do. To further control the number of face candidates, we propose two techniques, *deletion* and *blocking*, to reduce the scope of cycle searching by deleting or blocking the branches that do not produce any new face candidates.

Once the polyhedral structure (also called *incidence structure*) of an nD polyhedral scene is determined, the next task is to determine the geometric positions of the faces in the mD surrounding space. This is the problem of *geometric reconstruction.* It is generally assumed that $m = n$ and that the nD scene is an $(n-1)$D manifold, or equivalently, an nD manifold with boundary. In this paper, we focus on the geometric reconstruction from a single 2D line drawing, by assuming that the center (or direction) of the perspective (or parallel) projection from nD to 2D is given, and that no 2D face of the polyhedron is projected into an image line. The wireframe is *realizable* if there exists an $(n-1)$D manifold whose projection is the line drawing. The equalities that must be satisfied by the 2D coordinates for the wireframe to be realizable are called the *realizability conditions.* The geometric positions of the faces cannot be unique, but can be *parametrized by free parameters.* Thus geometric reconstruction is also referred to *realizability and parametrization.*

Example 2. Truncated pyramid.

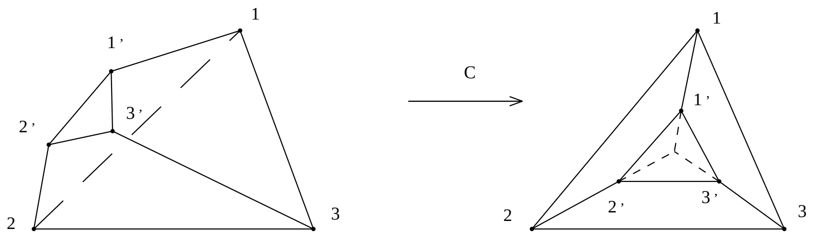

Fig. 2. 3D truncated pyramid and its 2D projection.

This is a classical example of 3D geometric reconstruction [5]. In Figure 2 (left) there are 2 triangular faces and 3 square ones. It represents a truncated pyramid formed by the 5 faces. However, the truncated pyramid can be realized in 3D if and only if the three image lines in Figure 2 (right) concur, otherwise it can only lie in a spatial plane, i.e., is flat.

In the literature of structural reconstruction, all attention is focused on the real-world case $n = 3$, and in most cases, on the realizability problem. Being a classical problem in both computer vision and combinatorial geometry, it has been studied from both numerical computation and symbolic computation aspects. In a series of papers [23], [24], [25], Sugihara formulated the problem as a set of equations called *fundamental equations*, and proved that the realizability is equivalent to the existence of solutions for a linear programming problem. The problem is much more difficult to solve if the geometric data are in symbolic form, and has been an active research topic in combinatorial geometry for over 20 years [5], [22], [30], [31]. In [5], Crapo proposed to reformulate the problem as a set of equations called *syzygy equations*, which is much smaller in number. Still the equations are very difficult to solve symbolically.

In the second part of this paper, we extend the study of the geometric reconstruction of polyhedra from 3D to nD. We propose a simple and effective method to solve simultaneously the realizability and parametrization problems symbolically. The method can be used to find *linear construction sequences* for the 3D geometric configurations of a class of polyhedra with nonzero genus, a result that has never been obtained before [12].

There are three key ideas in our algorithm:

(5) Using Grassmann-Cayley algebra, we decompose the fundamental equations into two

systems: the *system of extensors* and the *system of heights.* We only need to solve the former system, for which we can employ the powerful *vectorial equation-solving technique* [10].

(6) In solving the equations of extensors, we *introduce new parameters locally*, propagate the local solutions to obtain more parametrized ones. Essentially this is a procedure of transforming the system of extensors into a *system of new parameters.*

(7) To simplify the system of new parameters, algebraic factorization is necessary but very difficult. For $n = 3$, we propose an efficient method called *calotte factorization*, to factor the system of new parameters *geometrically.*

This paper is arranged as follows. In Section 2 we analyze some properties of a general nD wireframe object and its 2D line drawing, and lay down the foundation of high-dimensional reconstruction. In Section 3 and 4 we propose some powerful techniques for nD structural and geometric reconstructions.

2. nD Wireframe Object and 2D Line Drawing

We consider the wireframe models in which the filling patterns are affine flats. This pattern has a strong constraint, called the rD *face adjacency constraint* [21]: if two rD faces share two $(r-1)$D faces which are in different $(r-1)$D affine planes, then the two rD faces must be in the same rD affine plane, i.e., are *coplanar.*

We assume that the 2D line drawing of an nD wireframe object is obtained by a perspective or parallel projection from an mD affine space to the image plane, such that all the edges and vertices are revealed, and if three vertices are not collinear in the mD space, nor should their images. So in the line drawing,

- if two edges are collinear, so are they in the mD scene;
- if two edges cross not at a vertex, they do not intersect in the mD scene.

2.1. nD Perspective and Parallel Projections

A *perspective projection* from mD to 2D is the composition of $m-2$ successive perspective projections whose projective centers are linearly independent vectors, and at least one center is an affine point. A *parallel projection* from mD to 2D is the composition of $m-2$ successive parallel projections whose projective centers are linearly independent directions.

As is well known, both projective and affine geometries can be efficiently represented by Grassmann-Cayley algebra and bracket algebra [22] in a coordinate-free manner. Below we give a brief introduction of this algebra.

Let $\mathcal{V}^{m+1}$ be an $(m+1)$D vector space representing the mD affine space. The *outer product* is the unique associative, multilinear and anticommutative product defined among the vectors of $\mathcal{V}^{m+1}$. The outer product of r vectors is called an *r-extensor.* Any linear combination of r-extensors is called an *r-vector.* The *Grassmann space* $G(\mathcal{V}^{m+1})$ generated by $\mathcal{V}^{m+1}$ is the graded vector space of r-vectors for r from 0 to n. The *Grassmann-Cayley algebra* of $G(\mathcal{V}^{m+1})$ is the Grassmann space together with two products among its elements: the outer product, denoted by juxtaposition, and the *meet product*, denoted by "$\wedge$", which is the dual of the outer product.

The space of $(m+1)$D-vectors is one dimensional. Let I be a basis of this space, then the *bracket* of an $(m+1)$-vector x with respect to I is its coordinate

$$[x] = x/I. \tag{2.1}$$

Bracket algebra refers to the ring of brackets.

In Grassmann-Cayley algebra, an r-extensor represents the $(r-1)$D projective space spanned by the r vectors. So in the mD projective space, a point is represented by a nonzero vector, which is unique up to scale. The line passing through points $\mathbf{1}, \mathbf{2}$ is represented by *bivector* $\mathbf{12}$. The rD plane passing through points $\mathbf{C}_1, \ldots, \mathbf{C}_{r+1}$ is represented by the r-extensor $\mathbf{C}_1\mathbf{C}_2 \cdots \mathbf{C}_{r+1}$. The intersection of an rD plane A and an sD plane B, if $r+s > m+1$, is represented by the meet $A \wedge B$. A perspective (or parallel) projection from mD to 2D is represented by [11]

$$\mathbf{P} \mapsto X\mathbf{P}, \quad \text{for all } \mathbf{P} \in \mathcal{R}^{m+1} \tag{2.2}$$

where X is the $(m-2)$-extensor representing the center (or direction) of the projection. The space $\{X\mathbf{P} \,|\, \mathbf{P} \in \mathcal{R}^{m+1}\}$ is a 3D vector space representing the image plane. Any decomposition of X into the outer product of $m-2$ vectors induces a decomposition of the projection into $m-2$ "classical" projections in which the perspective center is a point or point at infinity.

2.2. Constraints for Structural Reconstruction

Structural reconstruction is to recover for r from 2 up to n the rD polyhedral structure, i.e., incidence structure, i.e., boundary $(r-1)$-chains. Since it is seldom possible to judge if a single cycle should be a boundary chain, in general we need to find a number of cycles which are *face candidates*, and then select from them a subset satisfying the properties of real faces. Structural reconstruction is also called *face identification*.

In a wireframe model, a 0D *face* is a vertex, a 0D *cycle* is the two vertices of an edge, and a 1D *face* is an edge. For $r > 0$, an rD *cycle* is a set of rD faces such that (1) if two faces intersect, the intersection belongs to their iD faces for $0 \leq i < r$, (2) any $(r-1)$D face of one rD face is shared by exactly one other rD face in the set. For $r > 1$, an rD *face* is an $(r-1)$D cycle filled by the rD affine flat surrounded by it.

Some geometric constraints must be satisfied for an $(r-1)$D cycle to become an rD face. In previous work on face identification, it is generally assumed that any two neighboring faces are not coplanar. We feel that this is too strong a constraint to include many interesting models, so we discard it.

If two rD faces share at least two $(r-1)$D faces which are in different $(r-1)$D planes, then the two rD faces must be in the same rD affine plane, i.e., *coplanar*. The union of a set of rD coplanar faces is called an rD *polyface*, and the faces are said to be *merged* together. An advantage of this concept is that usually we do not need to decompose a polyface into non-overlapping faces.

By a perspective projection, a 2D face is projected onto a 2D region of the image plane whose boundaries or boundary do not intersect. The 2D *non-self-intersection constraint* [9] says that if two edges intersect not at a vertex, then they cannot be in the same 1D cycle.

If there are at least three vertices which are collinear in the mD space, then the 2D projection of the line passing through the vertices is called a *line* in the line drawing. The edges within a line are the *real parts*, and the virtual connections between real parts are the *virtual parts*. The 2D *non-interior-intersection constraint* [15] says that if two 1D cycles intersect at only two vertices and the line segment between the two vertices intersects both the enclosed regions of the 1D cycles in the image plane, then either the two cycles form a polyface, i.e., are coplanar, or at most one can be assigned as a face. This is because if both cycles are faces and are non-coplanar, then the virtual part between the two vertices must be the projection of a real part of the line of intersection, contradicting the assumption that all edges are visible by the projection.

In a 2D line drawing it is impossible to distinguish between the interior and the exterior of an rD object for $r > 2$. So the 2D non-interior-intersection constraint has no high dimensional generalization.

If an rD face F is not in an rD cycle C but its intersection with C is exactly the boundary of F, then F is called a *chord* of C. A face with chord can always be decomposed along its chord into two coplanar faces. On the other hand, if a cycle with chord is not assigned as a face, then it can be decomposed into two cycles by adding the chord into it, and by assigning both cycles to be faces they do not need to be coplanar. To gain more degree of freedom in the reconstruction, an rD cycle with chord is not assigned as a face. This is the rD *chordless constraint*.

Let F be an sD face for $s \geq 0$. Any rD face containing F is called an rD *F-face*. When $r > s$, an rD *F-cycle* refers to a set of rD F-faces in which any $(r-1)$D F-face of one rD F-face is shared by exactly one other rD F-face, and not all rD F-faces are in the same rD plane.

In many applications it is required that the nD polyhedral object be a manifold with boundary, i.e., the boundary is an $(n-1)$D *manifold*. The formal definition is that at every point of the boundary the intersection of the boundary with a sufficiently small mD disk centered at the point is homeomorphic to an $(n-1)$D disk. In general, the following constraint, called $(n-1)$D *manifold constraint*, suffices to guarantee that an $(n-1)$D cycle C satisfying all previous constraints in this subsection be an $(n-1)$D manifold: at any vertex $\mathbf{V}$ of the cycle, there is at most one $(n-1)$D $\mathbf{V}$-cycle in C. This constraint has never occurred in the literature of face identification before.

2.3. mD Geometric Reconstruction

Let the perspective or parallel projection from mD to 2D be induced by the center $X = \mathbf{C}_3\mathbf{C}_4 \cdots \mathbf{C}_m$, where the $\mathbf{C}$'s are vectors in the $(m+1)$D vector space of homogeneous coordinates. Each $\mathbf{C}$ represents a point or direction (point at infinity). The geometric reconstruction from 2D to mD can be realized step by step from $(r-1)$D to rD, for r from 3 to m. The lift from $(r-1)$D to rD is the reverse procedure of the projective from rD to $(r-1)$D centered at $\mathbf{C}_r$. In this subsection we always assume that $r \geq 3$.

Assume that the line drawing represents an mD manifold with boundary. Further assume that both $\mathbf{C}_r$ and the $(r-1)$D image plane $\mathcal{I}_{r-1}$ are given and are *in generic positions* in the rD affine space $\mathcal{I}_r$. In fact,

$$\mathcal{I}_r = \mathcal{I}_{r-1}\mathbf{C}_r = \mathcal{I}_{r-2}\mathbf{C}_{r-1}\mathbf{C}_r = \cdots = \mathcal{I}_2\mathbf{C}_3 \cdots \mathbf{C}_r. \tag{2.3}$$

The input of the lift from $(r-1)$D to rD is (1) the $(r-1)$D coordinates of the image vertices, (2) the incidence structure between the $(r-2)$D and $(r-1)$D faces. The output is (1) the realizability conditions of the *rD faces* satisfied by the the *2D coordinates* of the image vertices, (2) the parametrized solution space of the vertices and $(r-1)$D faces in $\mathcal{I}_r$. Since the perspective center is generic, the equality constraints on the $(r-1)$D coordinates are reduced to equalities on the 2D coordinates of the vertices.

Let $\mathcal{V}^{r+1}$ be the homogeneous coordinate space of $\mathcal{I}_r$. Let $\{\mathbf{C}_0, \mathbf{C}_1, \mathbf{C}_2\}$ be a basis of $\mathcal{V}^3$ such that $\mathbf{C}_1, \mathbf{C}_2$ are directions of the image plane $\mathcal{I}_2$. Then $\{\mathbf{C}_0, \mathbf{C}_1, \ldots, \mathbf{C}_r\}$ is a basis of $\mathcal{V}^{r+1}$. Let the homogeneous coordinates of a vertex $\mathbf{V}^r$ in $\mathcal{I}_r$ be $(h^0, h^1, \ldots, h^r)$, where h^r is called the *height* of the vertex with respect to $\mathbf{C}_r$. Let the homogeneous coordinates of an $(r-1)$D face F^{r-1} in $\mathcal{I}_r$ be $(a_0, a_1, \ldots, a_{r-1}, -1)$. Then vertex $\mathbf{V}^r$ is in face F^r if and only if

$$\sum_{i=0}^{r-1} a_i h^i = h^r. \tag{2.4}$$

When the vertices and $(r-1)$D faces range over the whole $(r-1)$D incidence structure, we obtain a set of linear homogeneous equations in the a's and h^r's, called *Sugihara's rD fundamental equations.* Solving (2.4) for the realizability conditions if the h_i take symbolic values for $1 \le i < r$, is a very difficult task.

For any basis $\{\mathbf{C}_i \,|\, 1 \le i \le r+1\}$ of $\mathcal{V}^{r+1}$, its *dual basis* in the Grassmann space generated by $\mathcal{V}^{r+1}$ is the r-extensors $\{\mathbf{C}_j^* \,|\, 1 \le j \le r+1\}$ such that $[\mathbf{C}_i \mathbf{C}_j^*] = \delta_{ij}$. In fact,

$$\mathbf{C}_j^* = (-1)^{j-1} \mathbf{C}_1 \mathbf{C}_2 \cdots \check{\mathbf{C}}_j \cdots \mathbf{C}_{r+1}, \tag{2.5}$$

where $\check{\mathbf{C}}_j$ denotes that $\mathbf{C}_j$ does not occur in the product.

The $(r-1)$-extensor

$$\mathbf{B}^{r-1} = \sum_{i=0}^{r-1} a_i \mathbf{C}_i^* \tag{2.6}$$

is called the *inhomogeneous coordinates* of face F^{r-1}. Two faces are coplanar if and only if their inhomogeneous coordinates are identical. Now (2.4) can be written as an equality in the bracket algebra generated by $\mathcal{I}_{r-1}$:

$$h^r = [\mathbf{V}^{r-1} \mathbf{B}^{r-1}], \tag{2.7}$$

where $\mathbf{V}^{r-1} = \sum_{i=0}^{r-1} h^i \mathbf{C}_i$ is the projection of $\mathbf{V}^r$ in $\mathcal{I}_{r-1}$.

For any $(r+1)$-tuple of vertices $\mathbf{V}_1^r, \mathbf{V}_2^r, \ldots, \mathbf{V}_{r+1}^r$ in face F^{r-1}, their projections $\mathbf{V}_1^{r-1}, \mathbf{V}_2^{r-1}, \ldots, \mathbf{V}_{r+1}^{r-1}$ in the image plane $\mathcal{I}_{r-1}$ satisfy the following *Grassmann-Plücker relation* [22]:

$$\sum_{i=1}^{r+1} (-1)^{i-1} [\mathbf{V}_i^{r-1} \mathbf{B}^{r-1}][\mathbf{V}_1^{r-1} \mathbf{V}_2^{r-1} \cdots \check{\mathbf{V}}_i^{r-1} \cdots \mathbf{V}_{r+1}^{r-1}] = 0. \tag{2.8}$$

When F^{r-1} ranges over all $(r-1)$D faces with more than r vertices, we obtain a set of equations linear in the h^r's by substituting (2.7) into (2.8):

$$\sum_{i=1}^{r+1} (-1)^{i-1} h_i^r [\mathbf{V}_1^{r-1} \mathbf{V}_2^{r-1} \cdots \check{\mathbf{V}}_i^{r-1} \cdots \mathbf{V}_{r+1}^{r-1}] = 0. \tag{2.9}$$

They are called *Crapo's rD syzygy equations.*

On one hand, the fundamental equations and the syzygy equations have the same solutions for the h^r's. On the other hand, the number of syzygy equations is much smaller than the number of fundamental equations. The syzygy equations can also be derived by eliminating the $\mathbf{B}$'s from the fundamental equations (2.7).

Notice that for $r > 3$, both systems have a lot of redundant equations. Since an $(r-1)$D face F^{r-1} is in an rD face F^r if and only if there is an r-tuple of vertices of F^{r-1} incident to F^r but not incident to any $(r-2)$D affine plane, in the case that F^{r-1} has more than r vertices, the syzygy equations of its $(r+1)$-tuples of vertices are all equivalent to each other. To remove the redundancy, we require the incidence structure between the $(r-2)$D and $(r-1)$D faces be in the input of rD lift, instead of only the incidence structure between the vertices and $(r-1)$D faces.

2.4. Rigidity

By the definition of a cycle, the supporting affine plane of an $(r-1)$D cycle has dimension at least r. By the geometric reconstruction from rD to $(r+1)$D, for a given $(r-1)$D cycle in the rD affine plane, if the dimension of the configuration space of the lifted cycle (which is defined to be the maximal number of free parameters in the parametrization) is $k+r+1$, then the *rigidity* of the cycle is defined to be $-k$, and the *flexibility* of the cycle is defined to be k. A cycle of rigidity 0, or -1, or < -1 is said to be *rigid*, or *elastic*, or *plastic* respectively.

For example if $r = 2$, a 1D cycle of $k+3$ vertices has rigidity $-k$, because when lifting from 2D to 3D along the perspective center, the cycle has $k+3$ free parameters which are exactly the heights of all the vertices. Thus rigid 1D cycles are triangular and elastic 1D cycles are square ones.

The *principle of rigidity* in structural reconstruction states that (1) rigid cycles are always identified as faces, (2) elastic cycles are more likely to be faces than plastic ones. The explanation is as follows:

1. Rigid cycles are natural object delimiters in the wireframe, they are either real or interior faces of the object. If they are assigned as faces, they never force any two faces of different planes to be coplanar, i.e., they do not cause any geometric incompatibility.

2. If the object is not assumed to be a manifold, then taking all rigid cycles as real faces conforms to the *principle of psychological selection* to be introduced in Section 3., that more faces can be produced which pass through more lower dimensional ones and do not reduce the number of higher dimensional ones.

3. If the object is required to be a manifold, then taking all rigid cycles as real faces induce a decomposition of the object into smaller ones of the same dimension, and by the manifold assembly algorithm to be introduced in Section 3., all interior faces can be removed.

4. Elastic cycles are next to rigid ones in simplicity. Experiments show that they are the next most plausible face candidates.

On one hand, the rigidity of a general cycle can be determined only by geometric reconstruction. On the other hand, the rigidity is to be employed in structural reconstruction, far before geometric reconstruction starts. This paradox can be resolved as follows: The principle of rigidity serves only as a heuristic rule in ordering the face candidates. It is not a prerequisite that all cycles be ordered strictly by their rigidity in face identification. In practice we use the following algorithm to search for a class of rD rigid cycles for $r > 1$, called *rooted rigid cycles*:

Two non-coplanar rD faces sharing one $(r-1)$D face are called *neighbors.* For every pair of rD neighbors, do the following:

(1) Initially let set C contains only the pair.

(2) Repeatedly put into C all the rD faces and polyfaces sharing with C at least $r+1$ vertices which are not in the same $(r-1)$D plane.

(3) If C is an rD cycle then it is rigid.

Example 3. For a 2D cycle, if at each vertex there are at most three edges, then the cycle must be rooted rigid. Figure 1(a) and Figure 2 are such examples.

3. Structural Reconstruction

Even for the classical case $m = n = 3$, structural reconstruction is very difficult. The number of cycles in a graph is generally exponential in the number of vertices. Given that all the cycles which are potential faces are found, the number of their combinations is exponential in the number of the cycles. So this is a problem of double-exponential complexity even for $m = n = 3$.

The *guideline* in designing an efficient algorithm should not be to find a polynomial-time algorithm for all optimal solutions. We propose that it should be to *reduce the time for finding the first optimal solution.*

The what is an "optimal solution"? For different purposes there are different criteria. We propose the following *sequence of criteria* according to their precedence in our general-purpose algorithm:

1. The highest dimension n of the object should be found and reached.
2. The solution should be most likely identified by a human.

   Our *principle of psychological selection* is that for $r > 1$, the rD face identification should make as many $(r-1)$D faces as possible participating in as many rD faces as possible, such that the sequence of numbers of non-coplanar iD faces is maximal lexicographically for i from n down to r.
3. If the solution is required to be a manifold of a given dimension, then any such solution is optimal.
4. If there are several optimal solutions, then as many of them as possible should be found.

Below we propose six powerful techniques for structural reconstruction based on the above guideline and criteria. Without loss of generality, we only describe the classical case of finding 1D cycles for 2D face identification. By the following correspondences, the techniques and concepts therein (except for Subsection 3.5.) can be readily generalized to rD cycle-finding for $(r+1)$D face identification, for $r > 1$:

$$\begin{aligned} \text{vertex (0D face)} &\longrightarrow && (r-1)\text{D face}, \\ \text{edge (1D face)} &\longrightarrow && r\text{D face}, \\ \text{cycle (1D cycle)} &\longrightarrow && r\text{D cycle}, \\ \text{face (2D face)} &\longrightarrow && (r+1)\text{D face}. \end{aligned}$$

3.1. Localization

It is well recognized that to search for the real faces one does not need to find all cycles. We further recognized that to speed up the finding of the first optimal solution ones can simply search for face candidates *locally.*

Let C be a wireframe model. A *local wireframe model* of C is a subset of the vertices of C together with all the higher dimensional faces formed by the vertices. A *localization filter*, or *localization*, of C is a sequence of local wireframe models $S_1 \subset S_2 \subset \ldots \subset S_k = C$ in which each successor introduces more vertices than its predecessor. With the introduction of new vertices, all the edges among them and the existing vertices are introduced.

Localization is often realized by *propagation through edges.* Starting from a vertex called the *origin*, we localize the wireframe by considering only the subgraph of the origin and its neighboring vertices. Within the local wireframe we identify the faces. Then we set the origin to be the current local wireframe, and repeat the localization and identification procedure. By introducing new vertices according to the closeness of their relations with the existing ones, the complexity of cycle searching can be reduced. Below are some typical localizations.

The first is the *descendent localization*: (1) Start from a set C containing only one vertex called the *origin*, put all the neighboring vertices of C into C. (2) Put all the vertices collinear with at least two vertices of C into C. Repeat until C no longer changes. (3) Put all the neighboring vertices of C into C. (4) Repeat Steps 2 and 3 until all vertices are in C. Each repeat is a round of localization.

The second is the *singleton localization*: Steps 1, 2, 4 are the same with those in descendent localization. Step 3 is as follows: (3) Put into C those vertices called *singletons*, which are common neighbors of at least two vertices of C, and if there is no singleton at all then put into C its neighboring vertices, called *twins.* This localization makes much easier the generation of new cycles.

The third is the *rigidity localization*: Steps 1, 2, 4 are the same with those in the neighbor localization. Step 3 is as follows: (3) Put into C those singletons each forming a rigid or elastic cycle with some vertices in C, and if there is no such vertex at all then use Step 3 of the singleton localization. Locally, rigid cycles are constructed first, followed by elastic and plastic ones. This localization proves to be the most efficient.

Example 4. In Figure 3(a) there is no collinearity constraint prescribed. The three localizations are identical:

$$\{1\} \subset \{1,2,3,4\} \subset \{1,2,3,4,5,6,7,8,9,0\} \subset \{1,2,3,4,5,6,7,8,9,0,a,b,c\},$$

which proceeds till all vertices are included.

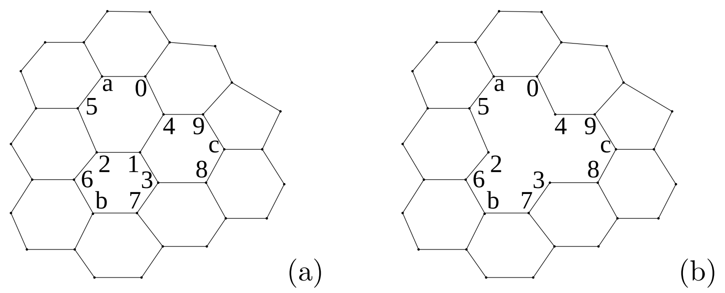

Fig. 3. Localization and deletion.

3.2. Deletion

Two techniques accompany each round of localization: deletion and path blocking. The purpose of deletion is to reduce the size of a local wireframe by cutting off those edges and vertices which no longer contribute to face identification.

When $n = 3$, the idea of deleting edges in cycle searching can be found in [15]. There the object is a 2D manifold in which any edge occurs in exactly two faces, so if two faces containing the edge have been found, the edge can be deleted. At any time, those vertices through which there is only one edge can always be deleted.

For a general object, the criterion for an edge to be deletable should be that *the deletion does not influence the final identification of 2D faces and polyfaces*. By our principle of psychological selection, an edge E in the procedure of localization can be deleted if by assigning any new cycle through it as a face, (1) the number of non-coplanar faces does not increase, (2) if any new vertex $\mathbf{V}$ is to be added into the polyface containing the face, there is always a cycle of edges in the polyface that passes through $\mathbf{V}$ but not E, i.e., deleting E does not prevent $\mathbf{V}$ from joining the polyface via a cycle of edges.

Deletion Theorem. *In cycle searching, any saturated edge outside lines, and any saturated real part of a line, can be deleted.*

We shall explain the term "saturated" but omit the proof of the theorem. An edge is said to be *saturated* if all the edges protruding from its vertices are already in faces passing through the edge. A connected real part of a line is said to be *saturated* if every edge of it is saturated.

In Figure 3(a), if three cycles $125a04$, $138c94$ and $126b73$ are already assigned as faces, then edges $12, 13, 14$ are saturated and can be deleted. Then vertex 1 is no longer connected to any other vertex and can be deleted. See Figure 3(b).

3.3. Path Blocking

In Figure 3(b), the vertices in the localization form a big cycle $25a049c837b6$. This cycle cannot be a face, otherwise all three constructed faces have to be merged. In searching for more face candidates passing through a fixed vertex, those faces having been identified can block off some search branches by preventing identified faces from merging, according to our principle of psychological selection. This technique is extremely useful in reducing the number of branches in cycle searching.

For a general object, if a branch of edges intersects a face in at least three vertices which are not collinear, then the branch is blocked by the face. The block is called a *face block.* If a branch meets two different face blocks, then it is blocked permanently. If along a branch there is only one face block, then the branch of edges can be merged with the face to form a polyface.

In Figure 3(b), suppose we want to find a new cycle passing through vertex 2. From 2 to 6, the path is blocked by face $62137b$. From 2 to 5, the path is blocked by face $04125a$. The two different face blocks permanently block any new cycle from passing through branch 625.

In [15], the models are 2D manifolds in which no two neighboring faces are coplanar. If a cycle is identified as a face then no other branch passing through two edges of it can generate a face. Here one face suffices to block off the branch permanently. For a general object, this blocking does not work.

There are other types of blocks. The 2D non-self-intersection constraint can block some branches permanently. These blocks are called *intersection blocks.* The 2D non-interior-intersection constraint can block some branches from including the interior of an existing face. These blocks are called *interior blocks.* The 2D chordless constraint can permanently block some branches from generating cycles with chords. These blocks are called *chord blocks.*

3.4. Local Optimization

Assume that all the constructed cycles satisfy the constraints in Subsection 2.2.. For a group of cycles, if when assigning all of them to be new faces, either two of them merge, or one face merges with an existing face, then the group of cycles is said to be *degenerate.* For a degenerate group of cycles, by the principle of psychological selection, generally not all the cycles are assigned as faces.

An optimal selection of a subset of cycles within the group should maximize the number of non-coplanar faces, or equivalently, minimize the number of merges. If there are several optimal selections, then the algorithm bifurcates and the *state-space tree* [15] is generated, or extended if it has been generated. Each optimal selection is to be explored to find all solutions.

In rigidity localization, the above *local optimization* can be greatly simplified. Recall that in each round of localization, there are two kinds of new vertices: singletons and twins. In face identification, elastic cycles have priority over plastic cycles, and singletons have priority over twins. Thus, there are four *levels of priority* for non-rigid cycles. Our strategy is to restrict the local optimization to cycles of the same level of priority. The details can be read from the following example.

Example 5. A diagonal is drawn in a cube (Figure 4), which makes the face identification very difficult to reach dimension three [21]. By the local optimization within rigidity localization, the unique 3D explanation can be easily obtained.

The following new vertex series is produced by rigidity localization:

$$1 \longrightarrow 2,3,4,5 \longrightarrow 6,7,8. \tag{3.1}$$

Only in the last step do the cycles appear. The singletons generated by elastic cycles are

$$6:\{1264, 1364\}; \quad 7:\{1275, 1273, 1375\}; \quad 8:\{1284, 1485\}. \tag{3.2}$$

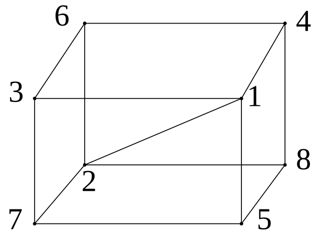

Fig. 4. Local optimization.

The other elastic cycles are 2637, 2648, 2758. For each singleton, its generating elastic cycles are degenerate. The 3 groups of cycles form 12 combinations, among which only the combination (6 : 1364, 7 : 1375, 8 : 1485) is optimal. No state-space tree is generated.

3.5. Repair and Assembly

Example 6. In Figure 5, there is a line 1563 with virtual part 56. In the literature, usually two solutions are found, one with faces 123675 and 415863, the other with faces 123685 and 415763. The latter is geometrically impossible, because the 2D non-interior-intersection constraint is violated.

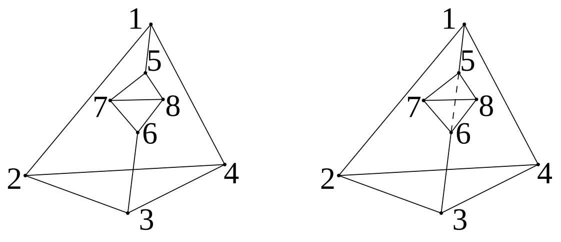

Fig. 5. Repair and assembly.

We introduce a simple technique called *repair and assembly*, to avoid producing the fake solution. It is also very helpful in simplifying cycle searching, although it only applies to 2D face identification. The technique is composed of three steps:

(1) If there is any virtual part of a line, *repair* the line drawing by connecting the virtual part. The new edges are said to be *virtual*.

(2) Make 2D face identification in the repaired line drawing. In Figure 5, two tetrahedra are obtained.

(3) The *second assembly principle* states that if two 3D faces intersect at an edge E but not at any face, then they can be assembled at the edge if and only if within the line drawing, a face F_1 at E in one 3D face is within the 2D region of a face F_2 at E in the other 3D face. If there are no two such faces then one 3D face must be deleted; else the assembling is realized by replacing the two faces by their difference in the polyface generated by them.

Assemble the faces containing the virtual edges according to this principle.

(4) Remove the virtual edges to recover the original wireframe.

3.6. Manifold Assembly

In many applications it is required that the object be a 2D manifold. For this special purpose, there are two alternatives to revise our general-purposed structural reconstruction algorithm:

Anterior Approach: Employ the apriori constraints of a 2D manifold in the general algorithm from the start, by revising the deletion, path blocking and local optimization techniques, similar to the algorithm in [15].

Posterior Approach: Neglect the given information and use the general algorithm to produce a set of 3D faces which are themselves 2D manifolds. Assemble the 3D faces into a single 2D manifold. This approach, called *manifold assembly*, appears to be more efficient than the previous one.

Example 7. Figure 6(a) shows a torus in 3D space. Without the requirement that the output be a 2D manifold, it will be shown in the next subsection that our structural reconstruction algorithm produces a 4D-face explanation in which the boundary is composed of 6 triangular columns (3D faces), with side faces $F_1F_{1'}F_{1''}$, $F_2F_{2'}F_{2''}$, $F_3F_{3'}F_{3''}$, $F_1F_2F_3$, $F_{1'}F_{2'}F_{3'}$, $F_{1''}F_{2''}F_{3''}$ respectively. Denote the columns by their side faces. We need to assemble them to obtain the torus. Below we use this example to explain our manifold assembly technique.

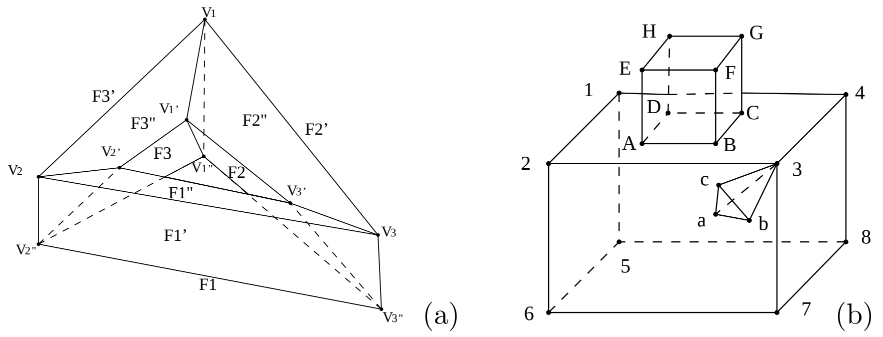

Fig. 6. 2D manifold assembly.

The input is a set of 3D faces whose boundaries are 2D manifolds, denoted by $\mathcal{S}$. The output is a single 2D manifold. There are several steps:

(1) Compute the pairwise intersections of the 3D faces. Divide $\mathcal{S}$ into different connected components. In each component $\mathcal{C}$, count the *degree* of each edge in the 2D intersections, with respect to the 2D faces in $\mathcal{S}$.

For a set of rD faces, the *degree* of an $(r-1)$D face with respect to the set is the number of elements in the set passing through the $(r-1)$D face.

In Figure 6(a), the intersection between any two columns is a face. The columns form a single connected component. Each edge in the graph has degree 3. If face $F_{1''}$ is deleted, then the degree of every edge of $F_{1''}$ is reduced to 2.

(2) Now the assembling starts formally. Choose a 3D face called the *origin*, which has the maximal number of 3D neighbors in the component. Use $\mathcal{O}$ to denote the origin, and remove it from $\mathcal{C}$.

In Figure 6(a), any column has 5 neighbors. Choose any of them, say $F_1F_{1'}F_{1''}$, as the origin.

(3) The *first assembly principle* states that if two 3D faces intersect at a face whose edges are of degree greater than two, then the 3D faces can be assembled at the face by removing it; if one edge has degree two, then one 3D face must be deleted.

Use this principle repeatedly, update $\mathcal{C}$ and $\mathcal{O}$ accordingly, and move the assembled result from $\mathcal{C}$ to $\mathcal{O}$, until no 3D face in $\mathcal{C}$ has 2D intersection with $\mathcal{O}$.

In Figure 6(a), we can assemble columns $F_1F_{1'}F_{1''}$ and $F_{1''}F_{2''}F_{3''}$, and denote the result by $\mathcal{O}$. Then face $F_{1''}$ is deleted, and its neighbors $F_1, F_{1'}, F_{2''}, F_{3''}$ each have a degree-2 edge. The four neighbors disallow the other four columns to be annexed to $\mathcal{O}$. As a result, vertex $\mathbf{V}_{1''}$ is absent from $\mathcal{O}$. Although $\mathcal{O}$ itself is a 2D manifold, it does not provide the whole line drawing with such an explanation.

On the other hand, if we assemble $F_1F_{1'}F_{1''}$ and $F_2F_{2'}F_{2''}$, or start from $F_{1''}F_{2''}F_{3''}$ and assemble it with $F_{1'}F_{2'}F_{3'}$, we can get the torus explanation in two different manners.

(4) Use the second assembly principle repeatedly, update $\mathcal{C}$ and $\mathcal{O}$ accordingly, and move the assembled result from $\mathcal{C}$ to $\mathcal{O}$, until no 3D face in $\mathcal{C}$ has 1D intersection with $\mathcal{O}$.

If $\mathcal{O}$ is not a manifold and has an edge at whose endpoints $\mathbf{V}_1, \mathbf{V}_2$ the 2D manifold constraints are violated, then $\mathcal{O}$ can be assembled at the edge if and only if its $\mathbf{V}_1$-cycles of faces can be assembled at the edge by the second assembly principle. If $\mathcal{O}$ cannot be assembled at the edge then the whole assembling exits, else use the principle repeatedly until $\mathcal{O}$ no longer changes.

In Figure 5 (right), there are two tetrahedra sharing a common edge 56. The pair of faces 12365 and 567 are merged into a polyface and then replaced by face 123675. The pair of faces 15634 and 568 are merged and then replaced by face 158634. However, edge 56 is NOT deleted, although it is no longer in any face.

(5) The *third assembly principle* states that if two 3D faces intersect at a vertex $\mathbf{V}$ but not at any face or edge, then they can be assembled at the vertex if and only if within the line drawing, a face F_1 at $\mathbf{V}$ in one 3D face is within the 2D region of a face F_2 at $\mathbf{V}$ in the other 3D face. If no such two faces then one 3D face must be deleted; else the assembling is realized by replacing the pair of faces by their difference in the polyface generated by them.

Use this principle repeatedly, update $\mathcal{C}$ and $\mathcal{O}$ accordingly, and move the assembled result from $\mathcal{C}$ to $\mathcal{O}$, until $\mathcal{C}$ is empty.

If $\mathcal{O}$ is not a manifold, then if it has a vertex $\mathbf{V}$ where the 2D manifold constraint is violated, then $\mathcal{O}$ can be assembled at $\mathbf{V}$ if and only if its $\mathbf{V}$-cycles of faces can be assembled at the vertex by the third assembly principle. If $\mathcal{O}$ cannot be assembled at the vertex then the whole assembling exits, else use the principle repeatedly until $\mathcal{O}$ no longer changes.

In Figure 6(b) there are three 3D faces: two cubes and a tetrahedron. Vertex 3 is shared by the tetrahedron and a cube. Face 2376 can be merged with either face 3ab or face 3ac, leading to two different manifold structures.

(6) The *fourth assembly principle* states that if two 3D faces do not intersect, then they can be assembled at two 2D faces if and only if one 2D face is within the 2D region of the other 2D face. To assemble the two 3D faces is to replace the pair of faces by their difference in the polyface generated by them.

Now each set $\mathcal{C}$ has been replaced by the corresponding set $\mathcal{O}$. Check if each $\mathcal{O}$ is a manifold, and if not then the whole assembling exits. Use the fourth assembly principle repeatedly among the $\mathcal{O}$'s until the result is stable.

In Figure 6(b), the two cubes can be assembled at either the pair of faces $(1234, ABCD)$, or the pair $(1584, ABCD)$, leading to two different manifolds.

(7) If no manifold is constructed or more solutions are needed, then changing the order of 3D faces in the assembling sequence, including changing the origin, may lead to different results.

Remark. (a) rD assembling differs from rD merging in that in merging we do not delete anything, but in assembling we not only delete $(r-1)$D and rD faces, but also generate new $(r-1)$D and rD faces.

(b) If we employ the knowledge that Figure 6(a) represents a 3D manifold, then before the assembling we can simply delete one of the six columns, because its 3D content must be the algebraic union of the 3D contents of the other five.

3.7. The Main Algorithm

Input: (1) A 2D line drawing composed of vertices and edges. A vertex is represented by its 2D coordinates, an edge by two vertices.
(2) A set of lines. A line is represented by a sequence of collinear edges.
(3) A vertex as the origin of localization. The default is a vertex contained in a maximal number of edges.

Output: Objects of dimension > 1: faces and polyfaces.

Initialization: (1) Find all pairs of edges intersecting not at a vertex. (2) Repair lines with virtual parts.

Step 1. Localization start: Start from the origin, use rigidity localization to generate a set of new vertices.

Step 2. Cycle searching: Generate faces by the depth-first cycle searching strategy, together with the deletion, path blocking and local optimization techniques.

Step 3. Dimension upgrading: Start from the 2D local wireframe constructed so far, construct higher dimensional faces in a hierarchical order, following a procedure similar to Steps 1 to 3.

Step 4. Assembling: This occurs if there is any repair in the local wireframe, or the result is required to be a manifold. The assembling is also local.

Step 5. Localization end: Go back to Step 1 for another round of localization. Terminate after all vertices are included.

Step 6. More solutions: Explore the state-space tree or change the origin to get more solutions.

Remark. Although a specific set of 2D coordinates are given in the input, they are used only to test the inequalities occurring in the 2D non-self-intersection constraint, the 2D non-interior-intersection constraint and the 3D assembly. A solution based on these coordinates is acceptable as long as the polyhedral structures are compatible, no matter if the coordinates satisfy the realizability conditions for the geometric reconstruction.

Example 8. In Figure 6(a), the 4D interpretation is obtained as follows:

From 1D to 2D: The sequence of new vertices in the localization is:

$$\mathbf{V}_1 \text{ (origin) } \longrightarrow \mathbf{V}_2, \mathbf{V}_3, \mathbf{V}_{1'}, \mathbf{V}_{1''} \longrightarrow \mathbf{V}_{2'}, \mathbf{V}_{2''}, \mathbf{V}_{3'}, \mathbf{V}_{3''}. \tag{3.3}$$

In the first round, two rigid cycles are found:

$$G = \mathbf{V}_1\mathbf{V}_2\mathbf{V}_3, \quad G_1 = \mathbf{V}_1\mathbf{V}_{1'}\mathbf{V}_{1''},$$

no edge is deleted. In the second round, the 4 new vertices are singletons constructed by 4 elastic cycles:

$$\begin{array}{ll} \mathbf{V}_{2'} : F_{3''} = \mathbf{V}_1\mathbf{V}_2\mathbf{V}_{1'}\mathbf{V}_{2'}, & \mathbf{V}_{2''} : F_{3'} = \mathbf{V}_1\mathbf{V}_2\mathbf{V}_{1''}\mathbf{V}_{2''}, \\ \mathbf{V}_{3'} : F_{2''} = \mathbf{V}_1\mathbf{V}_3\mathbf{V}_{1'}\mathbf{V}_{3'}, & \mathbf{V}_{3''} : F_{2'} = \mathbf{V}_1\mathbf{V}_3\mathbf{V}_{1''}\mathbf{V}_{3''}. \end{array}$$

Then the 4 edges passing through $\mathbf{V}_1$ are deleted, so is $\mathbf{V}_1$ itself. The 4 new vertices also generate 4 rigid cycles and 5 elastic cycles in which they are twins:

$$\begin{array}{lll} G_2 = \mathbf{V}_2\mathbf{V}_{2'}\mathbf{V}_{2''}, & G_3 = \mathbf{V}_3\mathbf{V}_{3'}\mathbf{V}_{3''}, & G' = \mathbf{V}_{1'}\mathbf{V}_{2'}\mathbf{V}_{3'}, \\ G'' = \mathbf{V}_{1''}\mathbf{V}_{2''}\mathbf{V}_{3''}, & F_1 = \mathbf{V}_{2'}\mathbf{V}_{3'}\mathbf{V}_{2''}\mathbf{V}_{3''}, & F_2 = \mathbf{V}_{1'}\mathbf{V}_{3'}\mathbf{V}_{1''}\mathbf{V}_{3''}, \\ F_3 = \mathbf{V}_{1'}\mathbf{V}_{2'}\mathbf{V}_{1''}\mathbf{V}_{2''}, & F_{1'} = \mathbf{V}_2\mathbf{V}_3\mathbf{V}_{2''}\mathbf{V}_{3''}, & F_{1''} = \mathbf{V}_2\mathbf{V}_3\mathbf{V}_{2'}\mathbf{V}_{3'}. \end{array}$$

After this all edges are deleted, so no more cycles are constructed.

From 2D to 3D: It happens in every local wireframe of (3.3) but produces results only in the last one. The sequence of new faces (and edges therein) in the localization is

$$\begin{aligned} \mathbf{V}_1\mathbf{V}_2 \text{ (origin) } \longrightarrow F_{3'}, F_{3''}, G \longrightarrow F_3, G_1, G_2, F_{1'}, F_{2'}, G'', F_{1''}, F_{2''}, G' \\ \longrightarrow \mathbf{V}_{3'}\mathbf{V}_{3''}, F_1, F_2, G_3. \end{aligned} \tag{3.4}$$

The second local wireframe of (3.4) is generated by three rigid 2D cycles:

$$C_3 = G_1F_3F_{3'}F_{3''}G_2, \quad C' = GF_{1'}F_{2'}F_{3'}G'', \quad C'' = GF_{1''}F_{2''}F_{3''}G'.$$

Then faces $F_{3'}, F_{3''}, G$ and hence edge $\mathbf{V}_1\mathbf{V}_2$ are deleted. The last local wireframe of (3.4) is generated by three 2D rigid cycles which introduce edge $\mathbf{V}_{3'}\mathbf{V}_{3''}$:

$$C = G'F_1F_2F_3G'', \quad C_1 = G_2F_1F_{1'}F_{1''}G_3, \quad C_2 = G_1F_2F_{2'}F_{2''}G_3.$$

After this all faces are deleted.

From 3D to 4D: Only in the last wireframe of (3.4) does it produce any result. The localization is

$$G \text{ (origin) } \longrightarrow C', C'' \longrightarrow C, C_1, C_2, C_3. \tag{3.5}$$

The last wireframe is generated by the rigid cycle through C', C''. Thus the final result is a 3D rigid cycle, or equivalently, a 4D face.

We have implemented all the algorithms in VC++ 6.0, have tested all the examples in [1], [14], [15], [21], in addition to higher dimensional ones made by ourselves. We We have made a comparison between our algorithm and the existing fastest algorithm for face identification of 2D manifolds in [15]. It appears that our algorithm can handle complex manifold objects of over 10,000 faces by an IBM PC of Intel 2.60GHz CPU and 248MB RAM within one and a half hours, while their algorithm has to be stopped after running many more hours.

4. Geometric Reconstruction

The geometric reconstruction from $(r-1)$D to rD can be formulated as solving either Sugihara's fundamental equations (2.7) for the unknowns h's and $\mathbf{B}$'s, or Crapo's syzygy equations (2.9) for the unknowns h's. The problem is obviously linear for numerical data.

If the 2D coordinates are symbolic, then they must satisfy some unknown realizability conditions for the geometric reconstruction. After they are found, the realizability conditions must be further *triangulated* [32] in order to obtain a sequence of explicit geometric constructions for the 2D line drawing. From this aspect, all 2D coordinates are unknowns and the equations are highly nonlinear with a large number of unknowns. This explains the difficulty in symbolic geometric reconstruction.

Without loss of generality, we set $r = 3$ in this section and focus on the 2D to 3D lift. The syzygy equations are obtained by eliminating the $\mathbf{B}$'s from the fundamental equations. To further eliminate the h's from the system, and then triangulate the equations of the 2D coordinates, is usually very difficult.

How about eliminating the h's from the fundamental equations, and then further eliminating the $\mathbf{B}$'s? The fundamental equations (2.7) can be trivially split into two systems. The first system, called the $\mathbf{B}$-*system*, or *bivector system*, is that the height of any vertex $\mathbf{V}_i$ computed from any of its incident faces $F_1, \ldots, F_k$ are the same:

$$[\mathbf{iB}_1] = [\mathbf{iB}_2] = \cdots = [\mathbf{iB}_k], \text{ for any } 1 \le j \le k. \tag{4.1}$$

Here $\mathbf{i}$ denotes the 2D coordinates of vertex $\mathbf{V}_i$. The second system, called the *h-system*, or *height system*, is that one face F_1 is sufficient to characterize the height of its vertex $\mathbf{V}_i$:

$$h_i = [\mathbf{iB}_1]. \tag{4.2}$$

The $\mathbf{B}$-*system* appears to be much easier to solve than the syzygy equations, although it has more unknowns and equations.

The $\mathbf{B}$-*system* is the starting point of our geometric reconstruction. We can use the powerful *vectorial equation-solving* [10] method to solve it. The following three formulas will used in this paper, whose proofs are easy and are omitted.

Type B.0.

$$\begin{cases} [\mathbf{V}_1\mathbf{B}] & = & [\mathbf{V}_1\mathbf{B}'] \\ [\mathbf{V}_2\mathbf{B}] & = & [\mathbf{V}_2\mathbf{B}'] \\ \mathbf{V}_1\mathbf{V}_2 & \neq & 0 \end{cases}$$

Solution: $\mathbf{B} = \mathbf{B}' + \omega_{12}\mathbf{V}_1\mathbf{V}_2$ for new parameter ω_{12}.

Type B.1.

$$\left\{\begin{array}{lcl} [\mathbf{V}_1\mathbf{B}] & = & [\mathbf{V}_1\mathbf{B}'] \\ [\mathbf{V}_2\mathbf{B}] & = & [\mathbf{V}_2\mathbf{B}'] \\ [\mathbf{V}_3\mathbf{B}] & = & [\mathbf{V}_3\mathbf{B}'] \\ [\mathbf{V}_1\mathbf{V}_2\mathbf{V}_3] & \neq & 0 \end{array}\right. \tag{4.3}$$

Solution: $\mathbf{B} = \mathbf{B}'$.

Type B.2.

$$\left\{\begin{array}{lcl} [\mathbf{V}_1\mathbf{B}] & = & [\mathbf{V}_1\mathbf{B}'] \\ [\mathbf{V}_2\mathbf{B}] & = & [\mathbf{V}_2\mathbf{B}'] \\ \ldots & & \ldots \\ [\mathbf{V}_k\mathbf{B}] & = & [\mathbf{V}_k\mathbf{B}'], \quad \text{where } k > 3 \\ [\mathbf{V}_1\mathbf{V}_2\mathbf{V}_3] & \neq & 0 \end{array}\right. \tag{4.4}$$

Solution: An expression of $\mathbf{B}$ by $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3$ and $k-3$ syzygy equations.

$$\left\{\begin{array}{l} [\mathbf{V}_1\mathbf{V}_2\mathbf{V}_3]\,\mathbf{B} = [\mathbf{V}_1\mathbf{B}']\mathbf{V}_2\mathbf{V}_3 - [\mathbf{V}_2\mathbf{B}']\mathbf{V}_1\mathbf{V}_3 + [\mathbf{V}_3\mathbf{B}']\mathbf{V}_1\mathbf{V}_2, \\ [\mathbf{V}_1\mathbf{B}'][\mathbf{V}_2\mathbf{V}_3\mathbf{V}_j] - [\mathbf{V}_2\mathbf{B}'][\mathbf{V}_1\mathbf{V}_3\mathbf{V}_j] + [\mathbf{V}_3\mathbf{B}'][\mathbf{V}_1\mathbf{V}_2\mathbf{V}_j] \\ \qquad\qquad\qquad -[\mathbf{V}_j\mathbf{B}'][\mathbf{V}_1\mathbf{V}_2\mathbf{V}_3] = 0, \text{ for } 3 < j \leq k. \end{array}\right.$$

To avoid bifurcation in vectorial equation-solving, some inequalities are necessary. In this section, we adopt the following **Practical Assumption**:
In the line drawing, no three neighboring vertices of a face are collinear.

4.1. Parametric Propagation

To solve the **B**-system, using elimination methods usually leads to complicated bifurcations, in which most branches are geometrically meaningless. We need a method to control the bifurcation, by *transforming the system into one with a minimal number of unknowns and equations.*

We propose a technique called *parametric propagation* to make the transformation. The idea is to solve the **B**-system *locally* by introducing suitable parameters, like the solving of type-**B.0** equations. The *propagation* is similar to the localization in structural reconstruction: First we choose a **B** as the "origin", and solve for other **B**'s neighboring to it when taken as faces, by introducing a minimal number of new parameters. The solved **B**'s are then put into the origin, and the propagation continues. In the end, the **B**-system is transformed into two subsystems, one with the **B**'s explicitly expressed by the new parameters, the other with the parameters only. The latter is called the *system of parameters.* The details can be read from the following example.

Example 9. The torus in Figure 6(a) is called the *Sugihara torus* [25]. Its **B**-system is as follows.

$$\begin{cases} [\mathbf{1B}_{2'}] = [\mathbf{1B}_{2''}] = [\mathbf{1B}_{3'}] = [\mathbf{1B}_{3''}] \\ [\mathbf{2B}_{1'}] = [\mathbf{2B}_{1''}] = [\mathbf{2B}_{3'}] = [\mathbf{2B}_{3''}] \\ [\mathbf{3B}_{2'}] = [\mathbf{3B}_{2''}] = [\mathbf{3B}_{1'}] = [\mathbf{3B}_{1''}] \\ [\mathbf{1'B}_{2}] = [\mathbf{1'B}_{2''}] = [\mathbf{1'B}_{3}] = [\mathbf{1'B}_{3''}] \\ [\mathbf{2'B}_{1}] = [\mathbf{2'B}_{1''}] = [\mathbf{2'B}_{3}] = [\mathbf{2'B}_{3''}] \\ [\mathbf{3'B}_{1}] = [\mathbf{3'B}_{1''}] = [\mathbf{3'B}_{2}] = [\mathbf{3'B}_{2''}] \\ [\mathbf{1''B}_{2}] = [\mathbf{1''B}_{2'}] = [\mathbf{1''B}_{3}] = [\mathbf{1''B}_{3'}] \\ [\mathbf{2''B}_{1}] = [\mathbf{2''B}_{1'}] = [\mathbf{2''B}_{3}] = [\mathbf{2''B}_{3'}] \\ [\mathbf{3''B}_{1}] = [\mathbf{3''B}_{1'}] = [\mathbf{3''B}_{2}] = [\mathbf{3''B}_{2'}] \end{cases} \tag{4.5}$$

Parametric propagation:

Round 0: Choose face $F_{3''}$ as the origin. Algebraically this is equivalent to setting $\mathbf{B}_{3''} = 0$.

Round 1: Propagate towards a neighbor of the origin, say F_3. Algebraically this is equivalent to solving a system of type **B.0**:

$$[\mathbf{1'B}_3] = 0,\ [\mathbf{2'B}_3] = 0 \xrightarrow{B.0} \mathbf{B}_3 = \omega_{1'2'}\mathbf{1'2'}.$$

Round 2: Now only $\mathbf{B}_{3'}$ can be solved without introducing new parameters, i.e., only $F_{3'}$ can be constructed geometrically. Propagation towards $F_{3'}$.

$$\begin{aligned} &[\mathbf{1B}_{3'}] = 0,\ [\mathbf{2B}_{3'}] = 0,\ [\mathbf{1''B}_{3'}] = \omega_{1'2'}[\mathbf{1'2'1''}],\ [\mathbf{2''B}_{3'}] = \omega_{1'2'}[\mathbf{1'2'2''}] \\ &\xrightarrow{B.2} [\mathbf{121''}]\mathbf{B}_{3'} = \omega_{1'2'}[\mathbf{1'2'1''}]\mathbf{12}, \quad \omega_{1'2'}\mathbf{12} \wedge \mathbf{1'2'} \wedge \mathbf{1''2''} = 0. \end{aligned}$$

A Cayley factorization [29] has been carried out in the solution.

Round 3: To construct any new face, at least one new parameter must be introduced. Propagation towards $F_{1''}$.

$$[\mathbf{2B}_{1''}] = 0,\ [\mathbf{2'B}_{1''}] = 0 \xrightarrow{B.0} \mathbf{B}_{1''} = \omega_{22'}\mathbf{22'}.$$

Round 4: Propagation towards $\mathbf{B}_{2''}$, i.e., construct $F_{2''}$ in Figure 6(a).

$$\begin{aligned} &[\mathbf{1B}_{2''}] = 0,\ [\mathbf{1'B}_{2''}] = 0,\ [\mathbf{3B}_{2''}] = -\omega_{22'}[\mathbf{232'}],\ [\mathbf{3'B}_{2''}] = \omega_{22'}[\mathbf{22'3'}] \\ &\xrightarrow{B.2} [\mathbf{131'}]\mathbf{B}_{2''} = \omega_{22'}[\mathbf{232'}]\mathbf{11'}, \quad \omega_{22'}\mathbf{11'} \wedge \mathbf{22'} \wedge \mathbf{33'} = 0. \end{aligned}$$

Rounds 5-8: Propagations towards $\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{B}_{1'}$, $\mathbf{B}_{2'}$. They are similar to Round 4.

Solution: By setting $\mathbf{B}_{3''} = 0$ and using new parameters $\omega_{1'2'}, \omega_{22'}$, the original $\mathbf{B}$-system is changed into

$$\left\{\begin{aligned} \mathbf{B}_{3''} &= 0, \\ \mathbf{B}_3 &= \omega_{1'2'}\mathbf{1'2'}, \\ [\mathbf{121''}]\mathbf{B}_{3'} &= \omega_{1'2'}[\mathbf{1'2'1''}]\mathbf{12}, \\ \mathbf{B}_{1''} &= \omega_{22'}\mathbf{22'}, \\ [\mathbf{131'}]\mathbf{B}_{2''} &= \omega_{22'}[\mathbf{232'}]\mathbf{11'}, \\ [\mathbf{2'3'2''}]\mathbf{B}_1 &= \omega_{1'2'}[\mathbf{1'2'2''}]\mathbf{2'3'} - \omega_{22'}[\mathbf{22'3'}]\mathbf{2'2''}, \\ [\mathbf{1'3'1''}]\mathbf{B}_2 &= \omega_{1'2'}[\mathbf{1'2'1''}]\mathbf{1'3'} - \omega_{22'}[\mathbf{22'3'}]\mathbf{1'1''}, \\ [\mathbf{232''}]\mathbf{B}_{1'} &= \omega_{1'2'}[\mathbf{1'2'2''}]\mathbf{23} + \omega_{22'}[\mathbf{232'}]\mathbf{22''}, \\ [\mathbf{131''}]\mathbf{B}_{2'} &= \omega_{1'2'}[\mathbf{1'2'1''}]\mathbf{13} + \omega_{22'}[\mathbf{232'}]\mathbf{11''}, \end{aligned}\right. \tag{4.6}$$

together with

$$\left\{\begin{aligned} \omega_{22'}\mathbf{11'} \wedge \mathbf{22'} \wedge \mathbf{33'} &= 0, \\ \omega_{1'2'}\mathbf{12} \wedge \mathbf{1'2'} \wedge \mathbf{1''2''} &= 0, \\ (\omega_{1'2'}[\mathbf{1'2'3'}] - \omega_{22'}[\mathbf{22'3'}])\mathbf{1'1''} \wedge \mathbf{2'2''} \wedge \mathbf{3'3''} &= 0, \\ (\omega_{1'2'}[\mathbf{1'2'2''}] - \omega_{22'}[\mathbf{22'2''}])\ \mathbf{23} \wedge \mathbf{2'3'} \wedge \mathbf{2''3''} &= 0, \\ \omega_{22'}([\mathbf{11''3''}][\mathbf{232'}][\mathbf{2'3'2''}] + [\mathbf{131''}][\mathbf{22'3'}][\mathbf{2'2''3''}]) & \\ = \omega_{1'2'}([\mathbf{131''}][\mathbf{1'2'2''}][\mathbf{2'3'3''}] - [\mathbf{133''}][\mathbf{1'2'1''}][\mathbf{2'3'2''}]). & \end{aligned}\right. \tag{4.7}$$

By parametric propagation, we have successfully reduced the 27 equations in (4.5) to the 5 equations in (4.7), and reduced the 8 bivector unknowns to 2 scalar unknowns.

The next task is to solve for the ω's from (4.7) so that the 2D faces given by (4.6) constitute a 2D manifold in 3D.

4.2. Calotte Factorization

In (4.7), the first 4 equations are in factored form, so each can be decomposed into two equations, one with the parameters and the other without. This is a great simplification. Although the last equation in (4.7) itself cannot be factored, it becomes so if the other 4 equations are used.

The algebraic factorization for the last equation is very delicate. Below we propose a very simple technique called *calotte factorization*, to produce solutions in factored form for equations of type **B.2**, using the *local geometric information.* It is a geometric method for algebraic factorization. Although this technique only applies to the reconstruction from 2D to 3D, it is general enough and very effective in simplifying the system of parameters.

The idea is inspired by Crapo's work [5], where it is suggested that for bracket polynomials produced in geometric computation, the geometric background can help finding rational Cayley factorization. Let F be a face of m vertices, called the *center.* A *regular m-calotte* centered at F [5] is defined as a 2D F-cycle. The faces of the cycle are the *circumfaces.* Let $1, 2, \ldots, m$ be the vertices of the center, and let $11', 22', \ldots, mm'$ be the edges shared by neighboring circumfaces. Then the calotte can be denoted by $(12\ldots m,\ 1'2'\ldots m')$. In a *general calotte*, the center may not be a face.

Crapo's Theorem. (*cf.* [5]) *Let* $(12\ldots m,\ 1'2'\ldots m')$ *be a regular* m*-calotte, let the inhomogeneous coordinates of the center and the circumfaces be* $\mathbf{B}_0$ *and* $\mathbf{B}_i$ *for* $1 \le i \le m$*. If the center and* $m-1$ *circumfaces have been constructed, then the last circumface can be constructed if and only if*

$$(\mathbf{B}_i - \mathbf{B}_0)c = 0, \quad \textit{or equivalently,} \quad (\mathbf{B}_{i+1} - \mathbf{B}_i)c = 0 \ \ \textit{for some} \ \ 1 \le i \le m, \tag{4.8}$$

where

$$\begin{aligned} c = &-[\mathbf{1'12}][\mathbf{2'23}]\cdots[(\mathbf{m-1})'(\mathbf{m-1})\mathbf{m}][\mathbf{m'm1}] \\ &+[\mathbf{2'12}][\mathbf{3'23}]\cdots[\mathbf{m'(m-1)m}][\mathbf{1'm1}] \end{aligned}$$

is the Crapo binomial. The equation in (4.8) is called the calotte equation.

For a general 3-calotte, no matter if its center is a face or not, the calotte equation is

$$(\mathbf{B}_2 - \mathbf{B}_1)\mathbf{11'} \wedge \mathbf{22'} \wedge \mathbf{33'} = 0. \tag{4.9}$$

Calotte factorization is to replace the syzygy equation in the solution of type-**B.2** equations by a calotte equation, using Crapo's Theorem in the procedure of parametric propagation. If several calotte equations are available, then any of them will do.

Example 10. In the parametric propagation of Example 9, the calottes are produced along with the construction of new faces:

$$\begin{aligned} \mathbf{B}_{3''} \ &\longrightarrow \mathbf{B}_3 \\ &\longrightarrow \mathbf{B}_{3'},\ \text{calotte } (11'1'', 22'2'') \\ &\longrightarrow \mathbf{B}_{1''}, \mathbf{B}_{2''},\ \text{calotte } (123, 1'2'3') \\ &\longrightarrow \mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_{1'}, \mathbf{B}_{2'},\ \text{calottes} \\ &\qquad (1'2'3', 1''2''3''), (22'2'', 33'3''), (11'1'', 33'3''), (123, 1''2''3''). \end{aligned}$$

The equations of the first 4 calotte are exactly the first 4 equations in (4.7) when applying (4.6). The last equation in (4.7) is a syzygy equation obtained by solving for $\mathbf{B}_{2'}$, i.e., constructing $F_{2'}$, in the last round of propagation. By Crapo's Theorem, this equation can be replaced by the calotte equation of $(11'1'', 33'3'')$. The result after applying (4.6) is

$$\begin{aligned} (\omega_{1'2'}[\mathbf{131'}][\mathbf{1'2'1''}]\mathbf{1'3'} - \omega_{22'}([\mathbf{131'}][\mathbf{22'3'}]\mathbf{1'1''} + [\mathbf{1'3'1''}][\mathbf{232'}]\mathbf{11'})) \\ \mathbf{13} \wedge \mathbf{1'3'} \wedge \mathbf{1''3''} = 0. \end{aligned} \tag{4.10}$$

4.3. Solving the System of Parameters

Now we are ready to solve the system of parameters. We illustrate this by solving (4.7). Since the reconstruction result is a manifold, both $\omega_{1'2'}$ and $\omega_{22'}$ must be nonzero.

Lemma 1. *If* $\omega_{1'2'}, \omega_{22'}$ *are both nonzero, then at most one of the following three equalities holds:*

$$\omega_{1'2'}[\mathbf{131'}][\mathbf{1'2'1''}]\mathbf{1'3'} - \omega_{22'}([\mathbf{131'}][\mathbf{22'3'}]\mathbf{1'1''} + [\mathbf{1'3'1''}][\mathbf{232'}]\mathbf{11'}) = 0 \tag{4.11}$$

$$\omega_{1'2'}[\mathbf{1'2'2''}] - \omega_{22'}[\mathbf{22'2''}] = 0 \tag{4.12}$$

$$\omega_{1'2'}[\mathbf{1'2'3'}] - \omega_{22'}[\mathbf{22'3'}] = 0 \tag{4.13}$$

After deleting nonzero factors, the system of parameters is decomposed into three subsystems:

$$\left\{\begin{array}{l}\mathbf{11'}\wedge\mathbf{22'}\wedge\mathbf{33'}=0\\ \mathbf{1'1''}\wedge\mathbf{2'2''}\wedge\mathbf{3'3''}=0\\ \mathbf{12}\wedge\mathbf{1'2'}\wedge\mathbf{1''2''}=0\\ \mathbf{23}\wedge\mathbf{2'3'}\wedge\mathbf{2''3''}=0\end{array}\right.,\quad \left\{\begin{array}{l}\mathbf{11'}\wedge\mathbf{22'}\wedge\mathbf{33'}=0\\ \mathbf{12}\wedge\mathbf{1'2'}\wedge\mathbf{1''2''}=0\\ \mathbf{13}\wedge\mathbf{1'3'}\wedge\mathbf{1''3''}=0\\ \mathbf{23}\wedge\mathbf{2'3'}\wedge\mathbf{2''3''}=0\end{array}\right.,\quad \left\{\begin{array}{l}\mathbf{11'}\wedge\mathbf{22'}\wedge\mathbf{33'}=0\\ \mathbf{1'1''}\wedge\mathbf{2'2''}\wedge\mathbf{3'3''}=0\\ \mathbf{12}\wedge\mathbf{1'2'}\wedge\mathbf{1''2''}=0\\ \mathbf{13}\wedge\mathbf{1'3'}\wedge\mathbf{1''3''}=0\end{array}\right. \tag{4.14}$$

Lemma 2. *If*

$$\left\{\begin{array}{r}\mathbf{12}\wedge\mathbf{1'2'}\wedge\mathbf{1''2''}=0\\ \mathbf{23}\wedge\mathbf{2'3'}\wedge\mathbf{2''3''}=0\\ \mathbf{11'}\wedge\mathbf{22'}\wedge\mathbf{33'}=0\\ \mathbf{1'1''}\wedge\mathbf{2'2''}\wedge\mathbf{3'3''}=0\end{array}\right. \tag{4.15}$$

then $\mathbf{13}\wedge\mathbf{1'3'}\wedge\mathbf{1''3''}=0,\ \mathbf{11''}\wedge\mathbf{22''}\wedge\mathbf{33''}=0$. *If*

$$\left\{\begin{array}{l}\mathbf{12}\wedge\mathbf{1'2'}\wedge\mathbf{1''2''}=0\\ \mathbf{13}\wedge\mathbf{1'3'}\wedge\mathbf{1''3''}=0\\ \mathbf{23}\wedge\mathbf{2'3'}\wedge\mathbf{2''3''}=0\\ \mathbf{11''}\wedge\mathbf{22''}\wedge\mathbf{33''}=0\end{array}\right. \tag{4.16}$$

then $\mathbf{11'}\wedge\mathbf{22'}\wedge\mathbf{33'}=0,\ \mathbf{1'1''}\wedge\mathbf{2'2''}\wedge\mathbf{3'3''}=0$ *under the condition* $[\mathbf{1'2'3'}]\neq 0$.

By this lemma, in the first and third subsystems of (4.14), the six tuples

$$\begin{array}{ccc}(\mathbf{12},\mathbf{1'2'},\mathbf{1''2''}), & (\mathbf{13},\mathbf{1'3'},\mathbf{1''3''}), & (\mathbf{23},\mathbf{2'3'},\mathbf{2''3''}),\\ (\mathbf{11'},\mathbf{22'},\mathbf{33'}), & (\mathbf{11''},\mathbf{22''},\mathbf{33''}), & (\mathbf{1'1''},\mathbf{2'2''},\mathbf{3'3''})\end{array} \tag{4.17}$$

are all concurrent lines. The corresponding configuration is called the *Triple Desargues Configuration*. By Desargues Theorem, the six intersections $\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{A}'$, $\mathbf{B}',\mathbf{C}'$ of the six tuples lie on two lines, with each line passing through three intersections. In this configuration, both $\omega_{1'2'},\omega_{22'}$ are free of any equality constraint, and according to (4.6), the line drawing can be lifted to a spatial torus.

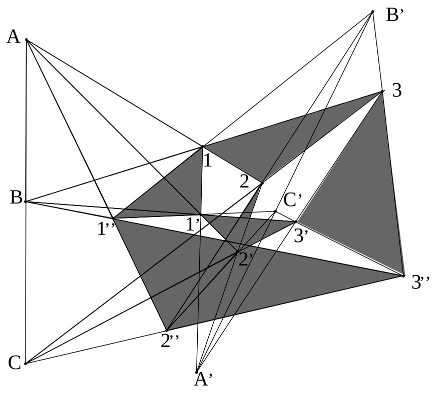

Fig. 7. Triple Desargues configuration.

For the second subsystem of (4.14), if it is not the Triple Desargues Configuration, then $[\mathbf{1''2''3''}] = 0$ and $\omega_{1'2'}[\mathbf{1'2'3'}] = \omega_{22'}[\mathbf{22'3'}]$. By (4.6), $\mathbf{B}_1 = \mathbf{B}_2 = \mathbf{B}_3$, the solution is not a manifold.

Theorem on Sugihara Torus. *A line drawing of Sugihara torus with vertices* $\mathbf{i}, \mathbf{i}', \mathbf{i}''$ *for* $1 \le i \le 9$ *is realizable in 3D if and only if it is a Triple Desargues Configuration, i.e., if and only if the following are concurrent 3-tuples of image lines:*

$$(\mathbf{ij}, \mathbf{i'j'}, \mathbf{i''j''}), \quad (\mathbf{ii'}, \mathbf{ii''}, \mathbf{i'i''}), \quad \text{for } 1 \le i < j \le 3.$$

The configuration space of all Sugihara tori allows global parametrization. Let $\mathbf{2}, \mathbf{3}, \mathbf{1}', \mathbf{2}', \mathbf{3}', \mathbf{1}'', \mathbf{2}''$ *be free points in the plane, let* $\mathbf{B}_{3''}$ *be a free bivector in the image plane and let* $\omega_{1'2'}, \omega_{22'}$ *be free parameters, then the following is a global parametrization.*

$$\left\{\begin{aligned}
\mathbf{1} &= \mathbf{2}(\mathbf{1'2'} \wedge \mathbf{1''2''}) \wedge \mathbf{1}'(\mathbf{22'} \wedge \mathbf{33'})\\
\mathbf{3}'' &= (\mathbf{23} \wedge \mathbf{2'3'})\mathbf{2}'' \wedge (\mathbf{1'1''} \wedge \mathbf{2'2''})\mathbf{3}'\\
\mathbf{B}_3 &= \mathbf{B}_{3''} + \omega_{1'2'}\mathbf{1'2'}\\
\mathbf{B}_{3'} &= \mathbf{B}_{3''} + \omega_{1'2'}([\mathbf{1'2'1''}]/[\mathbf{121''}])\mathbf{12}\\
\mathbf{B}_{1''} &= \mathbf{B}_{3''} + \omega_{22'}\mathbf{22'}\\
\mathbf{B}_{2''} &= \mathbf{B}_{3''} + \omega_{22'}([\mathbf{232'}]/[\mathbf{131'}])\mathbf{11'}\\
\mathbf{B}_1 &= \mathbf{B}_{3''} + \omega_{1'2'}([\mathbf{1'2'2''}]/[\mathbf{2'3'2''}])\mathbf{2'3'} - \omega_{22'}([\mathbf{22'3'}]/[\mathbf{2'3'2''}])\mathbf{2'2''}\\
\mathbf{B}_2 &= \mathbf{B}_{3''} + \omega_{1'2'}([\mathbf{1'2'1''}]/[\mathbf{1'3'1''}])\mathbf{1'3'} - \omega_{22'}([\mathbf{22'3'}]/[\mathbf{1'3'1''}])\mathbf{1'1''}\\
\mathbf{B}_{1'} &= \mathbf{B}_{3''} + \omega_{1'2'}([\mathbf{1'2'2''}]/[\mathbf{232''}])\mathbf{23} + \omega_{22'}([\mathbf{232'}]/[\mathbf{232''}])\mathbf{22''}\\
\mathbf{B}_{2'} &= \mathbf{B}_{3''} + \omega_{1'2'}([\mathbf{1'2'1''}]/[\mathbf{131''}])\mathbf{13} + \omega_{22'}([\mathbf{232'}]/[\mathbf{131''}])\mathbf{11''}\\
h_1 &= [\mathbf{1B}_{3''}], \quad h_2 = [\mathbf{2B}_{3''}], \quad h_3 = [\mathbf{3B}_{1''}]\\
h_{1'} &= [\mathbf{1'B}_{3''}], \; h_{2'} = [\mathbf{2'B}_{3''}], \; h_{3'} = [\mathbf{3'B}_{1''}]\\
h_{1''} &= [\mathbf{1''B}_3], \; h_{2''} = [\mathbf{2''B}_3], \; h_{3''} = [\mathbf{3''B}_1]
\end{aligned}\right. \tag{4.18}$$

(4.18) is in fact a *linear construction sequence* of the torus, i.e., a triangular form in which the polynomials are linear with respect to their leading variables. Algebraically, all the properties of resolvable sequences [26] needed in application are occupied by linear construction sequences.

Although a Sugihara torus is composed of three triangular columns as in Figure 2, the condition that the three columns can be reconstructed into manifolds does not guarantee that the torus can. What is more useful is the 4D structure, or equivalently, the 3D manifold structure of Figure 6(a). There are 6 triangular columns which form a cycle of 3D faces. In 3D one column is trivial and can be removed. That the torus can be reconstructed in 3D is equivalent to that the 5 columns can be respectively reconstructed in 3D, following Lemma 1 and 2. Further connection between high dimensional structural reconstruction and geometric reconstruction is a topic of our future research.

4.4. Higher Dimensional Case

For geometric reconstruction from 2D to nD where $n > 3$, the number of non-trivial incidence relations grows rapidly, and as a consequence, the system of parameters is much more complicated. By *trivial incidence relations* we mean those that can be deduced from lower dimensional ones.

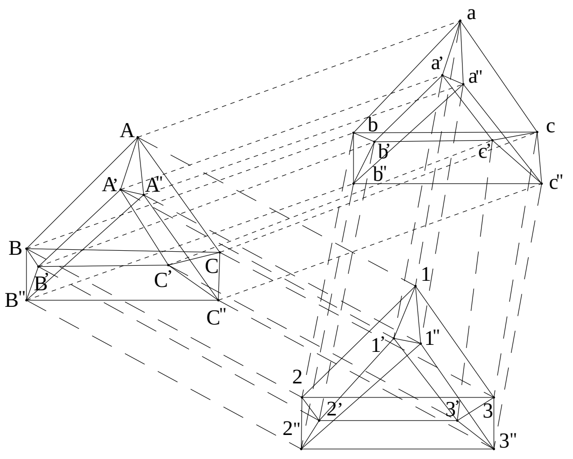

Fig. 8. 3D Sugihara torus.

Example 11. Since $T^r = \overbrace{S^1 \times S^1 \times \cdots \times S^1}^{r}$, when representing S^1 with a triangular cycle, we obtain a wireframe of T^r, called *rD Sugihara torus.*

For $r = 3$, there are 81 nontrivial 2D coplanarity constraints, and 3 nontrivial 3D coplanarity constraints. In Figure 8, the 3D coplanarities are that $1231'2'3'1''2''3''$, $abca'b'c'a''b''c''$ and $ABCA'B'C'A''B''C''$ are respectively 3D.

In geometric reconstruction from 2D to 3D, 4 new parameters are introduced, and are all free ones in manifold parametrization. From 3D to 4D, 2 new parameters are introduced, and are both free ones in manifold parametrization.

The parametric propagation and calotte factorization algorithms have been implemented with Maple 8 and tested by more than 20 examples.

5. Conclusion

In this paper, we study the fascinating problem of nD polyhedral scene reconstruction from a single 2D line drawing, from both structural and geometric aspects. We propose a number of very efficient techniques for the reconstructions, and test them by common examples for $n = 3$, showing the superiority of our algorithms. We believe our algorithms are the first for $n > 3$, and have a lot of room for further improvement despite their current efficiency.

References

[1] S. C. Agarwel and W. Waggenspack. Decomposition method for extracting face topologies from wireframe models. *Computer Aided Design* **24**(3): 123–140, 1992.

[2] J. Bokowski and B. Sturmfels. *Computational Synthetic Geometry*, **LNM 1355**, Springer, Berlin, Heidelberg, 1989.

[3] S. Courter and J. Brewer. Automated conversion of curvilinear wireframe models to surface boundary models. *Comput. Graph.* **20**(4): 171–178, 1986.

[4] H. Crapo and W. Whiteley. Stresses on Frameworks and Motions of Panel Structures: a Projective Geometric Introduction. *Structural Topology* **6**: 42-82, 1982.

[5] H. Crapo. Invariant-Theoretic Methods in Scene Analysis and Structural Mechanics. *J. of Symbolic Computation* **11**: 523-548, 1991.

[6] M. Ganter and J. Uicker. From wireframe to solid geometric: Automated conversion of data representations. *Computer in Mechanical Eng.* **2**(2): 40–45, 1983.

[7] P. Hanrahan. Creating volume models from edge-vertex Graphs. *Computer Graphics* **16**(3): 77–84, 1982.

[8] D. Huffman. Realizable Configurations of Lines in Pictures of Polyhedra, *Machine Intelligence* **8**: 493-509, 1977.

[9] Y. Leclerc and M. Fischler. An optimization based approach to the interpretation of single line drawings as 3d wireframes. *International J. Computer Vision* **8(2)**: 113–136, 1992.

[10] H. Li. Vectorial Equation-Solving for Mechanical Geometry Theorem Proving. *J. Automated Reasoning* **25**: 83-121, 2000.

[11] H. Li and G. Sommer. Coordinate-free projective geometry for computer vision. In: *Geometric Computing with Clifford Algebra*, G. Sommer (ed.), Springer Berlin Heidelberg, pp. 415-454, 2001.

[12] H. Li and L. Zhao. A symbolic approach to polyhedral scene analysis by parametric calotte propagation, *MM Research Preprints* **22**: 194-230, 2003. Available: http://www.mmrc.iss.ac.cn/pub/mm22.pdf/13.pdf.

[13] H. Li, Q. Wang, L. Zhao, Y. Chen and L. Huang. nD object representation and detection from single 2D line drawing, *MM Research Preprints* **23**: 178-195, 2004. Available: http://www.mmrc.iss.ac.cn/pub/mm23.pdf/hli-2.pdf

[14] J. Liu and Y. Lee. A graph-based method for face identification from a single 2d line drawing. *IEEE Trans. on PAMI* **23(10)**: 1106–1119, 2001.

[15] J. Liu, Y. Lee and C. Cham. Identifying faces in a 2d line drawing representing a manifold object. *IEEE Trans. on PAMI* **24(12)**: 1579–1593, 2002.

[16] A. Mackworth. Interpreting Pictures of Polyhedral Scenes, *Artificial Intelligence* **4**: 121-137, 1973.

[17] T. Marill. Emulation the human interpretation of line drawings as 3d objects. *International J. of Computer Vision* **6**(2): 147–161, 1991.

[18] K. Miyazaki. *An Adventure in Multidimensional Space: The Art and Geometry of Polygons, Polyhedra, and Polytopes.* Wileyinterscience Publ, 1983.

[19] J. Richter-Gebert. *Realization Spaces of Polytopes.* **LNM 1643**, Springer, Wien, 1996.

[20] L. Ros and F. Thomas. Overcoming Superstrictness in Line Drawing Interpretation. *IEEE Trans. PAMI* **24(4)**: 456-466, 2002.

[21] M. Shpitalni and H. Lipson. Identification of faces in a 2d line drawing projection of a wireframe object. *IEEE Trans. on PAMI* **18(10)**: 1000–1012, 1996.

[22] B. Sturmfels. *Algorithms in Invariant Theory.* Springer, Wien, 1993.

[23] K. Sugihara. Mathematical Structures of Line Drawings of Polyhedrons – Towards Man-Machine Communication by Means of Line Drawings. *IEEE Trans. on PAMI* **4**: 458-469, 1982.

[24] K. Sugihara. An Algebraic Approach to Shape-from-Image Problems, *Artificial Intelligence* **23**: 59-95, 1984.

[25] K. Sugihara. *Machine Interpretation of Line Drawings*, MIT Press, Cambridge, Mass, 1986.

[26] K. Sugihara. Resolvable Representation of Polyhedra. *Discrete Computational Geometry* **21(2)**: 243-255, 1999.

[27] A. Turner, D. Chapman and A. Penn. Sketching space. *Computers and Graphics* **24**, 869–879, 2000.

[28] N. White and W. Whiteley. The Algebraic Geometry of Stresses in Frameworks, *SIAM J. Alg. Discrete Math.* **4**: 481-511, 1983.

[29] N. White. Multilinear Cayley Factorization. *J. of Symbolic Computation* **11**: 421-438, 1991.

[30] W. Whiteley. From a Line Drawing to a Polyhedron. *J. Math. Psych.* **31**: 441-448, 1987.

[31] W. Whiteley. Matroids and Rigid Structures. In *Matroid Applications*, N. White *ed.*, Encyclopedia of Mathematics and Its Applications **40**, pp. 1-53, Cambridge University Press, Cambridge, 1992.

[32] W.-T. Wu. *Mathematics Mechanization*, Science Press and Kluwer Academic Publishers, Beijing 2000.